# Contents

1 General Information ................................................. 9

2 HOTINT User Manual ............................................... 47
   2.1 Multibody formulation ......................................... 48
      2.1.1 Solution vector ........................................... 48
      2.1.2 Main structure of the multibody kernel .................. 48
      2.1.3 Object library ............................................. 50
      2.1.4 The dynamic solver – implicit time integration ....... 50
      2.1.5 The static solver – incremental loading ............... 52
      2.1.6 Eigenmode computation .................................. 52
      2.1.7 Parameter Variation, Sensitivity Analysis, Identification and Optimization .................. 57
      2.1.8 The Element Concept ..................................... 63
      2.1.9 Nodes for Direct Connection of Finite Elements ....... 64
      2.1.10 The Concept of Loads .................................... 64
      2.1.11 Sensors for Measuring ................................... 65
      2.1.12 Geometric Elements for Bodies with Complex Geometry ......................... 65
   2.2 Getting started ................................................. 66
      2.2.1 Instructions for installing HOTINT on a MS-Windows computer ......................... 66
      2.2.2 First steps ................................................. 69
      2.2.3 Command Line Usage ...................................... 70
      2.2.4 Configure Notepad ++ for HOTINT ......................... 71
   2.3 HOTINT Windows User Interface ................................. 73
      2.3.1 Using the graphics window ................................ 73
      2.3.2 Mouse control ............................................. 73
      2.3.3 HOTINT main application window ......................... 73
      2.3.4 Specific buttons ........................................... 74
      2.3.5 HOTINT Main Menu ........................................ 75
   2.4 Creating your model in HOTINT ................................. 80
      2.4.1 Introduction .............................................. 80
      2.4.2 Model setup via the script language ..................... 80
      2.4.3 Model setup via the graphical user interface .......... 84
   2.5 Options Dialogs .................................................. 86
      2.5.1 Introduction .............................................. 86
      2.5.2 Hotint Options ........................................... 86
      2.5.3 Viewing Options ......................................... 89
      2.5.4 OpenGL Drawing Options ................................ 90
      2.5.5 Finite Element Drawing Options ......................... 92
      2.5.6 Body / Joint Options .................................... 94
      2.5.7 Data Manager ............................................. 96
## 2.5.8 Solver Options ............................................. 97

## 2.6 Data visualization and graphics export ......................................................... 99

## 2.7 Visualization Tool .......... 99

### 2.7.1 How to record a video ........................................... 101

## 2.8 HOTINT File and Folder Structure ............................................................... 103

### 2.8.1 Input Files .................................................. 103

### 2.8.2 Folder Structure .................................................. 103

## 3 HOTINT Reference Manual ............. 105

### 3.1 Preface .................................................. 105

#### 3.1.1 Examples ............................................. 105

#### 3.1.2 Data objects ............................................. 105

#### 3.1.3 Observable FieldVariables ............................................. 105

#### 3.1.4 Observable special values ............................................. 105

#### 3.1.5 Controllable special values ............................................. 105

### 3.2 Element ............................................. 106

#### 3.2.1 Mass1D ............................................. 106

#### 3.2.2 Rotor1D ............................................. 109

#### 3.2.3 Mass2D ............................................. 112

#### 3.2.4 Rigid2D ............................................. 115

#### 3.2.5 Mass3D ............................................. 118

#### 3.2.6 NodalDiskMass3D ............................................. 120

#### 3.2.7 Rigid3D ............................................. 122

#### 3.2.8 Rigid3DKardan ............................................. 126

#### 3.2.9 Rigid3DMinCoord ............................................. 129

#### 3.2.10 LinearBeam3D ............................................. 133

#### 3.2.11 RotorBeamXAxis ............................................. 137

#### 3.2.12 ANCFBeamShear3DLinear ............................................. 141

#### 3.2.13 ANCFBeamShear3DQuadratic ............................................. 147

#### 3.2.14 ANCFBeam3DTorsion ............................................. 153

#### 3.2.15 Hexahedral ............................................. 157

#### 3.2.16 Tetrahedral ............................................. 161

#### 3.2.17 Prism ............................................. 164

#### 3.2.18 Pyramid ............................................. 168

### 3.3 Connector ............................................. 173

#### 3.3.1 PointJoint ............................................. 174

#### 3.3.2 CoordinateConstraint ............................................. 179

#### 3.3.3 VelocityCoordinateConstraint ............................................. 182

#### 3.3.4 MultiCoordConstraint ............................................. 185

#### 3.3.5 SlidingPointJoint ............................................. 188

#### 3.3.6 SlidingPrismaticJoint ............................................. 192

#### 3.3.7 Rope3D ............................................. 195

#### 3.3.8 FrictionConstraint ............................................. 198

#### 3.3.9 Contact1D ............................................. 202

#### 3.3.10 PlaneConstraint ............................................. 205

#### 3.3.11 GenericBodyJoint ............................................. 210

#### 3.3.12 RevoluteJoint ............................................. 216

#### 3.3.13 PrismaticJoint ............................................. 219

#### 3.3.14 UniversalJoint ............................................. 222
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8.5</td>
<td>Moment Vector 3D</td>
<td>331</td>
</tr>
<tr>
<td>3.8.6</td>
<td>Gravity</td>
<td>332</td>
</tr>
<tr>
<td>3.8.7</td>
<td>Surface Pressure</td>
<td>333</td>
</tr>
<tr>
<td>3.8.8</td>
<td>Body Load Spatial</td>
<td>333</td>
</tr>
<tr>
<td>3.9</td>
<td>Sensor</td>
<td>335</td>
</tr>
<tr>
<td>3.9.1</td>
<td>FV Element Sensor</td>
<td>335</td>
</tr>
<tr>
<td>3.9.2</td>
<td>Element Sensor</td>
<td>336</td>
</tr>
<tr>
<td>3.9.3</td>
<td>Load Sensor</td>
<td>337</td>
</tr>
<tr>
<td>3.9.4</td>
<td>Multiple Sensor</td>
<td>339</td>
</tr>
<tr>
<td>3.9.5</td>
<td>System Sensor</td>
<td>339</td>
</tr>
<tr>
<td>3.9.6</td>
<td>FV Global Position Sensor</td>
<td>341</td>
</tr>
<tr>
<td>3.10</td>
<td>Sensor Processors</td>
<td>343</td>
</tr>
<tr>
<td>3.11</td>
<td>Geom Element</td>
<td>344</td>
</tr>
<tr>
<td>3.11.1</td>
<td>Geom Mesh 3D</td>
<td>344</td>
</tr>
<tr>
<td>3.11.2</td>
<td>Geom Cylinder 3D</td>
<td>345</td>
</tr>
<tr>
<td>3.11.3</td>
<td>Geom Sphere 3D</td>
<td>345</td>
</tr>
<tr>
<td>3.11.4</td>
<td>Geom Cube 3D</td>
<td>346</td>
</tr>
<tr>
<td>3.11.5</td>
<td>Geom Ortho Cube 3D</td>
<td>346</td>
</tr>
<tr>
<td>3.12</td>
<td>Set</td>
<td>348</td>
</tr>
<tr>
<td>3.12.1</td>
<td>Element Set</td>
<td>348</td>
</tr>
<tr>
<td>3.12.2</td>
<td>Global Node Set</td>
<td>348</td>
</tr>
<tr>
<td>3.12.3</td>
<td>Local Node Set A</td>
<td>349</td>
</tr>
<tr>
<td>3.12.4</td>
<td>Local Node Set B</td>
<td>349</td>
</tr>
<tr>
<td>3.12.5</td>
<td>Global Coord Set</td>
<td>349</td>
</tr>
<tr>
<td>3.12.6</td>
<td>Local Coord Set A</td>
<td>349</td>
</tr>
<tr>
<td>3.12.7</td>
<td>Local Coord Set B</td>
<td>350</td>
</tr>
<tr>
<td>3.12.8</td>
<td>Face Set A</td>
<td>350</td>
</tr>
<tr>
<td>3.12.9</td>
<td>Sensor Set</td>
<td>350</td>
</tr>
<tr>
<td>3.13</td>
<td>Mesh</td>
<td>352</td>
</tr>
<tr>
<td>3.13.1</td>
<td>Structural Mesh</td>
<td>353</td>
</tr>
<tr>
<td>3.13.2</td>
<td>Solid Mesh</td>
<td>354</td>
</tr>
<tr>
<td>3.14</td>
<td>Mesh Components</td>
<td>355</td>
</tr>
<tr>
<td>3.14.1</td>
<td>Primitive: Block</td>
<td>356</td>
</tr>
<tr>
<td>3.14.2</td>
<td>Primitive: Cylinder</td>
<td>357</td>
</tr>
<tr>
<td>3.14.3</td>
<td>Primitive: Quadrilateral</td>
<td>358</td>
</tr>
<tr>
<td>3.14.4</td>
<td>Primitive: Curve</td>
<td>359</td>
</tr>
<tr>
<td>3.14.5</td>
<td>Extended: Mirror</td>
<td>359</td>
</tr>
<tr>
<td>3.14.6</td>
<td>Extended: Extrude</td>
<td>360</td>
</tr>
<tr>
<td>3.14.7</td>
<td>Extended: Rotational</td>
<td>360</td>
</tr>
<tr>
<td>3.14.8</td>
<td>Extended: Lin2Quad</td>
<td>361</td>
</tr>
<tr>
<td>3.14.9</td>
<td>Extended: SplitHexes</td>
<td>362</td>
</tr>
<tr>
<td>3.14.10</td>
<td>Extended: Refine</td>
<td>362</td>
</tr>
<tr>
<td>3.14.11</td>
<td>Process: Transform</td>
<td>363</td>
</tr>
<tr>
<td>3.14.15</td>
<td>Loader: NetGen 2D</td>
<td>365</td>
</tr>
<tr>
<td>3.14.16</td>
<td>Loader: NetGen 3D</td>
<td>366</td>
</tr>
<tr>
<td>3.14.17</td>
<td>Loader: Neutral 3D</td>
<td>366</td>
</tr>
</tbody>
</table>
CONTENTS

3.14.18 Loader: STL .......................................................... 367
3.14.19 Loader: DataArrays ............................................... 367
3.14.20 Refinement ............................................................ 369
3.14.21 MeshElements ....................................................... 371

3.15 Command ................................................................. 375
3.15.1 AddElement ........................................................... 377
3.15.2 AddGeomElement ..................................................... 378
3.15.3 AssignGeomElementToElement .................................. 378
3.15.4 AddConnector ......................................................... 380
3.15.5 AddLoad ............................................................... 381
3.15.6 AddSensor ............................................................. 381
3.15.7 AddSensorProcessor ................................................. 382
3.15.8 AddMaterial .......................................................... 383
3.15.9 AddBeamProperties ................................................ 383
3.15.10 AddNode ............................................................. 384
3.15.11 Include ............................................................... 384
3.15.12 Print ................................................................. 385
3.15.13 PrintIf ............................................................ 386
3.15.14 ReadSTLFile ......................................................... 386
3.15.15 RotMat2Angles ...................................................... 387
3.15.16 LoadVectorFromFile ............................................. 387
3.15.17 TransformPoints .................................................... 388
3.15.18 ComputeInertia ..................................................... 389
3.15.19 Sum ................................................................. 390
3.15.20 Product ............................................................ 391
3.15.21 Transpose .......................................................... 392
3.15.22 CrossProduct ....................................................... 392
3.15.23 for ................................................................. 392
3.15.24 if ............................................................... 394
3.15.25 GenerateNewMesh ................................................ 395
3.15.26 GenerateBeam ...................................................... 396
3.15.27 GeneratePlate ...................................................... 397
3.15.28 GenerateBlock ..................................................... 399
3.15.29 GenerateCylinder .................................................. 400
3.15.30 LoadMesh .......................................................... 402
3.15.31 WriteMesh .......................................................... 403
3.15.32 Transform .......................................................... 403
3.15.33 Distort ............................................................. 405
3.15.34 Modify ............................................................ 406
3.15.35 Linear2Quadratic ................................................ 407
3.15.36 SplitHexes .......................................................... 408
3.15.37 Refine ............................................................ 409
3.15.38 Rotate ............................................................. 411
3.15.39 Mirror ............................................................. 412
3.15.40 Extrude ............................................................ 413
3.15.41 AddMeshToMBS .................................................... 414
3.15.42 GetNodesInBox .................................................... 415
3.15.43 GetNodesInCylinder .............................................. 416
3.15.44 GetNodesInSphere ................................................. 418
Chapter 1

General Information

Introduction

Development history and background information

The code HOTINT has been initiated by Johannes Gerstmayr in 1997 and, until now, gone over the following steps:

- solution methods and basic linear algebra routines for static solver (diploma thesis of the main developer, 1997)
- addition of time integration methods for the accurate solution of large-scale flexible and discontinuous multibody systems (up to 2004)
- integration with graphical interface in 2003 (with Yury Vetyukov)
- implementation of various structural finite elements, such as flexible beam and plate elements based on the absolute nodal coordinate formulation
- implementation of the floating frame of reference concept, as well as the component mode synthesis
- HOTINT made available to and further developed by Linz Center of Mechatronics (since 2007)
- HOTINT made available to and further developed by Austrian Center of Competence in Mechatronics (from 2008 to 2013)
- User version of HOTINT V1.1 available as freeware (2013)
- A open source version of HOTINT is available (end of 2013)

Current State of HOTINT

HOTINT mainly consists of the multibody kernel, the solver and linear algebra kernel, and the graphics and user interface, and currently comprises several hundred thousand lines of code. It has been particularly developed for the use of arbitrary classes of fully implicit Runge Kutta (IRK) methods. The IRK-tableaus can be defined in an external text-file and are given for several methods for 1 to 10 stages. The code makes advantage of the very high order reached through the use of fully implicit methods, which makes it especially then fast, when higher
accuracy is needed. In the current version, the $K$-form of IRK-equations has been implemented for the fast integration of 2nd order (mechanical) systems. Instead of trying to invert the mass matrix, which leads to large terms in the case of symbolic inversion, or instead of trying to add the system as a constraint equation (this has been done by some people who implemented their system into existing codes), you can now provide the mass matrix and the right hand side separately and the solver only solves one large system, but does not need the accelerations to be written explicitly as function of the remaining unknowns.

Summarizing, advanced methods from flexible multibody dynamics cover

• the efficient geometrical description for moving rigid bodies and bodies with superimposed small deformation,

• the application of special finite element methods, which are well suited for simulating large deformations of structural elements,

• high-order implicit time-integration schemes, in order to enforce stability for the numerical solution,

• a sophisticated treatment of algebraic equations for the arbitrary coupling of bodies, and for the incorporation of certain (boundary) conditions,

• and finally the reduction of the system size by a component mode synthesis (CMS).

General Information

Chief developer
Johannes Gerstmayr

Further developers
Larissa Aigner, Markus Dibold, Alexander Dorninger, Rafael Eder, Peter Gruber, Alexander Humer, Karin Nachbagauer, Astrid Pechstein, Daniel Reischl, Martin Saxinger, Markus Schörgenhumer, Michael Stangl, Yury Vetyukov, Simon Weitzhofer

Contact
support@hotint.org
Linz Center of Mechatronics GmbH
Altenbergerstr. 69, 4040 Linz, AUSTRIA
http://www.lcm.at

Thanks
The help and support from the contributors of the Institute of Technical Mechanics and Institute of Numerical Mathematics at the Johannes Kepler University of Linz is greatly appreciated.
I would like to acknowledge the important grant of the FWF ("Fond zur Förderung Wissenschaflicher Forschung" - the Austrian National Science Fund) within the project P15195-N03 and the APART project of the Austrian Academy of Sciences.

Parts of this software have been developed in the project "Nachhaltig ressourcenschonende elektrische Antriebe durch höchste Energie- und Material-Effizienz" (sustainable and resource saving electrical drives through high energy and material efficiency) which is part of the European Union program "Regionale Wettbewerbsfähigkeit ÖÖ 2007-2013 (Regio 13)" sponsored by the European Regional Development Fund (ERDF) and the Province of Upper Austria.

Parts of this software have been developed with the support of the Comet K2 Austrian Center of Competence in Mechatronics (ACCM).

Link

http://www.hotint.org
Copyright and license

HotInt General License (Version 1.0)

Copyright (c) 1997 - 2018 Johannes Gerstmayr, Linz Center of Mechatronics GmbH, Austrian Center of Competence in Mechatronics GmbH, Institute of Technical Mechanics at the Johannes Kepler Universitaet Linz, Austria. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This program contains SuperLU 5.0, ExtGL, BLAS 3.5.0, LAPACK 3.5.0, Spectra and Eigen covered under the following licenses:

SuperLU 5.0

Copyright (c) 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy)
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(3) Neither the name of Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ExtGl

Copyright (c) 2002, Lev Povalahev. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BLAS 3.5.0

The reference BLAS is a freely-available software package. It is available from netlib via anonymous ftp and the World Wide Web. Thus, it can be included in commercial software packages (and has been). We only ask that proper credit
be given to the authors.

Like all software, it is copyrighted. It is not trademarked, but we do ask the following:

If you modify the source for these routines we ask that you change the name of the routine and comment the changes made to the original. We will gladly answer any questions regarding the software. If a modification is done, however, it is the responsibility of the person who modified the routine to provide support.

LAPACK 3.5.0

Copyright (c) 1992–2013 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2000–2013 The University of California Berkeley. All rights reserved.

Copyright (c) 2006–2013 The University of Colorado Denver. All rights reserved.

$COPYRIGHT$

Additional copyrights may follow.

$HEADERS$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in this license in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following libraries are not linked to HOTINT directly but are requirements
of the software libraries BLAS and LAPACK.

LIBFORTRAN

Copyright (C) 2002–2013 Free Software Foundation, Inc.
Contributed by Paul Brook <paul@nowt.org>, and
Andy Vaught <andy@xena.eas.asu.edu>

Libgfortran is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3, or (at your option)
any later version.

Libgfortran is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

Under Section 7 of GPL version 3, you are granted additional
permissions described in the GCC Runtime Library Exception, version
3.1, as published by the Free Software Foundation.

You should have received a copy of the GNU General Public License and
a copy of the GCC Runtime Library Exception along with this program;
see the files COPYING3 and COPYING.RUNTIME respectively. If not, see
<http://www.gnu.org/licenses/>.

LIBGCC

Copyright (C) 2005–2014 Free Software Foundation, Inc.

GCC is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License as published by the Free
Software Foundation; either version 3, or (at your option) any later
version.

GCC is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
for more details.

Under Section 7 of GPL version 3, you are granted additional
permissions described in the GCC Runtime Library Exception, version
3.1, as published by the Free Software Foundation.

You should have received a copy of the GNU General Public License and
a copy of the GCC Runtime Library Exception along with this program;
see the files COPYING3 and COPYING.RUNTIME respectively. If not, see
<http://www.gnu.org/licenses/>.
CHAPTER 1. GENERAL INFORMATION

LibQuadmath

GCC Quad-Precision Math Library
Copyright (C) 2010, 2011 Free Software Foundation, Inc.
Written by François-Xavieroudert <fxoudert@gcc.gnu.org>
This file is part of the libquadmath library.
Libquadmath is free software: you can redistribute it and/or
modify it under the terms of the GNU Library General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.
Libquadmath is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Library General Public License for more details.
You should have received a copy of the GNU Library General Public
License along with libquadmath; see the file COPYING.LIB. If
not, write to the Free Software Foundation, Inc., 51 Franklin Street — Fifth
Floor,
Boston, MA 02110—1301, USA.

Libwinpthread

Copyright (c) 2011 mingw-w64 project

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

/*
 * Parts of this library are derived by:
 * Posix Threads library for Microsoft Windows
 * Use at own risk, there is no implied warranty to this code.
 * It uses undocumented features of Microsoft Windows that can change
 * at any time in the future.
 * (C) 2010 Lockless Inc.
 * All rights reserved.
 * Redistribution and use in source and binary forms, with or without
 * modification,
 * are permitted provided that the following conditions are met:
Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
Neither the name of Lockless Inc. nor the names of its contributors may be
used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AN
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

GNU LIBRARY GENERAL PUBLIC LICENSE Version 2

GNU LIBRARY GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is
numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software—to make sure the software is free for all its users.

This license, the Library General Public License, applies to some
specially designated Free Software Foundation software, and to any
other libraries whose authors decide to use it. You can use it for
your libraries, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the
users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification").

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independently of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices
stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no
charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a
table of data to be supplied by an application program that uses
the facility, other than as an argument passed when the facility
is invoked, then you must make a good faith effort to ensure that,
in the event an application does not supply such function or
table, the facility still operates, and performs whatever part of
its purpose remains meaningful.

(For example, a function in a library to compute square roots has
a purpose that is entirely well-defined independent of the
application. Therefore, Subsection 2d requires that any
application-supplied function or table used by this function must
be optional: if the application does not supply it, the square
root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library. To do
this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.
6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work
based on the Library, uncombined with any other library
facilities. This must be distributed under the terms of the
Sections above.

b) Give prominent notice with the combined library of the fact
that part of it is a work based on the Library, and explaining
where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License. Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License. However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Library or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions. You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all. For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).

To apply these terms, attach the following notices to the library. It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Library General Public License for more details.

You should have received a copy of the GNU Library General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301
USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

GNU GENERAL PUBLIC LICENSE Version 3

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.


The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.
CHAPTER 1. GENERAL INFORMATION

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.
No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.


You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.


You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place
additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.


You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".
A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowing reliance" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.
Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.


If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
CHAPTER 1. GENERAL INFORMATION

OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read

GCC Runtime Library Exception Version 3.1

GCC RUNTIME LIBRARY EXCEPTION

Version 3.1, 31 March 2009

Copyright (C) 2009 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

This GCC Runtime Library Exception ("Exception") is an additional
permission under section 7 of the GNU General Public License, version
3 ("GPLv3"). It applies to a given file (the "Runtime Library") that
bears a notice placed by the copyright holder of the file stating that
the file is governed by GPLv3 along with this Exception.

When you use GCC to compile a program, GCC may combine portions of
certain GCC header files and runtime libraries with the compiled
program. The purpose of this Exception is to allow compilation of
non-GPL (including proprietary) programs to use, in this way, the
header files and runtime libraries covered by this Exception.

0. Definitions.

A file is an "Independent Module" if it either requires the Runtime
Library for execution after a Compilation Process, or makes use of an
interface provided by the Runtime Library, but is not otherwise based
on the Runtime Library.

"GCC" means a version of the GNU Compiler Collection, with or without
modifications, governed by version 3 (or a specified later version) of
the GNU General Public License (GPL) with the option of using any
subsequent versions published by the FSF.
"GPL-compatible Software" is software whose conditions of propagation, modification and use would permit combination with GCC in accord with the license of GCC.

"Target Code" refers to output from any compiler for a real or virtual target processor architecture, in executable form or suitable for input to an assembler, loader, linker and/or execution phase. Notwithstanding that, Target Code does not include data in any format that is used as a compiler intermediate representation, or used for producing a compiler intermediate representation.

The "Compilation Process" transforms code entirely represented in non-intermediate languages designed for human-written code, and/or in Java Virtual Machine byte code, into Target Code. Thus, for example, use of source code generators and preprocessors need not be considered part of the Compilation Process, since the Compilation Process can be understood as starting with the output of the generators or preprocessors.

A Compilation Process is "Eligible" if it is done using GCC, alone or with other GPL-compatible software, or if it is done without using any work based on GCC. For example, using non-GPL-compatible Software to optimize any GCC intermediate representations would not qualify as an Eligible Compilation Process.

1. Grant of Additional Permission.

You have permission to propagate a work of Target Code formed by combining the Runtime Library with Independent Modules, even if such propagation would otherwise violate the terms of GPLv3, provided that all Target Code was generated by Eligible Compilation Processes. You may then convey such a combination under terms of your choice, consistent with the licensing of the Independent Modules.

2. No Weakening of GCC Copyleft.

The availability of this Exception does not imply any general presumption that third-party software is unaffected by the copyleft requirements of the license of GCC.

Spectra and Eigen

Both Spectra (http://yixuan.cos.name/spectra) and Eigen (http://eigen.tuxfamily.org) are covered by the Mozilla Public License Version 2.0 (https://www.mozilla.org/en-US/MPL/2.0/)

Mozilla Public License Version 2.0

1. Definitions

1.1. "Contributor" means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.
1.2. "Contributor Version"
means the combination of the Contributions of others (if any) used
by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution"
means Covered Software of a particular Contributor.

1.4. "Covered Software"
means Source Code Form to which the initial Contributor has attached
the notice in Exhibit A, the Executable Form of such Source Code
Form, and Modifications of such Source Code Form, in each case
including portions thereof.

1.5. "Incompatible With Secondary Licenses"
means
(a) that the initial Contributor has attached the notice described
in Exhibit B to the Covered Software; or
(b) that the Covered Software was made available under the terms of
version 1.1 or earlier of the License, but not also under the
terms of a Secondary License.

1.6. "Executable Form"
means any form of the work other than Source Code Form.

1.7. "Larger Work"
means a work that combines Covered Software with other material, in
a separate file or files, that is not Covered Software.

1.8. "License"
means this document.

1.9. "Licensable"
means having the right to grant, to the maximum extent possible,
whether at the time of the initial grant or subsequently, any and
all of the rights conveyed by this License.

1.10. "Modifications"
means any of the following:
(a) any file in Source Code Form that results from an addition to,
deletion from, or modification of the contents of Covered
Software; or
(b) any new file in Source Code Form that contains any Covered
Software.

1.11. "Patent Claims" of a Contributor
means any patent claim(s), including without limitation, method,
process, and apparatus claims, in any patent Licensable by such
Contributor that would be infringed, but for the grant of the
License, by the making, using, selling, offering for sale, having
made, import, or transfer of either its Contributions or its
Contributor Version.

1.12. "Secondary License"
means either the GNU General Public License, Version 2.0, the GNU
Lesser General Public License, Version 2.1, the GNU Affero General
Public License, Version 3.0, or any later versions of those licenses.

1.13. "Source Code Form"
means the form of the work preferred for making modifications.

1.14. "You" (or "Your")
means an individual or a legal entity exercising rights under this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants
Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date
The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope
The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

(a) for any code that a Contributor has removed from Covered Software; or

(b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.
This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of such Source Code Form. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice,
CHAPTER 1. GENERAL INFORMATION

provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor notifies You of the non-compliance by some reasonable means prior to 60 days after You have become back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after
Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation
Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look
for such a notice.

You may add additional accurate notices of copyright ownership.

**Exhibit B – "Incompatible With Secondary Licenses" Notice**

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.
Chapter 2

HOTINT User Manual
2.1 Multibody formulation

The present code is based on a redundant coordinate formulation for the modeling of the motion and deformation of bodies. This means that e.g. every rigid body has its own six degrees of freedom (DOF), no matter how this body is constrained by other bodies or even if it is fixed to the ground. The main reason for this formulation is the simple extensibility of the code regarding the development of new elements, constraints, forces, etc. The numerical efficiency is gained by adapted solvers for the sparse structure of the system equations, which leads to a similar effort as in recursive and minimal coordinate approaches.

Several main points have been focused in the multibody kernel:

- The application of implicit time integration algorithms shall be efficient
- The code shall be capable of structural and solid finite elements
- The code shall be extendable and open for new elements (e.g. non-mechanical, variable mass, variable topology, etc.)

Some things you should know:

Dimensions: dimensions are chosen by user, but should use standard international units: kg/-m/s.

Numbering: All lists, arrays or other ordering numbers start with 1 if not specified differently.

Elements: Bodies and connectors are elements. If you search for bodies or connectors (e.g. for editing) in the HOTINT program, you should look for elements.

2.1.1 Solution vector

The multibody system and solver always have two solution vectors. One containing either the initial vector or the actual solution (this is the solution vector) and another one that is used for the graphics drawing which is called drawing solution vector. The latter vector is utilized to independently draw the solution of a certain computed time instant during the computation (e.g. if the computation lasts very long or is of indefinite length).

The solution vector is split into a “position level” (not necessarily a real position) and “velocity level” part for the case of the second order differential variables. Assume that there are \( n \) second order differential equation variables, then the solution vector will contain first \( n \) position level coordinates and after that \( n \) velocity level coordinates. The local coordinates of a body (e.g. accessible via the sensor) are ordered in a similar way. The local second order differential variables of a body contain first \( m \) position level coordinates and after that another \( m \) velocity level coordinates.

2.1.2 Main structure of the multibody kernel

There were some main points to be fulfilled with the present multibody kernel:

- The formulation shall be easily accessible and maintainable via C++ functions
- The formulation shall be easily accessible and maintainable via the Windows user interface.
2.1. MULTIBODY FORMULATION

In the current implementation there is one base multibody system object which contains all information about the system. On top of this structure, there is a dynamic and static solver class, i.e. an implicit time integration method and a incremental nonlinear solver. The solver requires the multibody system to provide residuals and derivatives of the differential and algebraic equations based on assumed values.

The multibody system consists of the following components:

- Elements
- Nodes
- Loads
- Sensors
- Geometric elements

Every object of a multibody system, see Sec. 2.1.3 and Fig. 2.1, adds a certain set of their own (local) equations to the whole set of (global) equations. The crucial task of the Multibody System kernel is to assemble these global equations based on the connections of the multibody system, and to provide the system equations to the solver. Apart from that, the kernel is responsible for setting up the model, steering the simulation, organizing in- and output of file data, as well as accessing or modifying specific element data.
2.1.3 Object library

The object library provides a set of rigid bodies (links), basic joints, loads and sensors, similar to any other simulation code. As a main feature of HOTINT, there exists a variety of flexible bodies (Finite Elements), connectors (actuators, springs, and dampers), loads, sensors, and IO-Blocks (controllers) – as outlined in Fig. 2.1.

Among flexible bodies are structural Finite Elements for beams, available either in geometrically exact formulation (for large deformation processes, ropes, cables, etc.), or in linearized form (faster). Joints are designed such that complex combinations of bodies and joints are possible, e.g. a point of body may move along a deformable body’s axis (sliding joint). Since also flexible bodies are available, the object definition is based on generalized (redundant) coordinates for the bodies.

In the core part of the code, objects are represented by means of first order and second order differential equations, algebraic equations and jump or switching conditions. Furthermore, the objects include standardized coupling conditions (for joints and loads), graphical representation and measurable quantities (for sensors). These objects define bodies (links) and joints and may be easily extended.

2.1.4 The dynamic solver – implicit time integration

The numerical time-integration tool included in HOTINT is designed to compute the numerical solution of mixed first and second order differential equations (ODE) and differential-algebraic equations (DAEs) up to an index of 3. The numerical solution is obtained by using implicit Runge-Kutta (IRK) schemes like Gauss, Radau and Lobatto formulas. The code is developed for an arbitrary number of stages, so far 20 stages have been tested resulting in the conclusion that computing with as much as 10 stages can improve the speed of the
numerical simulation before the machine precision is limiting the convergence of the underlying Newton method.

Different IRK schemes are defined by tableaus of coefficients which are defined by means of ASCII-Files (file “tableaus.txt”). These files are automatically generated by means of built in functions of the code Mathematica 5.0. While it is known that multi-step solvers can integrate DAEs of index 2 (e.g. BDF), it has been found out that the inability to restart the multi-step method quickly after a discontinuous step makes it unattractive for discontinuous problems. Furthermore, the order of multi-step methods is limited by a comparatively low upper bound, while it is possible to show that an order of 20 for the integration is possible and can even be most efficient.

It shall be mentioned that in the special case, where a high accuracy of the solution of the DAE is needed, e.g. for sensitivity analysis or optimization methods, the very high order of IRK methods is very advantageous.

For the present case of the freeware HOTINT code, only low order IRK formulas are available, while the higher order methods will be available in future versions.

For a description of the methods see the paper (download via homepage of HOTINT):


2.1.4.1 Index 2 Formulation

In the present implementation, only the index 2 formulation can be chosen. The index 2 multibody formalism transforms all constraints to the velocity level. This leads to a highly stable and efficient formalism (the velocity level can be solved much easier than the position level).

The time integration algorithm forces the constraint conditions at the velocity level in each time step (at the integration points of each time step). The integration over the velocity does not exactly give the fulfillment of the position level constraint, thus a small drift occurs. The drift becomes considerably smaller with smaller step size and can be usually ignored.

**Recommendations:** Do not select too large time steps. If you have fast rotating bodies, it is important to guarantee sufficient time steps during each rotation of the bodies. It is usually sufficient to use between 20 and 100 steps per one rotation in order to get sufficient accuracy and small drift.

**Future implementations:** Stabilization techniques are already included in HOTINT, but they need to be built into the general framework. The stabilization as well as the error control of the drift will be available in future versions of HOTINT.
2.1.5 The static solver – incremental loading

The nonlinear solver sets all velocities and acceleration terms to zero. The solver tries to find a static solution (if possible) starting with the initial configuration. All loads are increased linearly between the virtual time 0 and 1, in order to achieve convergence for very nonlinear problems. The (virtual) time step (=load increment) can be theoretically set to 1, but then the load is applied in one step and the nonlinear problem needs to be solved at once. The static solver tries to decrease the load increment as far as necessary in order to achieve convergence, however, it is advantageous to specify a certain load increment which can help the solver to speed up the computation and avoid failed steps.

The static solver does not work for kinematical systems (statically underdetermined systems). Small rotational or translational springs can be added in order to transform the system to a statically determined system.

2.1.6 Eigenmode computation

There are different methods used in order to compute eigenmodes of the multibody system. The different methods are described below. The eigensolver in HOTINT does not work yet for general Lagrange-multiplier constraints, although it is known how to compute eigenmodes for problems with Lagrange multipliers, [11]. Presently, all penalty-based constraints can be used and constraints can be applied on single coordinates, e.g. in order to obtain clamped constraint conditions.

How to compute the eigenvalues and eigenmodes for a still standing multibody system:

Equations of motion: \( M(x) \ddot{x} + K(x) x = 0 \)

Computed are the eigenvalues/modes of the first order system \( A \):
2.1. MULTIBODY FORMULATION

\[
\ddot{x} = A x = \ddot{x} = [-M^{-1}K] x
\]
\[K v = \lambda M v\]  

M ... mass matrix of the multibody system
K ... stiffness matrix of the multibody system
v ... eigenvectors of matrix A
\(\lambda\) ... eigenvalues of matrix A

1.) Open the menu Edit Solver Options

2.) Set general options, they are independent from selected solver
2 a) Eigensolver.do_eigenmode_computation ... if checked \(\rightarrow\) eigenvalue computation on button START.
2 b) Eigensolver.linearize_about_actual_solution ... use actual solution as configuration for linearization of \(K/M\). Eigenvalues are computed for linearization around stored solution vector of last static/dynamic solution! All velocities are set to zero.
2 c) Eigensolver.use_gyroscopic_terms ... make sure that box is not checked
2 d) Eigensolver.eigenmodes_scaling_factor ... scaling factor for eigenmodes, eigenvectors are multiplied with this factor
2 e) Eigensolver.eigenmodes_normalization_mode ... \(0 \rightarrow\) standard mode, \(max(v) = 1; 1 \rightarrow v^Tv = 1\)
2 f) Eigensolver.use_n_zero_modes ... flag is not used in current version
2 g) Eigensolver.reuse_last_eigenvectors ... flag is not used in current version

3.) Set the subtree Eigensolver.solver_type ... define the solver type
0 ... LAPACK dsygv direct solver, LAPACK package used
The solver will calculate all possible eigenvalues/eigenmodes of the multibody system. Solver options are not offered. For information about the accuracy see the LAPACK documentation.

1 ... Arnoldi iterative solver (Matlab), Matlab licence is needed
Eigensolver.max_iterations ... maximum number of iterations for iterative eigenvalue solver
Eigensolver.accuracy ... tolerance for iterative eigenvalue solver
Eigensolver.n_eigvals ... number of eigenvalues and eigenmodes to be computed for sparse iterative methods
Eigensolver.n_zero_modes ... number of zero eigenvalues (convergence check)

2 ... LOBPCG iterative solver, implemented in HOTINT
Eigensolver.max_iterations ... maximum number of iterations for iterative eigenvalue solver
Eigensolver.accuracy ... tolerance for iterative eigenvalue solver
Eigensolver.use_preconditioning ... if checked \(\rightarrow\) set a value for \(\lambda\) in Eigensolver.preconditioner_lambda, \(\text{inv}(K + \lambda M)\)
Eigensolver.n_eigvals ... number of eigenvalues and eigenmodes to be computed for sparse iterative methods
Eigensolver.n_zero_modes ... number of zero eigenvalues (convergence check)

How to compute the eigenvalues and eigenmodes for a multibody system with gyroscopic terms:
Equations of motion: \( M(x) \ddot{x} + G(x, \dot{x}) \dot{x} + K(x) x = 0 \)

Computed are the eigenvalues/modes of the first order system \( A \):

\[
A = \begin{bmatrix}
0 & E \\
-M^{-1}K & -M^{-1}G
\end{bmatrix}
\]

\[
\begin{bmatrix}
\dot{x} \\
\ddot{x}
\end{bmatrix} = A \begin{bmatrix}
x \\
\dot{x}
\end{bmatrix}
\]

\[
A \nu = \lambda \nu
\]

\( M \) ... mass matrix of the multibody system  
\( K \) ... stiffness matrix of the multibody system  
\( G \) ... gyroscopy matrix of the multibody system  
\( \nu \) ... eigenvectors of matrix \( A \)  
\( \lambda \) ... eigenvalues of matrix \( A \)

1.) Open the menu Edit Solver Options

2.) Set general options, they are independent from selected solver  
2 a) Eigensolver.do_eigenmode_computation ... if checked \( \rightarrow \) eigenvalue computation on button START.  
2 b) Eigensolver.linearize_about_actual_solution ... use actual solution as configuration for linearization of \( K/M \). Eigenvalues are computed for linearization around stored solution vector of last static/dynamic solution!  
2 c) Eigensolver.eigenmodes_scaling_factor ... scaling factor for eigenmodes, eigenvectors are multiplied with this factor  
2 d) Eigensolver.eigenmodes_normalization_mode ... 0 \( \rightarrow \) standard mode, \( max(\bar{v}) = 1; 1 \rightarrow \bar{v}^T\bar{v} = 1 \); Attention: For a proper drawing representation the vector \( \bar{v} \) (used for normalization) contains only the the positions \( (x = [\bar{v}, \dot{\bar{v}}]) \)

3.) Check Eigensolver.use_gyroscopic_terms ... use gyroscopy terms for eigenvalue computation  
The eigenvalues/modes of the nonsymmetric matrix \( A \) are computed with the LAPACK dgeev solver. Other options for this solver are not necessary. Relating to the accuracy see the LAPACK documentation.

The computation of the eigenvalues/eigenmodes requires a inversion of the full mass matrix of the dynamic system. This could be a problem for very large systems.

**How to create a campbell diagram, e.g. for rotordynamics:**

It is very simple to create a campbell diagram with HOTINT. To create a campbell diagramm do the following steps:

1.) Set up a rotor model, e.g. by adding RotorBeamXAxis and NodalDiskMass3D elements and Node3DR123 nodes to the multibody system
2.1. MULTIBODY FORMULATION

1. a) Initialize all nodes Node3DR123 in dependence of the variable you want to vary, e.g. in the case of the Campbell diagram the rotor speed omega:

\[ \text{Initialization.node\_initial\_values} = [0, 0, 0, 0, 0, 0, 0, 0, \omega, 0, 0] \]

2.) Set the eigensolver, see "How to compute the eigenvalues and eigenmodes for a multibody system in motion"

3.) Set a parameter variation for omega (range and step size)

4.) Perform computation, the eigenvalues and varied parameter are stored in the solution file, e.g. solpar.txt in the output folder

5.) Open the plot tool and load output file:

5 a) Click External file and select the output file, e.g. solpar.txt

5 b) Select \text{n\_rot} and a eigenvalue of your choice, e.g. eigval1 and create a x/y plot

6.) For a Campbell diagram it is necessary to add a line with the frequency of the rotor speed. Create a txt file with the following lines:

Example

\begin{verbatim}
%Comment: y=x/60 (x in 1/min, y in Hz)
%\%1 2
%\%n\_rot frequency
0 0
x y
\end{verbatim}

Replace the x with the max. rotor speed and y with calculated frequency value, load the file and create a x/y plot.

7.) Label the plot

In the following is an excerpt of such a rotor example (the full example is included in examples/campbell):

Example

\begin{verbatim}
% Rotor Beam Example -> Campbell diagram
% \% parameters
% ...

n\_rot = 1000 \% rpm, vary this parameter
omega = 2*Pi*n\_rot/60 \% rad/s

%======================================================================
% rotordynamics model
%======================================================================

% add materials (the material does not depend on omega)
%

% add node 1 with initial velocity
n
{
\begin{verbatim}
node_type = "Node3DR123"
name = "node_1"
Geometry.reference_position = [0,0,0]
Initialization.node_initial_values =
[0, 0, 0, 0, 0, 0, 0, 0, 0, omega, 0, 0]
% initial values for all DOF of node: 1...6 => pos, 7...12 => vel
}
n1 = AddNode(n)
% ...similar for all other nodes

% add rotor beams
% ...
% add nodal disk masses
% ...
% add bearings
% ...

%======================================================================
% set parameter variation
%======================================================================
solveroptions.parametervariation.activate = 1 % do parameter variation
solveroptions.parametervariation.start_value = 0 % rpm
solveroptions.parametervariation.end_value = 80000 % rpm
solveroptions.parametervariation.arithmetic_step = 1000 % rpm
solveroptions.parametervariation.mbs_edc_variable_name = "n_rot" % name

%======================================================================
% set eigensolver
%======================================================================
SolverOptions.Eigensolver.use_gyroscopic_terms = 1 % use gyro terms
SolverOptions.Eigensolver.do_eigenmode_computation = 1 % must be set to 1
\end{verbatim}

Figure: Two disk rotor created with file campbell.txt
2.1.7 Parameter Variation, Sensitivity Analysis, Identification and Optimization

In order to study the global influence of certain parameters to the simulation results, a parameter variation can be performed, which e.g. gives a set of results with respect to one or two varied parameters. In order to investigate the local influence of specific parameters on the solution, a sensitivity analysis can be performed, which results in a matrix which shows the dependence of a cost function with respect to the change of parameters. Based on the functionality of the parameter variation, it is possible to perform optimization and parameter identification in HOTINT. The implemented genetic parameter identification algorithm, documented in [12], is used to search the best fitting model parameters in a systematic way. The cost function for the identification/optimization can be based on the difference of a reference solution, e.g. from measurements and simulated results. The algorithm searches the optimal parameters in a given parameter space (e.g. parameter ranges). Multiple minima of the cost function may occur and are no problem for the genetic algorithm. In contrast to Newton’s method, derivatives of the cost function with respect to the parameters are not required. In a first step, for each set of randomly chosen parameters, a simulation is performed and the cost function is evaluated. A specified number of best parameters is taken into account for the next generation of parameters, the surviving parameters. Based on these parameters a new set of parameters (children) are generated using the principle of mutation and the parameter search range is reduced. This procedure is repeated until the optimum is nearly reached.

Parameter Variation: In the menu Edit Solver Options under the subtree SolverOptions.ParameterVariation the parameter variation can be set.

For the start of the parameter optimization, following things have to be done:

1.) Set up your HOTINT model which contains a parameter for variation.

Figure: Campbell diagram created with the file examples/campell. The eigenfrequencies are the first bending modes.
2.) `SolverOptions.ParameterVariation.MBS_EDC_variable_name` ... define EDC-name of the parameter (e.g. "i")

3.) Define the range of the parameter value. The variation is repeated as long as the \( p_i \leq p_n \).
   3 a) `SolverOptions.ParameterVariation.start_value` ... start value of the parameter \( p_0 \)
   3 b) `SolverOptions.ParameterVariation.end_value` ... end value of the parameter \( p_n \)

4.) Define arithmetic or geometric step method ... arithmetic: \( p_i = p_{i-1} + \Delta p \); geometric: \( p_i = p_{i-1}f \); \( p_i \) ... parameter value at step \( i \);
   4 a) `SolverOptions.ParameterVariation.geometric` ... check for geometric step, else arithmetic step
   4 b) `SolverOptions.ParameterVariation.arithmetic_step` ... set \( \Delta p \)
   4 c) `SolverOptions.ParameterVariation.arithmetic_step` ... set \( f \)

5.) Activate variation algorithm
   5 a) Check field `SolverOptions.ParameterVariation.activate`

In the following there is a simple example code of a parameter variation. A parameter \( i \) is varied from 1 to 5 with a step size of 1 and displayed in the output window.

**Example**

```plaintext
% Test for printing in combination with parameter variation
HOTINT_data_file_version="1.1.498"
i=1

Print("The number is ")
Print(i)
Print(" \n")

SolverOptions.ParameterVariation.activate = 1
SolverOptions.ParameterVariation.start_value = 1
SolverOptions.ParameterVariation.end_value = 5
SolverOptions.ParameterVariation.arithmetic_step = 1
SolverOptions.ParameterVariation.MBS_EDC_variable_name = "i"
```

**Genetic Optimization:** Generally, the subtree `SolverOptions.Optimization` in the menu `Edit Solver Options` contains methods for optimization or in other words minimization of certain computation values (or cost function) from sensor signals in form of a search of the best-matching parameters. See the excerpt of the hid file of a two mass oscillator (examples/two_mass_oscillator), which shows the optimization of an unknown spring stiffness. The optimization is based on the difference to a reference two mass oscillator example.

For the start of the parameter optimization, following things have to be done:

1.) Set up your HOTINT model with at least one sensor (e.g. Two-Mass-Oscillator)

2.) Define computation value(s) from sensor signal(s) with `SolverOptions.Optimization.sensors`. They are minimized by the optimization; if more than one sensor computation value is defined, the sum of the computational value will be minimized
2.1. MULTIBODY FORMULATION

3.) Define optimized parameters and their limits
3 a) SolverOptions.Optimization.number_of_params ... number of parameters, which are optimized (e.g. 1)
3 b) SolverOptions.Optimization.param_name1 ... define EDC-name of optimized parameter (e.g. "k1_var")
3 c) SolverOptions.Optimization.[min|max]val1 ... define limits for the parameter search (e.g. SolverOptions.Optimization.minval = 0 and SolverOptions.Optimization.maxval = 1)
3 d) Repeat a)-c) until all parameters and limits are defined

4.) Check the Genetic Optimization Options SolverOptions.Optimization.Genetic

This option should only be modified, if the computation time or accuracy of the optimization process should be changed. For more accurate results increase the SolverOptions.Optimization.Genetic.initial_population_size, SolverOptions.Optimization.Genetic.surviving_population_size, or try to change the other options in SolverOptions.Optimization.Genetic. This is a very critical point, because the accuracy but also the computation time is increased. Further descriptions and more detailed insight to the influence of the genetic optimization parameters can be found in previous work (R. Ludwig and J. Gerstmayr, AUTOMATIC PARAMETER IDENTIFICATION FOR GENERIC ROBOT MODELS, MULTIBODY DYNAMICS 2011, ECCOMAS Thematic Conference, J.C. Samin, P. Fisette (eds.), Brussels, Belgium, 4-7 July 2011).

5.) Activate optimization algorithm
5 a) Check field SolverOptions.Optimization.activate
5 a1) Optional: check field SolverOptions.Optimization.run_with_nominal_parameters (for checking the consistency of the model and the nominal sensor computation value)
5 a2) Optional: check field SolverOptions.Optimization.restart (if genetic optimization should be restarted with already known parameters from previous genetic optimizations). This option saves computation time if results from previous optimization(s) should be used.
5 b) Set option SolverOptions.Optimization.method = "Genetic". Further algorithms are planned.

6.) Press OK-Button in Edit Solver Options, then Start! in the main window
The optimization repeats the simulation with different parameter sets and writes usually further informations about the optimization process into the Computation Output - Window. Furthermore, a result file is written into the path GeneralOptions.Paths.sensor_output_path with the filename defined in the option SolverOptions.Solution.ParameterFile.parameter_variation_filename (e.g. solpar.txt). This file is needed for the SolverOptions.Optimization.restart option, see 5 a2).

7.) Check result
7 a) Use optimized parameters as nominal parameters, simulate once (e.g. with 5 a1)
7 b1) If results are accurate enough → optimization process finished.
7 b2) otherwise repeat points 3.)-7.)

Important notes:
For a higher speed of the optimization it is useful to close the GUI Data Manager as well as the Computation Output. If you want to see some information about the optimization progress during the computation open the static output window instead (Results → Show Static Output). It is also recommended to uncheck SolverOptions.Solution.write_solution in Edit Solver Options. Otherwise the sensor data of every optimization step is written to the output file.
Figure: Two mass oscillator with stiffness and damping optimization.

An excerpt of the full example (included in examples/two_mass_oscillator) is written below:

Example

% parameters:
% ...
% k1= 500 % N/m, stiffness spring 1, nominal value
% k1_var= 350 % N/m, stiffness spring 1, arbitrary value, not used
% for optimization, this value is used if
% run_with_nominal_parameters= 1
% d2= 30 % N/(m/s), damping spring 2, nominal value
% d2_var= 10 % N/(m/s), damping spring 2, arbitrary value

% read vectors from file (measurement data) and create math functions

\%t = LoadVectorFromFile("...path...",1)
\%disp_m1 = LoadVectorFromFile("...path...",2)
\%vel_m1 = LoadVectorFromFile("...path...",3)
\%disp_m2 = LoadVectorFromFile("...path...",4)
\%vel_m2 = LoadVectorFromFile("...path...",5)

mathFunction
{
    MathFunction
    {
        piecewise_mode= 1
        piecewise_points= t
        piecewise_values= disp_m1
    }
}

nSensDisp1Ref = AddElement(mathFunction)

% ... similar for other math functions ...
% 2.1. MULTIBODY FORMULATION
%======================================================================
% model with varied parameters
%======================================================================

% ... add masses, spring dampers, sensors with varied parameters (in this
% case only the stiffness of spring damper 1 differs to nom. parameters)

spring_damper1
{
    % ...
    Physics.Linear.spring_stiffness= k1_var
}
nSpringDamper1Var= AddConnector(spring_damper1)

spring_damper2
{
    % ...
    Physics.Linear.damping= d2_var
}
nSpringDamper2Var= AddConnector(spring_damper2)

%======================================================================
% optimization
%======================================================================

nSensor= AddSensor(...) % this sensor measures the cost function (in
% this example it is the average of sum of the quadratic errors of the
% displacements and velocities of the masses)

SolverOptions
{
    Optimization
    {
        activate= 1 % set this flag for optimization
        run_with_nominal_parameters= 0 % 1..perform single simulation
        restart= 0 %0..create new parameter file
        method= "Genetic" % genetic: optimize using random parameters,
        % best parameters are further tracked.
        sensors= nSensor % sensor which measures the cost function
    }
    Genetic
    {
        initial_population_size= 20 % size of initial trial values.
        surviving_population_size= 15 % values which are further tracked
        number_of_children= 15 % number of children of surviving population
        number_of_generations= 4 % number of generations in genetic optimization
        range_reduction_factor= 0.5 % reduction of range of possible mutations
        randomizer_initialization= 0 % initialization of random function
        min_allowed_distance_factor= 0 % set to value greater than zero
    }
    Parameters
    {

number_of_params= 2 %Number of parameters to optimize.
param_name1= "k1_var" %Parameter name.
param_minval1= 3e2 %Lower limit of parameter.
param_maxval1= 7e2 %Upper limit of parameter.
param_name2= "d2_var" %Parameter name.
param_minval2= 10 %Lower limit of parameter.
param_maxval2= 70 %Upper limit of parameter.
}
}

Results of the optimization taken from solpar.txt file:

\begin{align*}
k_1 &= 498.642 \text{ N/m} \\
d_2 &= 30.0115 \text{ N/(m/s)} \\
cost \text{ function} &= 3.21674 \times 10^{-5}; \text{ The cost function in this example is the sum of squares of deviations between positions and velocities (see example file for more detail).}
\end{align*}

For a visualization of the optimization results open Results → PlotToolDialog. Select External File as Data Sources and choose the optimization file (e.g. solpar.txt as default filename) in the output folder. Click k1_var and Ctrl + cost_function_value and select Add x/y. Change the Point Style to X and the Line Style to invisible. Add a Title label X-Axis and Y-Axis. Save the picture and repeat the procedure for d2_var. You should get the figures below.

![Cost function diagram k1](image-url)

\textbf{Figure:} Cost function in dependence of spring stiffness k1.
Figure: Cost function in dependence of spring damping $d_2$.

2.1.8 The Element Concept

Elements: Bodies and connectors are elements. In fact, an element only needs to provide a set of differential and algebraic equations and it can add forces to other elements. A rigid body modeled with Euler parameters includes one constraint for the four Euler parameters and a hydraulic actor includes a differential equation for the pressure build-up equations. Therefore, bodies and connectors are treated within the same framework. Even forces or sensors could be elements, however, there would be too much overhead in treating just everything as an element.
2.1.9 Nodes for Direct Connection of Finite Elements

Sometimes it is more efficient to connect two elements without the application of constraints. E.g. in the case of nodal finite elements it is advantageous if the connected elements share nodal coordinates. Therefore, it is possible in HOTINT to define nodes, which can be afterwards used to assign nodal coordinates to elements.

A node is defined only for a certain number of coordinates (degrees of freedom – DOF), e.g. for a 2D position node DOF = 2, for a 3D position node DOF = 3, for a node using position and gradient, the DOF = 12 per node. Additionally, the nodal position in the reference configuration can be assigned to the node. This position can be later on used to find nodes or to automatically determine the nodal number depending on the nodal coordinate.

The nodes are consecutively ordered starting with the nodal number 1. The elements can afterwards refer to this number. When editing nodes, the available nodal numbers are shown.

2.1.10 The Concept of Loads

Loads are used to add forces at the right hand side of the second order differential equations that describe the dynamics of a body. Loads are directly linked to bodies and they do not have own generalized coordinates (unknowns). However, loads can depend on the body coordinates or body deformation (e.g. in the case of pressure).

The loads can have a time-dependency which is evaluated in every step of the computation. Loads can only be applied to bodies that provide according information of the work of external linear, angular or integrated loads.
2.1. MULTIBODY FORMULATION

2.1.11 Sensors for Measuring

Sensors are used to measure certain quantities of the multibody system at the current state of the computation. The output of a sensor is usually written to output files at certain time steps (See Computation Settings dialog). The solution file “sol.txt” contains the output of all sensors, each sensor in a row, versus the time (first row). Apart from output and controllers, sensors do not influence the computation.

While local DOF sensors can be used to measure the coordinates of any element (e.g. of a constraint), the position, angle, distance and deflection sensors can only be applied to elements of the type body.

Note that local second order differential variables of a body contain first \([1 \ldots m]\) position level coordinates and another \([m+1 \ldots 2m]\) velocity level coordinates.

Sensors cannot have own generalized coordinates (unknowns).

2.1.11.1 angles and angular velocities

The orientation of an element can be measured with CARDAN angles (or also called BRYANT angles). The according field variable (see section 3.9.1) for the sensor is bryant_angles.

The transformation of a vector from the body-fixed coordinate system into global coordinates is as follows (see [20]):

1. rotation around global z axis, \(\Phi_x\)
2. rotation around local y axis, \(\Phi_y\)
3. rotation around local x axis, \(\Phi_z\)

It is possible to measure the angular velocity in local or global coordinate system.

Note that the time derivative of the rotation, \((\dot{\Phi}_x, \dot{\Phi}_y, \dot{\Phi}_z)\), is not equal to the angular velocity \((\omega_x, \omega_y, \omega_z)\), neither in global coordinates nor in local coordinates.

2.1.12 Geometric Elements for Bodies with Complex Geometry

Geometric elements are used to represent a realistic shape of complex bodies in the multibody simulation. Usually, a geometric element is either used to define objects in the background or it is attached to a (rigid) body.

Geometric elements can be either defined with geometric primitives or by triangular meshes (see the Section about GeomMesh). The only influence to the computation by GeomElements is present by the automatic computation of mass, volume and inertia from the GeomElements. Usually, the complexity of GeomElements does not influence the computational time (CPU time), except for the drawing and loading/saving of multibody models. In the case of big GeomMesh models, it is recommended that the redrawing time is set to a high value, e.g. set the redrawing to every 20 seconds.
2.2 Getting started

2.2.1 Instructions for installing HOTINT on a MS-Windows computer

To begin with, you need to download the HOTINT zip-archive, and extract it to a folder of your choice using a program such as Winzip or 7-Zip. Then run the executable “setup.exe”, and follow the setup instructions as shown below:

First, you will see the start screen of the HOTINT setup wizard:

![Welcome to the HOTINT Setup Wizard]

Click “Next” to proceed. Now you can choose the installation folder, and specify whether to install HOTINT for all users on your computer, or just for you.

ATTENTION: Do not use a folder which is locked by windows for admin use only, e.g. the default program folder. It is recommended to use a folder within the “documents” or “user” folders.
2.2. GETTING STARTED

Click "Next", read the license agreement, check "I Agree", and click "Next" in order to proceed; click "Cancel" otherwise.

HOTINT is now ready to be installed on your computer; click "Next" to start the installation.
After the installation process, the following screen appears:

Click “Close” to exit the setup wizard.

The installation of HOTINT now is complete, and your chosen installation directory should contain a number of “.dll”-files, as well as the following folders:

**documentation**  
Contains the HOTINT user documentation, an “.rtf” license text file, and an “example” folder with ready-to-use “.hid” example model
files.

**HotIntWin32**
Contains the subfolder release, where the HOTINT executable “hotint.exe” and the configuration file “hotint.cfg.txt” are located.

**output**
This is, by default, the output directory where the solution file “sol.txt” containing sensor data is stored. Furthermore, the solution data files are created here in a subdirectory “solution_data”, if the flag store_data_to_files is checked in the solver options under “Solution” (see §2.5.8 or section 3.16 for details).

**userdata**
This folder can be used for your user-defined model files (cf. §2.4).

HOTINT is started by running the executable “hotint.exe” in the “HotIntWin32\release” folder. For convenience, it is recommended to create a shortcut (e.g., on your desktop or in the start menu directory) referencing that executable. The following section guides you through your first steps in HOTINT.

### 2.2.2 First steps
Start HOTINT by double-clicking “hotint.exe” located in the subfolder “HotIntWin32\release” in your installation directory, or a corresponding shortcut. The program starts with an empty multibody model.

The best way to experience the capabilities of HOTINT is to load one of the examples included in the subfolder “examples” and start to experiment. Select “OpenMBS” in the “File”-menu, and navigate to the examples located in “documentation\examples”: 
Here, we choose the file “double_pendulum.txt”...

2.2.3 Command Line Usage

HOTINT can also be configured and started via the command line (or from MATLAB). The syntax is

```
hotint [option1=value [option2=value [...]]]
```

Some remarks:
- When starting hotint.exe from DOS/Matlab, the current directory MUST be the root directory of hotint.exe.
- To start the Windows command prompt, run the executable cmd.exe (under Windows 7, just type “cmd.exe” in the search bar in the start menu and hit enter; alternatively, or in other Windows versions, use the “Run” command). Any settings of HOTINT options via the command line are accounted for after the HOTINT model file – possibly containing specifications for some options too – has been read in.
- Single option specifications must not include spaces; mutually, they are separated by spaces.
- Use \" instead of ".
- In order to run several instances of HOTINT in parallel, append the &-character at the end of each line which calls hotint.exe.

In order to open a model with HOTINT from command line use the options

```
GeneralOptions.ModelFile.hotint_input_data_filename
```

if you want to open a script model (hid-file) or
2.2. GETTING STARTED

GeneralOptions.ModelFile.internal_model_function_name

if you want to run a C++ model (models compiled with HOTINT).
A few examples for starting HOTINT via the command line:

```
hotint.exe GeneralOptions.ModelFile.hotint_input_data_filename="D:\models\hotint_file.hid"
    GeneralOptions.Application.start_computation_automatically=1
hotint.exe SolverOptions.Solution.output_filename="dir/myfile.txt"
hotint.exe SolverOptions.Timeint.tableau_name="RadauIIA"
```

One example using MATLAB:

```
dos('hotint.exe SolverOptions.Solution.output_filename="matlabfile1.txt" &')
```

In C++ you can have user-defined options. You can set these options via the command line too:

```
hotint.exe MyOptions.usemodeNr=2
```

### 2.2.4 Configure Notepad++ for HOTINT

As described in 2.8.1 it is possible to set up systems with text-files. These files can be written and changed in any editor, e.g. notepad++. Some editors provide the functionality of syntax highlighting and an auto-complete function for user-defined languages.

For this purpose 2 specific files are stored on your computer during the installation process in the folder documentation:

- HOTINT.xml
- hotint_highlight_notepad.xml

In the following it is described how to set up notepad++, such that these functionalities can be used. If you are using a different editor, the steps may be very similar.

1. save file 'HOTINT.xml' to the notepad folder 'plugins\APIs'
   (e.g. C:\Program Files (x86)\Notepad++ \plugins\APIs)
2. open NOTEPAD++
3. click on icon
   ![Notepad++ Icon](image)
4. import file 'hotint_highlight_notepad.xml'
If you open a file with the extensions 'txt' or 'hid' with notepad++ there should be 2 new features now:

- highlighting of known keywords
- auto complete (ctrl + space) for known keywords
2.3. HOTINT WINDOWS USER INTERFACE

2.3 HOTINT Windows User Interface

2.3.1 Using the graphics window

The 3D graphics window is used to visualize the multibody model by user-defined representation of the bodies, joints and forces. The graphical representation might be a simplification of the parameters used to perform the dynamical simulation.

2.3.2 Mouse control

**Rotation:** Press the right mouse button and move up/downwards and left/right to rotate the model.

**Zooming:** Use the scroll wheel to zoom in / out or press the right mouse button and “Shift” and move up/downwards.

**Zoom selection:** Use “Shift” and the left mouse button and select a rectangle to be zoomed into.

**Moving:** Press the left mouse button to move the model on the screen.

**Perspective:** Press the right mouse button, “Shift” and “Ctrl” and move up/downwards to change the distance of the camera to the object in order to change its perspective (the closer you zoom, the more distorted it gets).

2.3.3 HOTINT main application window
The main HOTINT window is used to load, save and edit models, start the computation, or modify computation parameters and viewing settings. After a computation the results can plotted as well as animated.

![Computation Output window](image)

The “Computation Output” window is used to print important messages, show computation results, the computation state, computation background information (e.g. number of Newton iterations), error and warning messages.

### 2.3.4 Specific buttons

The following buttons are available in the main view in HOTINT:
2.3. HOTINT WINDOWS USER INTERFACE

Start computation of multibody system
Pause computation of multibody system
Stop computation of multibody system
Restart computation of multibody system
Save HOTINT options (i.e. the configuration except for the solver settings)
Reload the selected (internal) model or the open script file
Enable/Disable rotation of model – for planar examples
Zoom whole model
Show x-y plane
Show x-z plane
Show y-z plane
Show user-defined view (see viewing options)
Choose / hide axes position
Automatic rotation
Save single image, directory is specified in record frames dialog
Open the record frames dialog in order to capture a series of images for an animation
Click to change viewing options to stored ones
Ctrl + Click to store current viewing options
3 independent settings are possible
Filenames can be set in GeneralOptions.SavedViewingOptions

2.3.5 HOTINT Main Menu

Outlining, the HOTINT main menu comprises the following entries which are described in more detail below:

- File
- View
- Edit
- Add Object
- System
- Computation
- Results
- ?
2.3.5.1 File

Select Model
Select a multibody model

New MBS
Create a new multibody model

Open MBS
Open an existing MBS file

Save MBS (as)
Save the current multibody model (as...)

Exit
Exit program

Recent:
List of recent files

2.3.5.2 View

Edit Hotint Options
Open the Hotint options dialog: Access all options concerning the MBS and the program, except for the solver options. See subsection 2.5.2 or section 3.16 in the reference manual for details.

Save Hotint Options
Saves the current configuration of Hotint options

Show Data Manager
Open dialog for viewing and animating the results of the computation; see subsection 2.5.7 for further information.

Show Output Window
Show the output window which reports important information, current state of the simulation, errors, etc. during the computation and modeling

Viewing Options
Open the viewing options dialog: configure redrawing, animation settings, grid (raster), standard view; see subsection 2.5.3 or section 3.16 in the reference manual for details.

OpenGL Options
Set the options for OpenGL 3D graphics: define lights positions and intensities, transparency, shading model and lighting; see subsection 2.5.4 or section 3.16 in the reference manual for details.

FE Drawing options
Dialog mainly to change settings for finite elements: Contour-iso plots, color/grey mode, shrinking factor, stress-type, tiling, resolution, line thickness; see subsection 2.5.5 or section 3.16 in the reference manual for details.

Body / Joint Options
Used to configure the user-input and drawing of bodies and joints: Rotation input mode, show body number, body frame, body transparency
Show joints, joint transparent and joint number; see subsection 2.5.6 or section 3.16 in the reference manual for details.

X-Y / X-Z / Y-Z
View X-Y / X-Z / Y-Z plane

Default View
Select the default viewing orientation, defined in the viewing options
2.3.5.3 Add Object

Add Element  
Add a rigid or flexible body.

Add Connector  
Add a joint/constraint/connector or a control element.

Add Load  
Add a load to a rigid or flexible body: generalized coordinate load, body load, force vector, moment vector

Add Material  
Add material for finite elements

Add BeamProperties  
Add the properties of beam elements

Add Node  
Add a node for finite elements

Add Sensor  
Add a sensor in order to measure quantities of the computation: DOF sensor, position sensor, angle sensor, distance sensor, deflection sensor, multiple sensor

Add GeomElement  
Add a geometric element to a body (preferably to rigid bodies): mesh, mesh imported from STL file, cylinder, sphere, cube

Add Set  
Add a set of nodes or elements

For further information, refer to section 2.4 or sections 3.2-3.11 in the reference manual.

2.3.5.4 Edit

Undo  
Undo the last add, delete or edit command

Edit Element, Load, Material,...  
Edit the properties of the already added objects

For further information, refer to section 2.4 or sections 3.2-3.11 in the reference manual.

2.3.5.5 Delete

Delete Element, Load, Material,...  
Delete an already added object

2.3.5.6 System

Show System Properties  
Show some of the properties of the actual multibody system (number of elements, number of coordinates, constraints, etc.)

Verify System  
Check some of the system properties such as if all element, constraint, sensor and geometric element references are valid. Check if constraints and sensors are only attached to valid bodies, etc.

Show global variables  
Access and edit all parameters defined in the model data file

Run Macro (Add variable)  
You can load a (small) txt file in order to enter anything available in the script language. This can be used to add global variables.
2.3.5.7 Computation

**Edit Solver Options**
Access and edit all solver options (such as for time integration, the static solver, the non-linear Newton solver or eigensolver, and settings concerning the in- and output of solution files, sensor data and parameter files. See subsection 2.3.5.8 or section 3.16 in the reference manual for details.

**Save Solver Options**
Save the solver options to a configuration file

**Reset Simulation**
The call to this function is necessary to reset the system to its initial state when it was built. This function is called every time an element is added, removed or changed. The function includes:
- Restore to initial vector stored in elements
- Reset starting time to \( t=0 \)
- Remove all output from data manager
- Assemble the system
- Fit the model onto the screen

**Start Simulation**
Run the simulation from the starting time till the end time using the settings defined in the solver options

**Stop Simulation**
Terminate the simulation

**Pause**
Pause the computation which can be continued later

**Load Initial Vector**
Load a solution vector, which defines the initial conditions of the system, from a file. This vector can be smaller than the actual vector of initial unknowns, e.g. only initial positions can be loaded, while the initial velocities are used from the initial conditions defined in the elements.

**Store Solution Vector**
Store the solution at the current time instance in a file

**Print CPU Statistics**
Prints the approximate usage of CPU power for single parts of the multibody simulation (mass matrix, elastic forces, residual, linear solver, Jacobian, etc.)

2.3.5.8 Results

**Plot Tool Dialog**
Open the dialog for the plot tool which offers creating, editing, scaling, labeling, and exporting plots from one or several sensor signals of the actual simulation or imported from a solution file. See section 2.7 for details.

**Plot Sensor**
Plot the output data of a sensor versus time

**Plot 2 Sensors XY**
Create an XY-Plot from two individually chosen sensor signals

**Sensor Watch**
Open a small window that shows the actual value of a sensor
Enable Output
Enable output written into the output window. The output can be deactivated in order to reduce the computation time for writing into the edit window. This might be especially advantageous for very long simulations.

Show Static Output
Show the output in a separate window which does not update and can be used to analyze or copy the output during the computation.

2.3.5.9 “?”
About
Shows the “About”-dialog with some basic information about HOTINT

Help
Opens the “Help”-contents
2.4 Creating your model in HOTINT

2.4.1 Introduction

Clearly, when working with multibody simulation tools, the subject of model setup and configuration is of central importance. In HOTINT, there are two possibilities to create a multibody system:

- creating a model file using the HOTINT script language (recommended)
- building a system via the graphical user interface (GUI) (not recommended)

Both options shall be illustrated briefly in the following subsections.

2.4.2 Model setup via the script language

2.4.2.1 Script language

The HOTINT script language is a versatile tool which supports a variety of commands for (automatized) generation of multibody system components, such as bodies, loads, or constraints, along with the definition of initial conditions and material parameters. Moreover, variables and, in future versions, certain programming structures (e.g. loops or conditionals), can be used together with a set of mathematical operations similarly to other programming languages. Furthermore, just like the user-defined variables, also any HOTINT option or parameter may be specified via an input file (cf. 2.5.1). Details on the handling of variables and some general remarks with respect to the syntax of the script language are given below; for further information on the HOTINT file and folder structure see section 2.8.

Parser

The Parser used in HOTINT allows to use basic mathematical operations in the model files. Furthermore it is possible to copy parts of the data structure and work with previously defined variables. More details are provided in the following.

Data structure

All assignments in the model file of the form “left-hand side = right-hand side” where the left-hand side names the variable or object that is assigned a value (identifier), and the right-hand side is a number, vector or an evaluable expression (value). Between identifiers and values there may be as many spaces or tabs as desired by the user. However, line breaks need to be set according to the specification.

A valid right hand side entry - or variable name - may include alphanumeric characters and underscores, but no interpunctuation characters; comments start with the % character.

For example, the syntax for the definition of a floating point variable with the identifier “a” and the value 3.0 is simply

\[ a = 3.0 \]

After this definition, “a” can be used and referred to at any point below in the script, for instance in the definition of another variable “b” combined with a basic mathematical operation

\[ b = a \]
Optionally, the data entries can be arranged in named tree-structured containers which can be defined using curly braces. Such containers may hold any set of data entries, and, moreover, can be nested, i.e. can contain other containers as well. Access to each level and entry in these data structures is possible using the "."-operator, similar to the access to (nested) class members in Java or C++. See the following example for clarification:

Assume we want to describe a material – let us call it "m1" – using its elastic modulus "E" and Poisson ratio "nu", we could create a container named "m1" via

```plaintext
m1
{
    E = 1E11
    nu = 0.45
}
```

and access the parameters then via

```plaintext
m1.E
```

or

```plaintext
m1.nu
```

at any point in the file. Note that, within "m1", i.e., within one level in a container, the parameters specified there also may be referred to "directly", e.g. 

```plaintext
m1  E = 1E11  nu = 0.45  temp = 2*E
```

Now, if we had several materials "m1","m2","m3"..., as the one above, we could also define a nested structure "materials" – again a container – holding any of these material containers, for instance

```plaintext
materials
{
    m1
    {
        E = 1E11
        nu = 0.45
    }
    m2
    {
        E = 1.5E11
        nu = 0.47
    }
    m3
    {
        E = 2E11
        nu = 0.46
    }
}
```

where the access works analogously, e.g.

```plaintext
... = materials.m2.E
```

In summary, the entries on the right-hand side in an assignment can be of the following types, depending on the type of the left-hand side:
bool = yes
integer = 1
float = 0.628e1
string = "text"
vector = [1.,2.,3.,4.]
matrix = [1.1,2.1;1.2,2.2]
Container = other_Container

Constants and variables

As shown exemplarily above, it is possible to assign existing variables to new names. The variables on the left-hand side can be accessed by their name and/or location in the data structure. The Parser itself also includes intrinsic constants like pi.

a = 1
b = 2
SubContainer
{
  b = 12
c = 13
}
roota = a
rootb = b
subb = SubContainer.b

Operations

It is also possible to perform simple mathematical operations like adding, multiplying and accessing components on the right-hand side. These features only work on previously assigned variables of the same type.

a = 2
b = 3
vec = [1,2,3]
mat = [1,2;3,4]

% VALID OPERATIONS:
c = a+b
d = a*b
vec2 = vec + vec
two = vec[2]
three = vec[b]
four = mat[2,2]
vec[2] = 7

% NOT WORKING:
vec3 = 3*vec
scalar = vec*vec
mat_succ = mat[ mat[1,1], mat[1,2]]
2.4. CREATING YOUR MODEL IN HOTINT

Built-in functions

Several mathematical functions are implemented in the Parser and can be used in right-hand side expressions. This feature includes

- **power**

  \[ a = \text{sqr}(3) \quad \% \text{ square} \]
  \[ b = \text{sqrt}(a) \quad \% \text{ square root} \]
  \[ c = 2^3 \quad \% \text{ power} \]

- **exponential and logarithm**

  \[ h1 = \text{exp}(5) \quad \% \text{ exponential} \]
  \[ h2 = \text{ln}(h1) \quad \% \text{ logarithm base e} \]
  \[ h3 = \text{log}(1000) \quad \% \text{ logarithm base e} \]
  \[ h4 = \text{log10}(1000) \quad \% \text{ logarithm base 10} \]

- **trigonometric**

  \[ e1 = \sin(\pi/2) \quad \% \text{ sinus function} \]
  \[ f1 = \cos(\pi/2) \quad \% \text{ cosinus function} \]
  \[ g1 = \tan(\pi/2) \quad \% \text{ tangens function} \]
  \[ e2 = \text{asin}(1) \]
  \[ f2 = \text{acos}(1) \]
  \[ g2 = \text{atan}(1) \]
  \[ g22 = \text{atan2}(1,1) \]
  \[ e3 = \text{sinh}(\pi/2) \]
  \[ f3 = \text{cosh}(\pi/2) \]
  \[ g3 = \text{tanh}(\pi/2) \]

- **unitarian operators and functions**

  \[ b = -a \quad \% \text{ change sign} \]
  \[ c = \text{fact}(10) \quad \% \text{ factorial} \]
  \[ i1 = \text{abs}(-273.15) \quad \% \text{ absolute value} \]
  \[ i2 = \text{fabs}(b) \quad \% \text{ absolute value} \]
  \[ d1 = \text{round}(1.61803399) \quad \% \text{ round to nearest integer} \]
  \[ d2 = \text{floor}(1.61803399) \quad \% \text{ next integer lower or equal} \]
  \[ d3 = \text{ceil}(1.61803399) \quad \% \text{ next integer larger or equal} \]
  \[ \text{tam} = \text{transpose(mat)} \quad \% \text{ transpose a matrix} \]
  \[ h = \text{heaviside}(a) \quad \% \text{ heaviside function} \]

- **two parameters**

  \[ \text{one} = \text{min}(1,2) \quad \% \text{ minimum} \]
  \[ \text{two} = \text{max}(1,2) \quad \% \text{ maximum} \]
  \[ \text{three} = \text{max}((\text{min}(1,2),\text{min}(3,4))) \]

- **vectors, and matrices**
2.4.2.2 Model setup

Any consistent file written in the script language ("HOTINT data input file", with ".hid" filename extension; cf. section 2.8) can be loaded and used in HOTINT. In short, it can contain any setting of options for HOTINT itself (see section 2.5 or 3.16 for details), and fully describe the multibody system. On the other hand, if a model is loaded and edited, or created completely via the GUI (see the following subsection 2.4.3), and then saved to a file, the output again will be in terms of the script language. For a detailed description of all supported commands, as well as corresponding example code fragments for illustration, please refer to the reference manual under section 3.15. Sections 3.23.11, on the other hand, contain detailed information about all multibody system components available in HOTINT, i.e. various types of elements such as rigid bodies or structural finite elements such as ANCF beam elements, connectors, loads, sensors, and geometrical elements.

Concludingly, it should be pointed out that the best way to get to know how the whole thing works probably is — as already mentioned — to start and experiment with ready-to-use example files (see also 2.2.2), which are located in the folder documentation/examples in your HOTINT directory and for download at the homepage. See also the the minimal examples in the reference manual.

2.4.3 Model setup via the graphical user interface

The generation and setup of a multibody system via the GUI is more or less self-explanatory: Use the main menu entries "Edit" (cf. 2.3.5.4) and “Add Object” (cf. 2.3.5.3) to edit existing or add new components to the system, specify parameters, and define initial conditions. The model can be saved — as model file in HOTINT script language — at any time. Before an object is added or edited via the GUI, the model is saved automatically. The resulting file is located in the application path and named model_asv.hmc.
However, note that the full functionality and flexibility is only accessible via the direct use of the script language. For details concerning the settings for parameters of single multibody system components please refer to the HOTINT reference manual, sections 3.2-3.11.
2.5 Options Dialogs

2.5.1 Introduction

Via the Windows user interface a wide range of options can be specified to customize HOTINT, concerning, for instance, the graphics, solver or in- and output. The corresponding option dialogs are documented in the following; for a full and detailed listing of all available options refer to the reference manual, section 3.16.

Note that any of these options can be set just like any variable in a script language model file (cf. also subsection 2.4.2) by using its full data name (category + data name according to the options reference). For example, if you would like to specify a maximum time step of 5 ms within the model file, you would just add the line (cf. “TimeInt” in the SolverOptions 3.16.1)

\[ \text{SolverOptions.Timeint.max_step_size} = 0.005 \]

In case of several settings within SolverOptions – or at any other level (such as “Timeint”) for that matter – you may use the syntax as with the nested “data containers” described in the subsection 2.4.2. See the following example for illustration:

```plaintext
SolverOptions
{
    end_time = 1 %1 second simulated time
    Timeint {
        max_step_size = 1e-5 %max. step size for time integration
        min_step_size = 1e-3*max_step_size %min. step size for time integration
    }
    Newton.max_modified_newton_steps = 20 %max. number of modified Newton steps
}
```

which would be equivalent to

```plaintext
SolverOptions.end_time = 1
SolverOptions.Timeint = max_step_size = 1e-5
SolverOptions.Timeint = min_step_size = 1e-3*max_step_size
SolverOptions.Newton.max_modified_newton_steps = 20
```

2.5.2 Hotint Options

Access: View → Edit Hotint Options
2.5. OPTIONS DIALOGS

2.5.2.1 LoggingOptions

LoggingOptions Specify which, how detailed, and in what intervals information concerning the model initialization and solution procedure should be written to the Output-Window and Log-File, respectively

Solver Special configurations for log information concerning the solution procedure

EDCParser Special configurations for log information concerning during the parsing of the model data file

2.5.2.2 GeneralOptions

Application A set of general options concerning the application itself. See “Application” under 3.16.3 in the reference manual for details

Paths Access and set paths of the executable, for the input of input data, and for video/single frame/image/PlotTool image exports

ModelFile See “ModelFile” under 3.16.3 in the reference manual for details

Measurement Choose units for angles and the legend and values of the contour plot

OutputWindow Limit the maximum number of characters in the output window

SavedViewingOptions Define the file names where viewing options shall be stored to when clicking one of the buttons ’1’, ’2’ or ’3’
2.5.2.3 ViewingOptions

**Animation**
Settings specifying how the animation via the Data Manager should be performed.

**Misc**
Various settings concerning the redraw frequency during the simulation, and the thickness or size of points and lines.

**GeomElements**
Settings concerning the GeomElements, e.g. line thickness.

**Origin**
Choose if and how the origin of the coordinate system should be displayed.

**Grid**
Specify and show a background coordinate grid.

**CuttingPlane**
Detailed options for the configuration of up to two cutting planes.

**StandardView**
Define standard views of the system via specification of rotation axes and corresponding angles.

**Bodies**
Options specifying how bodies in general, and rigid bodies and particles in particular, should be drawn and tagged; also includes settings for velocity vectors.

**FiniteElements**
Settings concerning the drawing and coloring of the contour plot, and of finite elements and corresponding meshes and nodes.

**Connectors**
Options specifying if and how constraints should be displayed.

**Loads**
Define if and how loads should be displayed.

**Sensors**
Define if and how sensors should be displayed.

**OpenGL**
Settings for lighting, light sources, transparency, shininess, and color intensity.

**ApplicationWindow**
Size and position of main window of HOTINT.

**DataManager**
Settings concerning the data manager, e.g. how often the solution is stored.

**OutputWindow**
Settings concerning the output window (left of main window).

**View3D**
Define the perspective and sensitivity of mouse movements.

2.5.2.4 PlotToolOptions

**PlotToolOptions**
General setting for the Plot Tool (cf. also section [2.7]), concerning redrawing, scaling, and some size factors for labeling and axis/tick styles.

**DataPoints**
Settings for marking of data points.

**View**
Configuration of size and position of the plot window and the plot itself.
2.5. OPTIONS DIALOGS

**Watches**
Initial size of sensor watch windows

**Axis**
Settings for ticks and labels for both x- and y-axis of the plots

**Grid**
Specification of line types for background coordinate grids in the plots

**Legend**
Specification if and where a legend should be shown

**SavePictures**
Options concerning the export of image files from a plot

### 2.5.3 Viewing Options

**Access:** View → Viewing Options

Viewing options allow changing some of the parameters for visualizing the multibody model:

- **redraw**
  Change the time between subsequent redraws of the model during the simulation in order to speed up the simulation

- **draw origin**
  Draw the origin (0,0,0) and the orientation of the global coordinate system

- **origin size**
  Length of the drawn axis of the origin

- **show contact points**
  If checked, contact points are shown
draw texts in front of bodies

This option will draw texts much closer to the viewer such that they are visible even if they are hidden in reality by an object. However, due to distortion, the texts might appear at slightly different positions.

animation

Your animation will run faster if you draw e.g. only every 10 or 50 frames of the stored computation steps.

animate from beginning

Pressing the animation button will always move to the beginning of the simulation.

OpenGL window size

For screen shots and animation, this lets you adjust the size of the visualization screen in pixels. Best results are obtained if you chose standard resolutions such as 640x480, 800x600, etc.

show startup banner

If checked, the startup banner is shown.

grid

Choose a grid type (orientation), the grid size (length = width), grid step and a grid reference point in order to show a grid for determining positions of the selected model.

standard views

The selection of these parameters allows you to define a standard rotation with respect to the global axis 1, 2 and 3 (= x, y, z) by certain angles. The standard view is x-horizontal and y-vertical, z points out of the x/y plane.

cutting plane

Define a cutting plane by its normal vector and distance from the origin in the direction of the normal; any part of the system lying beyond that plane (in direction of the normal) is cut, i.e. not displayed. A second cutting plane and additional configurations can be defined and accessed via the menu View → Edit Hotint Options → ViewingOptions → CuttingPlane.

2.5.4 OpenGL Drawing Options

Access: View → OpenGL Drawing Options
The OpenGL graphics includes some settings in order to customize the drawing. Yet it is not possible to choose the surface property of a single body, but the material is set for all bodies to the same values, like shininess, transparency, specular color. Sometimes a specific lighting model improves the visibility of an object or the understanding of its geometric complexity. Otherwise the default values can be kept.

There are two independent light sources included, it is possible to activate only one or both lights.

- **enable light** Enable the light source
- **include light position** Include light position in the computation of the intensity. If not checked, objects that are farther away from the light will have the same lighting conditions as near objects
- **ambient** Percentage of ambient light, the intensity of the light is independent of the direction of the light
- **diffuse** Percentage of diffuse light, the brightness is dependent on the position and orientation of the surface with respect to the light source
- **specular light** Percentage of specular light, creates highlight on surfaces like polished metal or mirror-like surfaces.
- **position** Position of the light source
- **transparency** The percentage defines the transparency of the material where 0% is not transparent and 100% is fully transparent. Note that the transparency is dependent on the order of the objects which are currently
not sorted in HOTINT. This can cause strange transparency effects in meshed objects.

**shininess**

This factor defines the radius of shininess of the specular light, 100% = small radius, 0% = very large radius

**specular color intensity**

Defines the amount of specular color reflected by the material

**immediate apply**

If this is activated, all changes in the dialog are immediately applied to the graphics window

**smooth shade model**

Use this to activate smooth shading, which improves the drawing of round surfaces. Otherwise, flat shading is activated (piecewise flat polygons)

**enable lighting**

If not activated, the brightness is not depending on the position of the light with respect to the surface

### 2.5.5 Finite Element Drawing Options

**Access:** View → FE Drawing Options

![Finite element drawing options](image)

#### 2.5.5.1 Contour plot options

**Maximum value**

If activated, the maximum value of the contour plot is limited to the specified value (in the specified units)

**Minimum value**

If activated, the minimum value of the contour plot is limited to the specified value (in the specified units)

**Adjust range**

Auto-adjust the range of the contour plot
2.5. OPTIONS DIALOGS

auto
If activated, the minimum and/or maximum value of the contour plot is chosen automatically, unless it is explicitly specified in the Minimum/Maximum value setting.

color tiling
The number of different colors in the contour iso-plot. The maximum is 32 different colors, a larger value leads to a continuous color.

invert colors
The color bar is inverted.

grey mode
Only black to white colors are used.

nonlinear scale
A nonlinear scale of colors is used. This can be interesting for Mises comparison plots e.g. with edge singularities.

Shrinking factor
The size of the finite elements is multiplied with this factor. Use a value of 1 for displaying the original size and e.g. 0.9 in order to display a reduced view of the elements.

Deformation scale factor
A factor by which all deformations are magnified in the graphic representation. For better visualization of small deformations you may use a large scale factor.

Show variable
The field variable chosen from this list is displayed in the contour plot.

Components
If a non-scalar field variable has been chosen, here the absolute value (magnitude) or component of the field variable which should be displayed in the contour plot can be chosen.

Units
Select units for the chosen field variable.

plot interpolated
If activated, field variables defined on a finite element mesh are plotted interpolated in the contour plot.

animate scaling factor
In order to view eigenmodes or static deformation, the scaling factor can be animated.

scale rigid body displacements
If activated, all rigid body displacements are scaled by the factor specified in the field “Deformation scale factor” (in the graphic representation).

2.5.5.2 Finite element drawing

show mesh
Shows the mesh outlines.

show modes
If checked, modes are shown via Chladni isolines.

show solution
Shows the mesh surface.

draw flat elements
If checked, draw plate elements as flat polygons, otherwise draw plate elements with specified thickness.
show nodes          Shows the nodes of the mesh
show node numbers  Displays the numbers corresponding to the nodes
draw surface elements only
                        If checked, only finite elements on the surface of a mesh are drawn
elem line thickness  Line thickness for element outline
node size           Size of nodes
axis tiling         Tiling specifies the number of quadrangles to draw a curved beam or plate element in axial direction
axis resolution     Resolution specifies the number of quadrangles used to draw the contour solution of a beam or plate element in axial direction
cross-section resolution
                        Resolution specifies the number of quadrangles used to draw the contour solution of a beam or plate element within the transverse direction (discretization of the cross-section)
solid FE resolution Resolution (tiling) used to approximate one solid finite element (triangle, quadrangle, hexahedral, tetrahedral, etc.)

2.5.6 Body / Joint Options

Access: View → Body/Joint Options
2.5. OPTIONS DIALOGS

2.5.6.1 General

**use degrees instead of rad.**

Checked = use degrees (0° - 360°) instead of radians (0 - 2π) for the input of angles and angular velocities. The stored values are always in radians.

**rotation input**

Select input mode for spatial rotations: Euler angles = rotation about Z-X-Z, RotationXYZ = rotation about X-Y-Z, Euler parameters = direct input of 4 Euler parameters.

**show loads**

If activated, all loads in the multibody system are shown.

**load draw size**

Specification of the size of the displayed loads.

2.5.6.2 Rigid Bodies

**show body numbers**

Checked = display element number of the body.

**show body local frame**

Checked = draw local frame of body.

**local frame size**

Drawing size of local body frame.

**bodies transparent**

Checked = draw bodies transparent with factor defined in OpenGL options.

**draw bodies smooth**

Interpolate GeomElement meshes with increased smoothness.

**show body outline**

Checked = draw the outline (edges) of a body or GeomElement.

**show body faces**

Checked = draw the surface of a body or GeomElement → if “show body outline” and “show body faces” is unchecked, the bodies are not drawn.

2.5.6.3 Connectors

**show connectors**

Checked = draw connectors.

**connectors transparent**

Checked = draw connectors transparent with a factor defined in OpenGL options.

**show connector numbers**

Checked = display element number of the connectors.

**show control objects**

Checked = control objects are drawn.

2.5.6.4 Sensors

**show sensors**

Checked = show sensors.

**sensors transparent**

Checked = draw sensors transparent with factor defined in OpenGL options.

**sensor size**

Size of sensor local axes.
2.5.7 Data Manager

Access: View → Show Data Manager

The Data Manager is used to draw the solution at certain time instants where the data has been stored internally. The data is stored either in internal memory or written to the hard disk in the output directory, depending on what was specified for the option Solver Options → Solution → store_data_to_files. Make sure to activate this option in cases where the simulation data would exceed the available main memory. The sliding bar can be used to view certain stored data units and analyze the solution, which is possible even during computation. It is preferable to set the redraw time of the model view very high (→ Viewing options → Redraw) in order to be able to smoothly animate the solution during a long computation. The analysis of the solution during the computation can help to detect model input or convergence errors at an early stage or allows you to run your simulation infinitely (set end time e.g. to 1e6) and to stop the simulation at the point of your consideration.

The button "Run animation" starts the animation either from the beginning (data unit 1) if Display Options → Animate from beginning is set, or otherwise from the current position of the slider bar.

There are two data formats: The .txt format which stores data in pure text (space-separated data):
- line 1: Version identifier
- line 2: checksums, first value = size of data, second value = checksum
- line 3: number of available data units
- line 4: first line of data unit: time, size of data, number1, number2, ....

The .dat format uses windows serialize functions and can not be edited.

Data unit
- Actual data unit drawn

Time
- Actual time instant drawn

delay
- Delay used between frames when running animations

Run animation
- Start animation

Load from a file
- Load a stored solution for animation. Note that only the stored solution that belongs to the same multibody model can be loaded.

Save to a file
- Save the data units into a file. You can choose to save in .txt format which saves the data of each time point in one line (row), or in .dat format. The .dat format is considerably faster and smaller in size.
2.5. OPTIONS DIALOGS

**Save special**

Save selected data units into a file: specify first data unit, last data unit and the increment between stored data units. The data can be stored in .txt, .dat and also as .sol file. Choosing the same number for the first and last data unit allows to use this solution as a .sol solution which can be used as an initial vector for a further computation.

![Save special dialog box](image)

### 2.5.8 Solver Options

**Access:** Computation → Edit Solver Options

![Edit Solver Options dialog box](image)

#### 2.5.8.1 SolverOptions

**SolverOptions** Set start and end time, and choose between dynamic and static computation

**Timeint** Settings concerning the time integration, such as minimum and maximum step size, the maximum index of the differential algebraic equations, or the time integration scheme
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>Settings of the static solver, e.g. concerning load increments</td>
</tr>
<tr>
<td>Newton</td>
<td>Parameters which specify the accuracy goal of the Newton solver, and other options regarding the latter (e.g. settings for numerical differentiation, maximum number of modified or full Newton steps,...)</td>
</tr>
<tr>
<td>Eigensolver</td>
<td>Settings concerning the modal (eigensystem) analysis, such as number of eigenvalues and maximum iterations, the accuracy goal, etc.</td>
</tr>
<tr>
<td>Linalg</td>
<td>Specify whether to use a sparse solver for the solution of the linear systems in the Newton procedure</td>
</tr>
<tr>
<td>Discontinuous</td>
<td>Settings regarding discontinuous systems (e.g. due to friction, contact, etc.)</td>
</tr>
<tr>
<td>Solution</td>
<td>A set of options defining how, in which intervals, and where the solution data and data of the parameter variation procedure should be stored</td>
</tr>
<tr>
<td>Element</td>
<td>Specify whether to store intermediate finite element matrices, and to compute the Jacobians elementwise</td>
</tr>
<tr>
<td>Parameter Variation</td>
<td>Settings concerning the parameter variation procedure: (de)activate, initial and final value, arithmetic or geometric step size, and the path and variable name of the parameter to be varied in the model data input file</td>
</tr>
<tr>
<td>Optimization</td>
<td>Settings regarding the optimization procedure: (de)activate, choice of method, settings for the respective parameters</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Specify if and how the sensitivity of sensor values with respect to certain parameters should be analysed</td>
</tr>
<tr>
<td>Misc</td>
<td>Various settings regarding, for instance, a default model data file, or multithreading in the computation</td>
</tr>
</tbody>
</table>
2.6 Data visualization and graphics export

2.7 Visualization Tool

In HOTINT it is possible to visualize the simulation data with an integrated tool. The Visualization Tool consists of two windows, one containing the most important control elements and a separate window for the plot itself. Both windows can be blinded out if required.

One can display the data directly from the current simulation run or from a file from a previous simulation. The data can be displayed as $y(t)$ using a single data set and also as $y(x)$ when two datasets are combined. The main advantages of using an integrated tool are that we are able to display the data on the fly and create serviceable graphs automatically.

As in most visualization tools each data line can be assigned a color, linestyle and a marker shape. Together with title, labels, positions and other options the graphs’ layout information may be stored for later use. As mentioned above a dialog provides access to the frequently used options. To keep the dialog slim, for both windows an additional context menu is implemented and some hardly ever used options are only available via the full options menu.

The tool is intended for visualization only, so we do not intend to include curve fitting routines to it. Still it is possible to create a consistent dataset for mathematical functions and add those to the graph. For a deep analysis of the result like curve fitting an external program must be used.

The model itself can be programmed such that for selected sensor values a visualization window is automatically created when the model is loaded. For simulations with multiple cycles it is possible to generate graphs with identical properties for comparison.

![Figure 2.2: PlotToolDialog](image)

2.7.0.1 Data Sources

The top section of the Dialog is dedicated to the selection of the data source. The left side allows to pick either the Sensors of the current model or an external (solution) file, most likely a solution file from an other computation. The right part displays the available datasets. With the Buttons any highlighted item in the left list can be added to the right list of drawn lines.

It is possible to plot a line over time ($T/Y$), but also combining two sensors for a ($X/Y$) graph. In this case exactly two lines must be selected in the list.
2.7.0.2 Graph Window

The middle section of the Dialog controls the content of the graph window, on the left side the caption and axis labels as well as the range of the plotted data can be chosen. On the right hand side the properties of an individual line can be changed. Note: the line style can only be changed for thin lines (restriction from Windows Draw function). The general options control the redraw intervals and whether the range is adapted to the full range during a computation.

![Phase Diagram](image)

Figure 2.3: PlotToolGraph. Displays a datasets over time

2.7.0.3 Export

It is possible to export the content of the graph window to a file. Destination folder and filename can be defined in the textboxes. The resolution and all formats for the output can also be chosen.

2.7.0.4 Other Buttons

The remaining individual Buttons in the Dialog have the following effect:

<table>
<thead>
<tr>
<th>Button</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Graph</td>
<td>reactivates the Graph Window</td>
</tr>
<tr>
<td>Hide</td>
<td>hides the control dialog ( reactivate in the Status Bar of the Graph Window )</td>
</tr>
<tr>
<td>Scale Graph</td>
<td>computes the range of the full dataset and rescales the axes accordingly</td>
</tr>
<tr>
<td>Redraw</td>
<td>performs a redraw operation manually (considers auto-rescale flag)</td>
</tr>
<tr>
<td>Axis Equal</td>
<td>forces equal scaling of both axes (mostly used for X-Y-Plots)</td>
</tr>
<tr>
<td>Print Graph</td>
<td>print dialog for the Graph Window</td>
</tr>
<tr>
<td>Update Options</td>
<td>Applies changes made in the HOTINT Options Dialog</td>
</tr>
</tbody>
</table>

The options available in control part of the dialog are only a selection the entire set. Many more are available in the HOTINT Options Dialog, in the subtree PlotToolOptions.
2.7. VISUALIZATION TOOL

Figure 2.4: PlotToolOptions. In this Dialog many settings for the Graph can be done here, e.g. sizes, grid, ...

2.7.1 How to record a video

In order to create a video of your simulation perform the following steps:

- Run the simulation
- Be sure that enough data is stored in data manager
- Create a folder where the image files shall be stored
- Viewing options: set resolution to the desired value (e.g. 1024x768)
- Set all drawing options (with or without mesh, sensors, loads, etc.)
- Remove (drag+drop) all windows (data-manager, options-dialog, etc.) from the main window
- Click on the video-camera button (cf. subsection 2.3.4) to open the “Record frames”-dialog
- Once you have activated the image recorder, images are written at every update of the drawing window, even when the simulation has been stopped and you just resize or move the window
- When you are setting the path where the images will be stored, be sure that the folder already exists and that your path ends with a backslash (e.g. D:\images\ and not D:\images)
• Choose the desired image file format (JPEG, BMP, or PNG)
• Click "Run animation" in the data manager
• Image files are now stored in the specified folder
• Use VideoMach, VirtualDub or comparable software to create a video from the single video frames

Additional hints:
• For video frames export, it is recommended to turn off any screen-saver, start your simulation (or load it from the database) and do not touch it until it has been finished.
• Usually it is preferable to run the simulation first and then use the stored data for the export of images. The whole procedure normally takes a several minutes, which, of course, depends on the complexity of the scene (e.g. number of elements) and the number of video frames.
• Clicking on the button left from the video-camera button lets you store single images into the directory specified above (see also subsection 2.3.4).
• If you are using Windows 7 you have to switch off "aero-design".
2.8 HOTINT File and Folder Structure

In this chapter, the file structure for saving multibody system models is described. The multibody system can be defined in an editable (".hid") format which allows the editing and creation of such files manually or automatically with external programs. However, one needs to be cautious when creating such files, because errors might lead to unexpected results!

The best way to get to know the file structure is to open an existing example file. Details on the HOTINT script language used in those files are provided in section 2.4.2.1.

2.8.1 Input Files

The new version of a text-file containing script language is called Hotint Input Data file - with file extension (".hid"). The file can be opened via the menu with "Open MBS". The filename is then stored in the variable "GeneralOptions.ModelFile.hotint_data_filename". Using the button "Reload MBS" it is possible to open this model again, which allows the user to edit the model in an editor and check the correct implementation with just one click. Alternatively, the Hotint Input Data file can be committed to "hotint.exe" by the drag & drop function of the mouse. If the filetype (".hid") is linked with the application "hotint.exe", the Hotint Input Data file can be opened also by double click of the mouse. A third variant to commit the Hotint Input Data file to HOTINT is to commit the Hotint Input Data file in the DOS-command line e.g. "hotint.exe filename.hid". In all three cases, the directory and filename is stored in the previously described Hotint Option 1. The input file has to contain the variable "HOTINT_data_file_version" before the first command. HOTINT uses this variable to check, if the (old) input file still can be used with the current (new) version of HOTINT.

2.8.2 Folder Structure

The paths are collected in the Options “GeneralOptions.Paths”. Most of them are located in the dialog “Edit Hotint Options”:

- **Application path**: path of the application ("hotint.exe").
- **Record frames path**: path for storage of single frames for creating animations (modify in dialog “Video frames recording/Path to the image”).
- **Hotint input data path**: path of the Hotint Data Input file (".hid").
- **Sensor output path**: path of the solution files from sensors (in dialog “Edit Solver Options”).

---

1Note: the Include-command of the script language searches a file with absolute paths and afterwards relative to the previously described path of the Hotint Data Input file.
Chapter 3

HOTINT Reference Manual

3.1 Preface

In this reference manual all available objects and options are described.

3.1.1 Examples

If there is provided a short example for an object, keep in mind that the examples may not have any physical meaning. The examples just show how to add the object to the system.

3.1.2 Data objects

The description of each object contains a table called Data objects. These are the variables, that can be changed in the GUI or set in the script language. Variables marked with \texttt{R} are \texttt{readonly} and can not be changed by the user.

3.1.3 Observable FieldVariables

If an object provides field variables, they are listed in the documentation of the object. How to measure these variables with a FVElementSensor is described in section 3.9.1.

3.1.4 Observable special values

If an object provides special (internal) values, they are listed in the documentation of the object. How to measure these variables with a ElementSensor is described in section 3.9.2.

3.1.5 Controllable special values

If an object provides special (internal) values, that can be changed during runtime, they are listed in the documentation of the object. How to change these variables with a IOElementDataModifier is described in section 3.4.16.
3.2 Element

These elements are available:

- Mass1D, 3.2.1
- Rotor1D, 3.2.2
- Mass2D, 3.2.3
- Rigid2D, 3.2.4
- Mass3D, 3.2.5
- NodalDiskMass3D, 3.2.6
- Rigid3D, 3.2.7
- Rigid3DKardan, 3.2.8
- Rigid3DMinCoord, 3.2.9
- LinearBeam3D, 3.2.10
- RotorBeamXAxis, 3.2.11
- ANCFBeamShear3DLinear, 3.2.12
- ANCFBeamShear3DQuadratic, 3.2.13
- ANCFBeam3DTorsion, 3.2.14
- Hexahedral, 3.2.15
- Tetrahedral, 3.2.16
- Prism, 3.2.17
- Pyramid, 3.2.18

Note:
In HOTINT several classes are treated as 'elements'. Connectors and control elements are also 'elements', and can therefore be edited and deleted in the GUI with the menu items of the elements.
In the script language the command AddElement is just available for the elements in the list above, but not for connectors or control elements.

3.2.1 Mass1D

Short description

A point mass in one dimensions with 1 position coordinate. The computation of the dynamics of the point mass is extremely simple. The Mass1D can be used for a lot of applications which can be represented by the same type of equations. If you interpret the 'mass' to be 'moment of inertia' and the 'position' to be 'angle', then you can realize a 1D rotatory element as well.
3.2. ELEMENT

Degrees of freedom

1 degree of freedom: the position in x-direction

Geometry

The global position $p_{\text{glob}}$ of a local point $p$ is computed as

$$p_{\text{glob}} = p_0 + A \left( \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix} + p \right)$$

with the reference position $p_0$ and the rotation matrix $A$.

Equations

$$m \ddot{x} = F$$

with the mass $m$ and the force $F$.

Limitations

The mass has no rotations, thus external moments can not be applied. The transformation of local to global coordinates is based on a translation, e.g. the global mass position is added to the local coordinates.

Data objects of Mass1D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>

Figure 3.1: Mass1D
element_type: string "Mass1D" specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!
name: string "Mass1D" name of the element
element_number: integer R 1 number of the element in the mbs
loads: vector [] Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty

Graphics

Graphics.RGB_color: vector [0.1, 0.1, 0.8] [red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]
Graphics.geom_elements: vector [] Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty
Graphics.use_alternative_shape: bool 0 Graphical representation of element with geom-objects that are attached to the element
Graphics.show_element: bool 1 Flag to draw element
Graphics.drawing_tiling: integer 8 tiling of circle/sphere to represent Mass1D; the drawing_tiling should be set small in order to improve efficiency, but large for nice graphical representations
Graphics.radius: double 0.1 drawing radius of mass
Graphics.reference_position: vector [0, 0, 0] Reference point for transformation of 1D objects to 3D; p = [X, Y, Z]
Graphics.rotation_matrix: matrix [1, 0, 0; 0, 1, 0; 0, 0, 1] Rotation matrix for transformation of 1D objects to 3D

Initialization

Initialization.initial_position: vector [0] initial values for position [x]
Initialization.initial_velocity: vector [0] initial values for velocity [v]

Physics

Physics.mass: double 0 total mass of point mass

Observable FieldVariables:
The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, magnitude</td>
</tr>
</tbody>
</table>

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
</table>

3.2. ELEMENT

<table>
<thead>
<tr>
<th>Internal.DOF</th>
<th>degrees of freedom (or generalized unknowns) of the element. range: 1-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.second_order_variable.velocity</td>
<td>velocities of second order variables of the element. range: 1-1</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element:
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10

Example

see file Mass1D.txt

```python
force
{
    load_type = "GCLoad"
    load_value= 1
}
nLoad=AddLoad(force)

Element1
{
    element_type= "Mass1D"
    loads= [nLoad]
    Physics.mass= 1
}
nElement = AddElement(Element1)

data
{
    sensor_type= "FVElementSensor"
    element_number= nElement
    field_variable= "position"
    component= "x"
}
AddSensor(data)
```

3.2.2 Rotor1D

Short description

A rotor with 1 degree of freedom (the rotation). Mathematically implemented like Mass1D but different geometric representation.

Degrees of freedom

1 degree of freedom: the rotation
Geometry

The global position $p_{glob}$ of a local point $p$ is computed as

$$p_{glob} = p_0 + A_0 A p \quad (3.3)$$

with the reference position $p_0$, the constant rotation matrix $A_0$ and the non-constant rotation matrix

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix} \quad (3.4)$$

Equations

$$I \ddot{\varphi} = M \quad (3.5)$$

with the moment of inertia $I$ and the torque $M$.

Figure 3.2: Rotor1D is represented as rotating disc.

Data objects of Rotor1D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rotor1D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Rotor1D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

Graphics
3.2. ELEMENT

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.1, 0.1, 0.8]</td>
<td>red, green, blue] color of element, range = 0..1, use default color [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.rotation_matrix</td>
<td>matrix</td>
<td>[1, 0, 0; 0, 1, 0; 0, 0, 1]</td>
<td>Rotation matrix for transformation of 1D objects to 3D</td>
</tr>
<tr>
<td>Graphics.radius</td>
<td>double</td>
<td>0.1</td>
<td>radius of rotor</td>
</tr>
<tr>
<td>Graphics.length</td>
<td>double</td>
<td>0.2</td>
<td>length of rotor</td>
</tr>
<tr>
<td>Physical fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical.force</td>
<td>vector</td>
<td>[0]</td>
<td>initial value for force</td>
</tr>
<tr>
<td>Physical.moment_of_inertia</td>
<td>double</td>
<td>0.0</td>
<td>mass moment of inertia in kg<em>m</em>m</td>
</tr>
<tr>
<td>Observable FieldVariables:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>bryant_angle</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>angular_acceleration</td>
<td>x, magnitude</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-1</td>
</tr>
</tbody>
</table>
Suitable Connectors:

The following connectors can be used to constrain the element:
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10

Example

see file Rotor1D.txt

```plaintext
force
{
    load_type = "GCLoad"
    load_value= 1
}

nLoad=AddLoad(force)

Element1
{
    element_type= "Rotor1D"
    loads= [nLoad]
    Physics.moment_of_inertia= 1
}

nElement = AddElement(Element1)

senspos
{
    sensor_type= "FVElementSensor"
    element_number= nElement
    field_variable= "bryant_angle"
    component= "x"
}

AddSensor(senspos)
```

3.2.3 Mass2D

Short description

A point mass in two dimensions with 2 position coordinates. The computation of the dynamics of the point mass is extremely simple, thus the Mass2D can be used for many body simulations (e.g. particles).

Degrees of freedom

2 degrees of freedom: the position in 2 coordinates

Equations

\[ m \ddot{x} = F \] (3.6)
3.2. ELEMENT

Limitations

The mass has no rotations, thus external moments can not be applied. The transformation of local to global coordinates is based on a translation, i.e., the global mass position is added to the local coordinates.

![Diagram](image)

**Figure 3.3: Mass2D**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Mass2D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Mass2D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: ‘nr_load1, nr_load2, ...’ or empty</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0.1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set Geometric elements to represent body ‘geomelem1, geomelem2, ...’ or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.reference_position</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Reference point for transformation of planar objects to 3D; $p = [X, Y, Z]$</td>
</tr>
<tr>
<td>Graphics.rotation_matrix</td>
<td>matrix</td>
<td></td>
<td>[1, 0, 0; 0, 1, 0; 0, 0, 1]</td>
<td>Rotation matrix for transformation of planar objects to 3D</td>
</tr>
<tr>
<td>Graphics.drawing_tiling</td>
<td>integer</td>
<td></td>
<td>8</td>
<td>tiling of circle/sphere to represent Mass2D; the drawing_tiling should be set small in order to improve efficiency, but large for nice graphical representations</td>
</tr>
</tbody>
</table>
Graphics.radius | double | 0.1 | drawing radius of mass

**Initialization**

<table>
<thead>
<tr>
<th>Initialization</th>
<th>double</th>
<th>[0, 0]</th>
<th>initial values for position [x,y]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Initialization</th>
<th>double</th>
<th>[0, 0]</th>
<th>initial values for velocity [vx,vy]</th>
</tr>
</thead>
</table>

**Physics**

<table>
<thead>
<tr>
<th>Physics.mass</th>
<th>double</th>
<th>0</th>
<th>total mass of point mass</th>
</tr>
</thead>
</table>

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section [3.1]

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-4</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element: CoordinateConstraint, [3.3.2] VelocityCoordinateConstraint, [3.3.3] MultiCoordConstraint, [3.3.4] FrictionConstraint, [3.3.8] Contact1D, [3.3.9] PlaneConstraint, [3.3.10] SpringDamperActuator2D, [3.3.20] PointJoint2D, [3.3.21]

**Example**

see file mass2D.txt
3.2 ELEMENT

Load1
{
    load_type= "GCLoad" % generalized force (here: actual force)
    generalized_coordinate= 2 % corresponding generalized coordinate
        % (here: y-direction)
    load_value= -0.02
}

nLoad = AddLoad(Load1)

Element1
{
    element_type= "Mass2D"
    loads= [nLoad]
    Initialization.initial_position= [0, 1]
    Physics.mass= 1
}

nElement = AddElement(Element1)

Sensor1
{
    name= "global y-position"
    sensor_type= "FVElementSensor"
    element_number= nElement
    field_variable= "position"
    component= "y"
}

AddSensor(Sensor1)

3.2.4 Rigid2D

Short description

A rigid body in 2D.

Degrees of freedom

The first 2 degrees of freedom are those describing the position in the xy-plane. The rotation around the local z-axis is parameterized with the third degree of freedom.

Geometry

The center of gravity, S, is defined by the vector initial_position, which is in global coordinates. The rotation of the body-fixed local coordinate system w.r.t. the global coordinate system is defined by the variable initial_rotation.

In order to define the position of a point P of the element, e.g. for connectors or sensors, the local coordinate system is used. The reference point is the center of mass, S, so the values of the local coordinates can be positive or negative.
### Data objects of Rigid2D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rigid2D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Rigid2D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

- **Graphics.RGB_color**
  - type: vector
  - default: [0.1, 0.1, 0.8]
  - description: [red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1]
- **Graphics.geom_elements**
  - type: vector
  - default: []
  - description: Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty
- **Graphics.use_alternative_shape**
  - type: bool
  - default: 0
  - description: Graphical representation of element with geom-objects that are attached to the element
- **Graphics.show_element**
  - type: bool
  - default: 1
  - description: Flag to draw element
- **Graphics.reference_position**
  - type: vector
  - default: [0, 0, 0]
  - description: Reference point for transformation of planar objects to 3D; p = [X, Y, Z]
- **Graphics.rotation_matrix**
  - type: matrix
  - default: [1, 0, 0; 0, 1, 0; 0, 0, 1]
  - description: Rotation matrix for transformation of planar objects to 3D
- **Graphics.body_dimensions**
  - type: vector
  - default: [0.1, 0.1, 0.01]
  - description: Dimensions of a regular cube [L_x, L_y, (L_z)]

**Physics**

- **Physics.moment_of_inertia**
  - type: double
  - default: 1.67e-007
  - description: [I_{ZZ}]
- **Physics.mass**
  - type: double
  - default: 0.0001
  - description: mass of the body in kg

**Initialization**

- **Initialization.initial_position**
  - type: vector
  - default: [0, 0]
  - description: [X, Y]
- **Initialization.initial_velocity**
  - type: vector
  - default: [0, 0]
  - description: [vX, vY]
- **Initialization.initial_rotation**
  - type: vector
  - default: [0]
  - description: rotation in rad

---

Figure 3.4: Rigid2D
3.2. ELEMENT

Initialization

<table>
<thead>
<tr>
<th>field variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, magnitude</td>
</tr>
</tbody>
</table>

Observable FieldVariables:

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, magnitude</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section [3.1]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-6</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-3</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-3</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element:
CoordinateConstraint, 3.3.2, VelocityCoordinateConstraint, 3.3.3, MultiCoordConstraint, 3.3.4, FrictionConstraint, 3.3.8, Contact1D, 3.3.9, PlaneConstraint, 3.3.10, SpringDamperActuator2D, 3.3.20, PointJoint2D, 3.3.21.

Example

see file Rigid2D.txt

L_x = 0.10 % length
L_y = 0.20 % width
L_z = 0.01 % height (for drawing and computation of mass)
myRigid2D % add rigid body
{
   element_type= "Rigid2D" %specification of element type.
   name= "my first two-dimensional rigid body" %name of the element
   Graphics.body_dimensions = [L_x, L_y, 0]
   Physics
   {
      mass= density*L_x*L_y*L_z
      moment_of_inertia= 1.0/12.0*mass*(L_x^2+L_y^2)
   }
   Initialization
   {
      initial_position= [0, 0] %[X, Y]
      initial_rotation= [0.0] %rot1_Z in rad
      initial_velocity= [0, 0] %[X, Y]
      initial_angular_velocity= [pi*0.5] %rad/s
   }
}

nElement = AddElement(myRigid2D)

3.2.5 Mass3D

Short description

A point mass in three dimensions with 3 position coordinates. The computation of the dynamics of the point mass is extremely simple, thus the Mass3D can be used for many body simulations (e.g. particles).

Degrees of freedom

3 degrees of freedom: the position in 3 coordinates

Limitations

The mass has no rotations, thus external moments can not be applied. The transformation of local to global coordinates is based on a translation, e.g. the global mass position is added to the local coordinates.

Data objects of Mass3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Mass3D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Mass3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td></td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>
3.2. ELEMENT

<table>
<thead>
<tr>
<th>Graphics.geom_elements</th>
<th>vector</th>
<th>[]</th>
<th>Set Geometric elements to represent body 'geomelem1, geomelem2, …' or empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geometry objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.drawing_tiling</td>
<td>integer</td>
<td>6</td>
<td>tiling of circle/sphere to represent Sphere</td>
</tr>
<tr>
<td>Graphics.radius</td>
<td>double</td>
<td>0.1</td>
<td>drawing radius of Mass</td>
</tr>
</tbody>
</table>

**Initialization**

<table>
<thead>
<tr>
<th>Initialization</th>
<th>vector</th>
<th>[0, 0, 0]</th>
<th>coordinates for initial position of Mass [X Y Z]</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>coordinates for initial velocity of Mass [X Y Z]</td>
</tr>
</tbody>
</table>

**Physics**

| Physics.mass | double | 0 | total mass of point Mass |

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section [3.1]

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-6</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-3</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-3</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element:

Example
see file AddElement.txt

```plaintext
elementMass3D
{
    element_type = "Mass3D"
    Physics.mass= 1
}
nElement = AddElement(elementMass3D)
```

### 3.2.6 NodalDiskMass3D

**Short description**

This is a disk mass for the purpose of rotordynamics applications and should be used together with the RotorBeamXAxis element.

**Nodes**

The DOF of the disk element are stored in a node. To create a new disk element the user has to define a 'Node3DR123' node. This node type has 6 DOF. The first 3 DOF describe the node displacement \((x, y, z)\) w.r.t local rotor element coordinate system, the last 3 DOF are angles of rotation \((\phi_x, \phi_y, \phi_z)\) w.r.t local rotor element coordinate system. The rotation about the local \(x\)-axis is considered as large, the rotations about the local \(y\) and \(z\)-axes are considered as small (linearized angles).

![Figure 3.5: NodalDiskMass3D](image)

**Data objects of NodalDiskMass3D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;NodalDiskMass3D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;NodalDiskMass3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2,...' or empty</td>
</tr>
</tbody>
</table>
### 3.2. ELEMENT

<table>
<thead>
<tr>
<th>Graphics.RGB_color</th>
<th>vector</th>
<th>[0.1, 0.1, 0.8]</th>
<th>red, green, blue] color of element, range = 0..1, use default color [-1,-1,-1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.drawing_tiling</td>
<td>integer</td>
<td>6</td>
<td>tiling of circle/sphere to represent Sphere</td>
</tr>
<tr>
<td>Graphics.thickness</td>
<td>double</td>
<td>0.1</td>
<td>drawing thickness of disk mass</td>
</tr>
<tr>
<td>Graphics.radius</td>
<td>double</td>
<td>0</td>
<td>drawing radius of disk mass</td>
</tr>
</tbody>
</table>

#### Physics

<table>
<thead>
<tr>
<th>Physics.full_mass_matrix</th>
<th>bool</th>
<th>1</th>
<th>set to 1 if influence of tilted mass should be considered in the mass matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.moment_of_inertia</td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>moments of inertia of the disk</td>
</tr>
<tr>
<td>Physics.mass</td>
<td>double</td>
<td>0</td>
<td>total mass of disk</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>1</td>
<td>node number to which the mass refers</td>
</tr>
</tbody>
</table>

---

**Observable Field Variables:**

The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>

---

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-12</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-6</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-6</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>
Suitable Connectors:

The following connectors can be used to constrain the element:
PointJoint, CoordinateConstraint, VelocityCoordinateConstraint, MultiCoordConstraint, Rope3D, FrictionConstraint, Contact1D, PlaneConstraint, GenericBodyJoint, RevoluteJoint, PrismaticJoint, RigidJoint, CylindricalJoint, SpringDamperActuator, RigidLink, RotatorySpringDamperActuator

Example

see file NodalDiskMass3D.txt

% define a node

node {
  node_type = "Node3DR123"
  Geometry {
    reference_position = [0,0,0]
    reference_rot_angles = [0,0,0]
  }
}
n = AddNode(node)

Disk {
  element_type = "NodalDiskMass3D"
  Graphics.radius = 0.2 %radius
  Physics {
    moment_of_inertia = [1, 1, 1] %moments of inertia
    mass = 1 %total mass
  }
  node_number = n %node number to which the mass refers
}
nDisk = AddElement(disk)

3.2.7 Rigid3D

Short description

A rigid body in 3D.

Degrees of freedom

The first 3 degrees of freedom are those describing the position. The rotation is parameterized with 4 degrees of freedom and one additional algebraic equation.
3.2. ELEMENT

Geometry

The center of gravity, $S$, is defined by the vector initial_position, which is in global coordinates, see figure 3.7. The rotation of the body-fixed local coordinate system w.r.t. the global coordinate system is defined by the Matrix initial_rotation.

In order to define the position of a point $P$ of the element, e.g. for connectors or sensors, the local coordinate system is used. The reference point is the center of mass, $S$, so the values of the local coordinates can be positive or negative.

![Figure 3.6: Rigid3D](image1)

![Figure 3.7: local and global coordinate system for a Rigid3D](image2)

Data objects of Rigid3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rigid3D&quot;</td>
<td>specification of element type. Once the element is added to the nbs, you MUST NOT change this type anymore!</td>
</tr>
</tbody>
</table>
### CHAPTER 3. HOTINT REFERENCE MANUAL

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Rigid3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geometrical objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>body_dimensions</td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Dimensions of a regular cube [L_x, L_y, L_z] in m</td>
</tr>
<tr>
<td><strong>Physics</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>moment_of_inertia</td>
<td>matrix</td>
<td>[0.167, 0, 0, 0, 0.167, 0, 0, 0, 0.167]</td>
<td></td>
</tr>
<tr>
<td>volume</td>
<td>double</td>
<td>1</td>
<td>volume of the body in m<em>m</em>m</td>
</tr>
<tr>
<td>mass</td>
<td>double</td>
<td>1</td>
<td>mass of the body in kg</td>
</tr>
<tr>
<td><strong>Initialization</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>initial_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>[X, Y, Z]</td>
</tr>
<tr>
<td>initial_velocity</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>[X, Y, Z]</td>
</tr>
<tr>
<td>initial_rotation</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>3 consecutive rotations (global rotation axes): [rot3_X, rot2_Y, rot1_Z] in rad</td>
</tr>
<tr>
<td>initial_angular_velocity</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Angular velocity vector in global coordinates: [ang_X, ang_Y, ang_Z] in rad/s</td>
</tr>
</tbody>
</table>

**Observable Field Variables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section [3.1]

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1]
### 3.2. ELEMENT

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-15</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element:

- PointJoint, 3.3.1
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- SlidingPointJoint, 3.3.5
- SlidingPrismaticJoint, 3.3.6
- Rope3D, 3.3.7
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10
- GenericBodyJoint, 3.3.11
- RevoluteJoint, 3.3.12
- PrismaticJoint, 3.3.13
- UniversalJoint, 3.3.14
- RigidJoint, 3.3.15
- CylindricalJoint, 3.3.16
- SpringDamperActuator, 3.3.17
- RigidLink, 3.3.18
- RotatorySpringDamperActuator, 3.3.19

**Example**

see file Rigid3D.txt

```plaintext
dimension = [1, 0.1, 0.1] %Dimensions of a regular cube [L_x, L_y, L_z] in m

my_data % compute inertia values
{
    density = 7850
    Cube.body_dimensions = dimension
}
inertia_values = ComputeInertia(my_data)

myRigid % add rigid body
{
    element_type= "Rigid3D" %specification of element type.
    name= "my first rigid" %name of the element
    Graphics.body_dimensions= dimension
    Physics
    {
        moment_of_inertia= inertia_values.moment_of_inertia
        volume= inertia_values.volume
        mass= inertia_values.mass
    }
    Initialization
    {
        initial_position= [0, 0, 0] %[X, Y, Z]
        initial_rotation= [0, pi/2, 0] %[rot3_X, rot2_Y, rot1_Z] in rad
    }
}
```
nElement = AddElement(myRigid)

3.2.8 Rigid3DKardan

Short description
A rigid body in 3D, implemented with bryant angles (also called Tait Bryan or Cardan angles).

Degrees of freedom
The first 3 degrees of freedom are those describing the position. The rotation is parameterized with 3 bryant angles with the sequence x-y-z. If you use this element for dynamic simulation of a fast rotating rigid body, it is advised to use the global x-axis as rotation axis.

Geometry
The center of gravity, S, is defined by the vector initial_position, which is in global coordinates, see figure 3.7. The rotation of the body-fixed local coordinate system w.r.t. the global coordinate system is defined by the Matrix initial_rotation.
In order to define the position of a point P of the element, e.g. for connectors or sensors, the local coordinate system is used. The reference point is the center of mass, S, so the values of the local coordinates can be positive or negative.

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rigid3DKardan&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Rigid3DKardan&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

Figure 3.8: Rigid3DKardan
3.2. ELEMENT

<table>
<thead>
<tr>
<th>loads</th>
<th>vector</th>
<th>[]</th>
<th>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.body_dimensions</td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Dimensions of a regular cube [L_x, L_y, L_z] in m</td>
</tr>
<tr>
<td><strong>Physics</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.moment_of_inertia</td>
<td>matrix</td>
<td>[0.167, 0, 0, 0; 0, 0.167, 0, 0; 0, 0, 0, 0.167]</td>
<td>[I_XX, I_XY, I_XZ; ...]</td>
</tr>
<tr>
<td>Physics.volume</td>
<td>double</td>
<td>1</td>
<td>volume of the body in m<em>m</em>m</td>
</tr>
<tr>
<td>Physics.mass</td>
<td>double</td>
<td>1</td>
<td>mass of the body in kg</td>
</tr>
<tr>
<td>Physics.rotations_sequence</td>
<td>string</td>
<td>&quot;xyz&quot;</td>
<td>rotations sequence, can be xyz, zxy or zxz</td>
</tr>
<tr>
<td><strong>Initialization</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialization.initial_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>[X, Y, Z]</td>
</tr>
<tr>
<td>Initialization.initial_velocity</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>[X, Y, Z]</td>
</tr>
<tr>
<td>Initialization.initial_rotation</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>3 consecutive rotations (global rotation axes): [rot3_X, rot2_Y, rot1_Z] in rad</td>
</tr>
<tr>
<td>Initialization.initial_angular_velocity</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Angular velocity vector in global coordinates: [ang_X, ang_Y, ang_Z] in rad/s</td>
</tr>
</tbody>
</table>

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1.

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section 3.1
### Suitable Connectors:

The following connectors can be used to constrain the element:
- PointJoint, [3.3.1]
- CoordinateConstraint, [3.3.2]
- VelocityCoordinateConstraint, [3.3.3]
- MultiCoordConstraint, [3.3.4]
- SlidingPointJoint, [3.3.5]
- SlidingPrismaticJoint, [3.3.6]
- Rope3D, [3.3.7]
- FrictionConstraint, [3.3.8]
- Contact1D, [3.3.9]
- PlaneConstraint, [3.3.10]
- GenericBodyJoint, [3.3.11]
- RevoluteJoint, [3.3.12]
- PrismaticJoint, [3.3.13]
- UniversalJoint, [3.3.14]
- RigidJoint, [3.3.15]
- CylindricalJoint, [3.3.16]
- SpringDamperActuator, [3.3.17]
- RigidLink, [3.3.18]
- RotatorySpringDamperActuator, [3.3.19]

### Example

see file Rigid3DKardan.txt

dimension = [1, 0.1, 0.1] %Dimensions of a regular cube [L_x, L_y, L_z] in m

my_data % compute inertia values
{
    density = 7850
    Cube.body_dimensions = dimension
}
inertia_values = ComputeInertia(my_data)

myRigid % add rigid body
{
    element_type= "Rigid3DKardan" %specification of element type.
    name= "my first rigid with kardan angles" %name of the element
    Graphics.body_dimensions= dimension
    Physics
    {
        moment_of_inertia= inertia_values.moment_of_inertia
        volume= inertia_values.volume
        mass= inertia_values.mass
    }
    Initialization
    {
        initial_position= [0, 0, 0] %[X, Y, Z]
        initial_rotation= [0, pi/2, 0] %[rot3_X, rot2_Y, rot1_Z] in rad
    }
}
nElement = AddElement(myRigid)
3.2.9 Rigid3DMinCoord

Short description

A rigid body with just one degree of freedom. Efficient formulation for robotic applications are possible with this body.

Degrees of freedom

The body just has 1 (own) degree of freedom (d.o.f.). Depending on the type of joint it is a translational or rotational one, see figure 3.9 and figure 3.10, which rotates or translates with respect to the i-th coordinate system around or along the Z-axis (additionally to the initial parameters $\theta$ and respectively $d$ in figure 3.11). If you look at the i-th body in a chain of such bodies, then the i-th body seems to have i d.o.f.s. In fact it also just adds 1 d.o.f. to the system. If you are using connectors or loads which use a d.o.f. directly (e.g. GCLoad or Coordinate-Constraint) you have to be careful with the settings. In these cases the i-th d.o.f. of the i-th body is the correct one.

Geometry

The reference frame of the body is defined with Denavit-Hartenberg parameters, see figure 3.11 and an (optional) additional rotation matrix. The local reference frame is shown with the following colors: x in green, y in blue and z in red. See figure 3.9 and figure 3.10.

Equations

The implementation is based on the so-called 'Projection Equation' by Bremer. For details see [13].

Limitations

The first body of a chain of such Rigid3DMinCoord bodies has to be fixed to ground. It is not possible yet to connect a robot built up with these elements to e.g. a Rigid3D. The implementation of the translational degree of freedom is up to now just tested for the case, that there is just one transl. d.o.f. and that this joint is the first one in the kinematic chain (=fixed to ground).

Figure 3.9: Rigid3DMinCoord with rotational degree of freedom
Figure 3.10: Rigid3DMinCoord with translational degree of freedom

Figure 3.11: Definition of the geometry with Denavit Hartenberg parameters [14]

**Data objects of Rigid3DMinCoord:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rigid3DMinCoord&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Rigid3DMinCoord&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: &quot;nr_load1, nr_load2, ...&quot; or empty</td>
</tr>
</tbody>
</table>

*Graphics*
### 3.2. ELEMENT

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td></td>
<td>Set Geometric elements to represent body ‘geom1, geom2, …’ or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
<tr>
<td>Graphics.position_offset</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>reference position, global vector to reference frame of first body. Only different from zero for first body!</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th>Geometry Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.prev_body</td>
<td>integer</td>
<td>0</td>
<td>element number of previous body in chain</td>
</tr>
<tr>
<td>Geometry.link_type</td>
<td>integer</td>
<td>1</td>
<td>1...rotation of body i around origin of local body-fixed frame (joint), 2...sliding joint</td>
</tr>
<tr>
<td>Geometry.next_link_position</td>
<td>vector</td>
<td>R [1, 0, 0]</td>
<td>Ir12 vector from origin of local frame (= joint) to end of body. [X Y Z] in first body fixed coordinate system.</td>
</tr>
<tr>
<td>Geometry.joint_local_frame</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Euler angles between global coordinate system to first body or between next_link_rotation and local coordinate system: 3 consecutive rotations (local rotation axes): [rot3_X, rot2_Y, rot1_Z] in rad</td>
</tr>
<tr>
<td>Geometry.DH_parameters</td>
<td>vector</td>
<td>[0, 0, 1, 0]</td>
<td>Denavit-Hartenberg Parameters: [theta (rad), d (m), a (m), alpha (rad)]</td>
</tr>
</tbody>
</table>

#### Physics

<table>
<thead>
<tr>
<th>Physics Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.mass</td>
<td>double</td>
<td>1</td>
<td>mass of the body in kg</td>
</tr>
<tr>
<td>Physics.center_of_gravity</td>
<td>vector</td>
<td>[0.5, 0, 0]</td>
<td>vector from link to center of gravity in local frame. Measured in first body fixed coordinate system.</td>
</tr>
<tr>
<td>Physics.moment_of_inertia</td>
<td>matrix</td>
<td>[0, 0, 0, 0, 0, 0.0833, 0, 0, 0, 0.0833]</td>
<td>[I XX,I XY,I XZ; …] w.r.t. center of gravity, defined in body fixed coordinate system.</td>
</tr>
<tr>
<td>Physics.moment_of_inertia_add</td>
<td>double</td>
<td>0</td>
<td>additional relative inertia moment (e.g. inertia of drive at link side)</td>
</tr>
</tbody>
</table>

#### Initialization

<table>
<thead>
<tr>
<th>Initialization Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization.initial_position</td>
<td>vector</td>
<td>[0]</td>
<td>in m or rad, depending on link_type</td>
</tr>
<tr>
<td>Initialization.initial_velocity</td>
<td>vector</td>
<td>[0]</td>
<td>in m/s or rad/s, depending on link_type</td>
</tr>
</tbody>
</table>

### Observable FieldVariables:

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1.

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angularVelocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>
Observable special values:

For more information see section \[3.1\]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element:
CoordinateConstraint, \[3.3.2\] VelocityCoordinateConstraint, \[3.3.3\] MultiCoordConstraint, \[3.3.4\] FrictionConstraint, \[3.3.8\] Contact1D, \[3.3.9\] PlaneConstraint, \[3.3.10\]

Example

see file Rigid3DMinCoordDoublePendulum.txt

```plaintext
grav.load_type= "Gravity"
ggrav.direction= 2 %global direction of the gravity
ggrav.gravity_constant= -9.81 %use negative sign if necessary
nLoad = AddLoad(grav)

pendulum {
  element_type= "Rigid3DMinCoord"
  loads= [nLoad]
  Geometry
  {
    prev_body= 0 % 0 is constraint to ground = first body in chain
    link_type= 1 % 1...rotational degree of freedom
    DH_parameters= [0, 0, 1, 0] % Denavit-Hartenberg Parameters: [theta (rad), d (m), a (m), alpha (rad)]
  }
  Physics
  {
    mass= 0.1 % mass of the body in kg
    center_of_gravity= [0.5, 0, 0] %vector to center of gravity
    moment_of_inertia= [0, 0, 0
      0, 0.008354166666666666, 0
      0, 0, 0.008354166666666666] %[I XX,I XY,I XZ; ...]
  }
}
AddElement(pendulum)
```
3.2. ELEMENT

% add second pendulum with same geometry and orientation to end of first pendulum
pendulum.Geometry.prev_body= 1  % element number of previous body in chain
AddElement(pendulum)

3.2.10 LinearBeam3D

Short description

The Beam3D element is a three dimensional elastic beam element which is aligned along the local x axis. It provides a decoupled calculation of bending in y and z direction, axial deformation in x direction and torsion about the x axis. Shear deformation is not considered. The decoupled calculation is a simplification of the real, nonlinear problem, but for small deformations the results coincidence highly with the exact solution.

Degrees of freedom

Bending is described by 4 DOF, the number of DOF for axial deformation as well as torsion is 2. These 12 DOF are stored in two nodes i and j. The DOF vector of the LinearBeam3D read as follows

\[ q^{(i)} = [q^{(i)}, q^{(j)}] = [x^{(i)}, y^{(i)}, z^{(i)}, \phi_x^{(i)}, \phi_y^{(i)}, \phi_z^{(i)}, x^{(j)}, y^{(j)}, z^{(j)}, \phi_x^{(j)}, \phi_y^{(j)}, \phi_z^{(j)}]^T. \] (3.7)

Nodes

To create a new beam element the user has to define two 'Node3DRxyz' nodes i and j. Every node of this type has 6 DOF. The first 3 DOF describe the node displacement (x, y, z) w.r.t global coordinate system, the last 3 DOF are angles of rotation (\( \phi_x, \phi_y, \phi_z \)) w.r.t global coordinate system. All angles are considered as small (linearized angles). The reference positions of the nodes define the beam ends at initial configuration and so the length of the beam. The beam orientation is defined due to reference rot angles of node i. The advantage of using nodes with global DOF is the possibility to discretize a beam element into small beams easily without needing complicated constraint conditions. The beam elements do not even have to be aligned along a straight line. If using the same node number for the boundary point of the adjoint beams, beam elements are constrained automatically, see figure 3.13.

Geometry

test

Equations

test

Limitations

Shear deformation is not considered. The decoupled calculation is a simplification of the real, nonlinear problem, but for small deformations the results coincidence highly with the exact solution.
Data objects of LinearBeam3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;LinearBeam3D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;LinearBeam3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td></td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics.RGB_color | vector | [0.1, 0.1, 0.8] | red, green, blue] color of element, range = 0..1, use default color:-1,-1,-1] |
| Graphics.geom_elements | vector | [ ] | Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty |
| Graphics.use_alternative_shape | bool | 0 | Graphical representation of element with geom-objects that are attached to the element |
| Graphics.show_element | bool | 1 | Flag to draw element |

**Geometry**

| Geometry.node_1  | integer | 1 | number of Node 1 |
| Geometry.node_2  | integer | 2 | number of Node 2 |

**Physics**
### 3.2. ELEMENT

| Physical | bool | 1 | include effect of axial deformation |
| Physics.material_number | integer | 1 | material number which contains the main material properties of the beam |

**Observable Field Variables:**

The following values can be measured with a FieldVariableElementSensor,\textsuperscript{3.9.1}. The sensor needs 2 informations: the field_variable itself and the component. For more information see section \textsuperscript{3.1}

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_torsion</td>
<td></td>
</tr>
<tr>
<td>beam_force_axial</td>
<td></td>
</tr>
<tr>
<td>beam_force_transversal</td>
<td>y, z</td>
</tr>
<tr>
<td>beam_moment_torsional</td>
<td></td>
</tr>
<tr>
<td>beam_moment_bending</td>
<td>y, z</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td></td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section \textsuperscript{3.1}

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element, range: 1-24</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element, range: 1-12</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element, range: 1-12</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
<tr>
<td>Internal.acceleration</td>
<td>accelerations of the element, range: 1-12</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element:

PointJoint, \textsuperscript{3.3.1} CoordinateConstraint, \textsuperscript{3.3.2} VelocityCoordinateConstraint, \textsuperscript{3.3.3} Multi-
Example
see file LinearBeam3D.txt

%======================================================================
% define a material
beam_material
{
    material_type = "Beam3DProperties"
    cross_section_type = 1 % rectangular cross section
    cross_section_size = [0.1,0.1]
    density = 1
    EA = 1
    ELy = 1
    EIz = 1
    Gkxy = 1
    Gkxz = 1
    GJkx = 1
    RhoA = 1
    RhoIx = 1
    RhoIy = 1
    RhoIz = 1
}
nBeamMaterial = AddBeamProperties(beam_material)

%======================================================================
% define two nodes
% define two nodes

node1
{
    node_type = "Node3DRxyz"
    Geometry
    {
        reference_position = [0,0,0]
        reference_rot_angles = [0,0,0]
    }
}
n1 = AddNode(node1)

node2
{
    node_type = "Node3DRxyz"
    Geometry
    {
        reference_position = [1,0,0]
        reference_rot_angles = [0,0,0]
    }
}
3.2. ELEMENT

beam
{
  element_type= "LinearBeam3D"
  Physics
  {
    material_number = nBeamMaterial
  }
  Geometry.node_1 = n1
  Geometry.node_2 = n2
}
nBeam = AddElement(beam)

3.2.11 RotorBeamXAxis

Short description

The RotorBeamXAxis element is a three dimensional elastic rotor beam element. It has exact the same characteristics and properties as the LinearBeam3D element except two differences. The first difference is that for a rotor element it is necessary to enable big rotation about the rotor axis instead of the small rotation of the LinearBeam3D. The second difference is that all element DOF are stored w.r.t. local beam coordinate system.

Degrees of freedom

Bending is described by 4 DOF, the number of DOF for axial deformation as well as torsion is 2. These 12 DOF are stored in two nodes i and j. The DOF vector of the LinearBeam3D read as follows

\[
q^{(i)} = [q^{(i)}, q^{(j)}] = [x^{(i)}, y^{(i)}, z^{(i)}, \phi_x^{(i)}, \phi_y^{(i)}, \phi_z^{(i)}, x^{(j)}, y^{(j)}, z^{(j)}, \phi_x^{(j)}, \phi_y^{(j)}, \phi_z^{(j)}]^T. \tag{3.8}
\]

Nodes

To create a new rotor beam element the user has to define two 'Node3DR123' nodes i and j. Every node of this type has 6 DOF. The first 3 DOF describe the node displacement \((x, y, z)\) w.r.t local rotor element coordinate system, the last 3 DOF are angles of rotation \((\phi_x, \phi_y, \phi_z)\) w.r.t local rotor element coordinate system. The rotation about the local x-axis is considered as large, the rotations about the local y and z-axes are considered as small (linearized angles). The reference positions of the nodes define the beam ends at initial configuration and so the length of the beam. The beam orientation is defined due to reference rot angles of node i.

Geometry

The rotor beam geometry is fully defined by 2 'Node3DR123' nodes and a 'Beam3DProperties' material element. Beam length and orientation is specified due to node positions and the beam cross section size due to the material. The rotor beam has a circular cross section.
Figure 3.14: RotorBeamXAxis

Data objects of RotorBeamXAxis:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;RotorBeamXAxis&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;RotorBeamXAxis&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1 number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics.RGB_color | vector | [0.1, 0.1, 0.8] | [red, green, blue] color of element, range = 0.1, use default color:[-1,-1,-1] |
| Graphics.geom_elements | vector | [] | Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty |
| Graphics.use_alternative_shape | bool | 0 | Graphical representation of element with geom-objects that are attached to the element |
| Graphics.show_element | bool | 1 | Flag to draw element |

**Geometry**

| Geometry.node_1   | integer | 1 | number of Node 1 |
| Geometry.node_2   | integer | 2 | number of Node 2 |

**Physics**

| Physics.axial_deformation | bool | 1 | include effect of axial deformation |
| Physics.material_number  | integer | 1 | material number which contains the main material properties of the beam |

**Observable Field Variables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section [3.1]
3.2. ELEMENT

<table>
<thead>
<tr>
<th>position</th>
<th>x, y, z, magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_torsion</td>
<td></td>
</tr>
<tr>
<td>beam_force_axial</td>
<td></td>
</tr>
<tr>
<td>beam_force_transversal</td>
<td>y, z</td>
</tr>
<tr>
<td>beam_moment_torsional</td>
<td></td>
</tr>
<tr>
<td>beam_moment_bending</td>
<td>y, z</td>
</tr>
<tr>
<td>acceleration</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td></td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-24</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-12</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-12</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
<tr>
<td>Internal.acceleration</td>
<td>accelerations of the element. range: 1-12</td>
</tr>
</tbody>
</table>

### Suitable Connectors:

The following connectors can be used to constrain the element:
- PointJoint, 3.3.1
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- SlidingPointJoint, 3.3.5
- SlidingPrismaticJoint, 3.3.6
- Rope3D, 3.3.7
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10
- GenericBodyJoint, 3.3.11
- RevoluteJoint, 3.3.12
- PrismaticJoint, 3.3.13
- UniversalJoint, 3.3.14
- RigidJoint, 3.3.15
- CylindricalJoint, 3.3.16
- SpringDamperActuator, 3.3.17
- RigidLink, 3.3.18
- RotatorySpringDamperActuator, 3.3.19

**Example**

see file RotorBeamXAxis.txt

```
%======================================================================
% define a material
beam_material
{
  material_type = "Beam3DProperties"
```
cross_section_type = 2 \% circular cross section
cross_section_size = [0.1]
density = 1
EA = 1
EIy = 1
EIZ = 1
GAKy = 1
GAKz = 1
GJkx = 1
RhoA = 1
RhoIx = 1
RhoIy = 1
RhoIz = 1
}
nBeamMaterial = AddBeamProperties(beam_material)

%======================================================================
% define two nodes

node1
{
  node_type = "Node3DR123"
  Geometry
  {
    reference_position = [0,0,0]
    reference_rot_angles = [0,0,0]
  }
}
  n1 = AddNode(node1)

node2
{
  node_type = "Node3DR123"
  Geometry
  {
    reference_position = [1,0,0]
    reference_rot_angles = [0,0,0]
  }
}
  n2 = AddNode(node2)

rotor_beam
{
  element_type= "RotorBeamXAxis"
  Physics
  {
    material_number = nBeamMaterial
  }
  Geometry.node_1 = n1
  Geometry.node_2 = n2
}
3.2. ELEMENT

nRotorBeam = AddElement(rotor_beam)

3.2.12 ANCFBeamShear3DLinear

Short description

ANCFBeamShear3DLinear is an ANCF beam finite element for multibody dynamics systems which is capable of large deformations and can be used for static as well as dynamic investigations. The beam finite element can reproduce axial, bending, shear and torsional deformation. A linear interpolation for the geometry and the displacement along the beam axis is chosen. The definition of the beam finite element is based on the absolute nodal coordinate formulation (ANCF), which uses slope vectors for the parameterization of the orientation of the cross section instead of rotational parameters. Two different formulations for the elastic forces of the beam elements are presented:

1. A structural mechanics based formulation of the elastic forces based on Reissner’s nonlinear rod theory including generalized strain measures. A term accounting for thickness and cross section deformation is included and shear locking is prevented.
2. A continuum mechanics based formulation of the elastic forces for a St. Venant Kirchhoff material which avoids the Poisson and shear locking phenomenon.

Degrees of freedom

The degrees of freedom of the \( i \)-th node are the nodal displacements and change of slope vectors and read as follows

\[
\mathbf{q}^{(i)} = [\mathbf{u}^{(i)T} \quad \mathbf{u}_{\eta}^{(i)T} \quad \mathbf{u}_{\zeta}^{(i)T}]^T.
\]  

(3.9)

Hence, nine degrees of freedom are specified in each node, therefore the two-noded linear beam element has 18 degrees of freedom.

Nodes

The element needs 2 nodes of type 'Node3DS2S3'. The element is described by two nodes at the end points of the beam (node 1 = left node, node 2 = right node). See Fig. 3.15 for a sketch of the two-noded linear beam element and the degrees of freedom per node.

Geometry

The deformed geometry of the ANCF beam finite elements is defined by position and two slope vectors in each node, see Fig. 3.15. The slope vectors \( \mathbf{r}_{\eta}^{(i)} \) and \( \mathbf{r}_{\zeta}^{(i)} \) are no unit vectors, therefore a cross section deformation is not prohibited. The displacement along the beam axis is interpolated with linear shape functions, while the orientation of the cross section is interpolated linearly. The slope vectors are the derivative vectors with respect to the coordinate system of the scaled straight reference element, see Fig. 3.16.

Description of the different modi
### CMF

The definition of the elastic forces is based on a continuum mechanics based formulation for a St. Venant Kirchhoff material using the relation between the nonlinear Green-Lagrange strain tensor and the second Piola-Kirchhoff stress tensor. The beam is defined as any other solid finite element and the volume integration can be chosen via the variables order_axial and order_crosssectional in this mode. Using the parameters integration_order_axial (default: 3) and integration_order_cross_section (default: 2) the respective integration orders (using Gaussian integration points) may be defined.

### SMF

The definition of the elastic forces is based on a structural mechanics based formulation based on Reissner’s nonlinear rod theory including generalized strain measures, namely the axial strain, the shear strains, the torsional strain, and the bending strains. The integration along the beam axis is performed as follows: two Lobatto integration points are used for the integration of the elastic forces covering cross section deformation and one Gauss point is used for the integration of the terms accounting for axial deformation, bending, shear and torsion.

### Additional notes

In general: For further details on the definition of the elastic forces, the strain measures or the cross section deformation see reference [15].

Cross section deformation: In order to penalize a possible cross section deformation of the beam, an additional term is added to the classical strain energy and can be varied by the penalization factors named penalty. See reference [15] for more details. Examples: Find static and linearized dynamic applications of the beam element as well as nonlinear dynamic examples and buckling tests in reference [16].
3.2. ELEMENT

Figure 3.15: The geometric description of the elements is based on a position vector \( r^{(i)} \) and two slope vectors \( r_{\eta}^{(i)} \) and \( r_{\zeta}^{(i)} \) in the \( i \)-th node. These vectors are defined on a scaled and straight reference element, given in coordinates \((\xi, \eta, \zeta)\).

Figure 3.16: Different configurations of the finite beam element: (a) scaled straight reference element and (b) the reference element depicted in the global coordinate system.

Data objects of ANCFBeamShear3DLinear:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>element_type</strong></td>
<td>string</td>
<td>&quot;ANCFBeamShear3DLinear&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>&quot;Element&quot; name of the element</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>element_number</strong></td>
<td>integer</td>
<td>R 1 number of the element in the mbs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>loads</strong></td>
<td>vector</td>
<td></td>
<td></td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th><strong>Graphics.RGB_color</strong></th>
<th>vector</th>
<th>[0.1, 0.1, 0.8]</th>
<th>[red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphics.geom_elements</strong></td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Graphics.use_alternative_shape</strong></td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geometric objects that are attached to the element</td>
</tr>
<tr>
<td><strong>Graphics.show_element</strong></td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
</tbody>
</table>

**ShearBeam**

<table>
<thead>
<tr>
<th><strong>ShearBeam.straight_beam</strong></th>
<th>bool</th>
<th>0</th>
<th>is straight beam in reference configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShearBeam.beamformulation</strong></td>
<td>integer</td>
<td>4</td>
<td>2 = CMF, 4 = SMF</td>
</tr>
<tr>
<td><strong>ShearBeam.calc_linear</strong></td>
<td>bool</td>
<td>0</td>
<td>Linearized strain computation in cont. mech. formulation (CMF)</td>
</tr>
<tr>
<td><strong>ShearBeam.nnodes</strong></td>
<td>integer</td>
<td>R 2</td>
<td>number of nodes</td>
</tr>
<tr>
<td><strong>ShearBeam.integration_order_axial</strong></td>
<td>integer</td>
<td>3</td>
<td>Axial integration order</td>
</tr>
<tr>
<td><strong>ShearBeam.integration_order_cross_section</strong></td>
<td>integer</td>
<td>2</td>
<td>Cross section integration order, takes effect only in cont. mech. formulation (CMF)</td>
</tr>
<tr>
<td><strong>ShearBeam.ip_number_per_disc_quadrant</strong></td>
<td>integer</td>
<td>1</td>
<td>Number of integration points per disc quadrant in angular direction, required if cross_section_type of Beam3DProperties is circular or tubular</td>
</tr>
<tr>
<td><strong>ShearBeam.penalty_kappa</strong></td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Penalty term for kappa [kappa1,kappa2,kappa3]</td>
</tr>
<tr>
<td><strong>ShearBeam.penalty_gamma</strong></td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Penalty term for gamma [gamma1,gamma2,gamma3]</td>
</tr>
<tr>
<td><strong>ShearBeam.penalty_E</strong></td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Penalty term for green lagrange strains (E) [Eyy,Ezz,Eyz]</td>
</tr>
</tbody>
</table>

**Geometry**

<table>
<thead>
<tr>
<th><strong>Geometry.body_dimensions</strong></th>
<th>vector</th>
<th>[1, 0.1, 0.1]</th>
<th>Dimensions of the beam. [L_x (length), L_y (width), L_z (height)]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geometry.node_number1</strong></td>
<td>integer</td>
<td>1</td>
<td>Global number of node 1 (left), node must already exist</td>
</tr>
<tr>
<td><strong>Geometry.node_number2</strong></td>
<td>integer</td>
<td>2</td>
<td>Global number of node 2 (right), node must already exist</td>
</tr>
</tbody>
</table>

**Physics**

| **Physics.material_number** | integer | 1 | Material number which contains the main material properties of the beam                                              |

**Initialization**

<table>
<thead>
<tr>
<th><strong>Initialization.node1_reference_position</strong></th>
<th>vector</th>
<th>[0, 0, 0]</th>
<th>Position of node 1 (left) in reference configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization.node1_reference_slope_2</strong></td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Slope vector 2 of node 1 (left) in reference configuration.</td>
</tr>
<tr>
<td><strong>Initialization.node1_reference_slope_3</strong></td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Slope vector 3 of node 1 (left) in reference configuration.</td>
</tr>
<tr>
<td><strong>Initialization.node2_reference_position</strong></td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Position of node 2 (right) in reference configuration</td>
</tr>
</tbody>
</table>
3.2. ELEMENT

**Initialization.**

<table>
<thead>
<tr>
<th>node2_reference_slope_2</th>
<th>vector</th>
<th>[0, 0, 0]</th>
<th>slope vector 2 of node 2 (right) in reference configuration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>node2_reference_slope_3</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>slope vector 3 of node 2 (right) in reference configuration.</td>
</tr>
</tbody>
</table>

**RotorDynamics**

| angular_velocity | double | 0 | Element is rotating with angular_velocity around axis. |

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>rotations_312</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_curvature</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_torsion</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_moment_bending</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_moment_torsional</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_shear</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_axial_extension</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_force_transversal</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_force_axial</td>
<td>x, y, z, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.kinetic_energy</td>
<td>force in the rope</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>length of the rope</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element:

PointJoint, 3.3.1 CoordinateConstraint, 3.3.2 VelocityCoordinateConstraint, 3.3.3 MultiCoordConstraint, 3.3.4 SlidingPointJoint, 3.3.5 SlidingPrismaticJoint, 3.3.6 Rope3D, 3.3.7 FrictionConstraint, 3.3.8 Contact1D, 3.3.9 PlaneConstraint, 3.3.10 GenericBodyJoint, 3.3.11
RevoluteJoint, PrismaticJoint, UniversalJoint, RigidJoint, CylindricalJoint, SpringDamperActuator, RigidLink, RotatorySpringDamperActuator.

Example

see file ANCFBeamShear3DLinear.txt

```plaintext
my_material {
    material_type= "Beam3DProperties"
    cross_section_type= 1
    cross_section_size= [0.1, 0.1]
    EA= 20000
    EIy= 16.66666666666667
    EIz= 16.66666666666667
    GAkxy= 7692.307692307694
    GAkzz= 7692.307692307694
    GJkx= 10.81538461538462
    RhoA= 72
    RhoIx= 0.12
    RhoIy= 0.06
    RhoIz= 0.06
    density= 7200
}

nMaterial = AddBeamProperties(my_material)

node {
    node_type = "Node3DS2S3"
    Geometry.reference_position = [0,0,0]
    Geometry.reference_slope2 = [0,1,0]
    Geometry.reference_slope3 = [0,0,1]
}
nNode1 = AddNode(node)

node.Geometry.reference_position = [1,0,0]
nNode2 = AddNode(node)

my_beam {
    element_type = "ANCFBeamShear3DLinear"
    Physics.material_number = nMaterial
    ShearBeam.beamformulation = 4
    Geometry.node_number1 = nNode1
    Geometry.node_number2 = nNode2
}
nElement = AddElement(my_beam)

ViewingOptions.FiniteElements.Nodes.show = 1
ViewingOptions.FiniteElements.Nodes.node_size = 0.05
```
3.2.13 ANCFBeamShear3DQuadratic

Short description

ANCFBeamShear3DQuadratic is an ANCF beam finite element for multibody dynamics systems which is capable of large deformations and can be used for static as well as dynamic investigations. The beam finite element can reproduce axial, bending, shear and torsional deformation. A quadratic interpolation for the geometry and the displacement along the beam axis is chosen.

The definition of the beam finite element is based on the absolute nodal coordinate formulation (ANCF), which uses slope vectors for the parameterization of the orientation of the cross section instead of rotational parameters. Two different formulations for the elastic forces of the beam elements are presented:

1. A structural mechanics based formulation of the elastic forces based on Reissner’s nonlinear rod theory including generalized strain measures. A term accounting for thickness and cross section deformation is included and shear locking is prevented.

2. A continuum mechanics based formulation of the elastic forces for a St.Venant Kirchhoff material which avoids the Poisson and shear locking phenomenon.

Degrees of freedom

The degrees of freedom of the \( i \)-th node are the nodal displacements and change of slope vectors and read as follows

\[
q^{(i)} = [u^{(i)T} \quad u^{(i)\eta}_\eta \quad u^{(i)\chi}_\chi]^T.
\]  

(3.10)

Hence, nine degrees of freedom are specified in each node, therefore the three-noded quadratic beam element has 27 degrees of freedom.

Nodes

The element needs 3 nodes of type 'Node3DS2S3'. The element is described by three nodes: at the end points and the mid point of the beam (node 1 = left node, node 2 = right node, node 3 = mid point). See Fig. 3.15 for a sketch of the three-noded quadratic beam element and the degrees of freedom per node.

Geometry

The deformed geometry of the ANCF beam finite elements is defined by position and two slope vectors in each node, see Fig. 3.15. The slope vectors \( r^{(i)}_\eta \) and \( r^{(i)}_\chi \) are no unit vectors, therefore a cross section deformation is not prohibited. The displacement along the beam axis is interpolated with quadratic shape functions, while the orientation of the cross section is interpolated linearly. The slope vectors are the derivative vectors with respect to the coordinate system of the scaled straight reference element, see Fig. 3.16.

Description of the different modi
### Additional notes

In general: For further details on the definition of the elastic forces, the strain measures or the cross section deformation see reference [15].

Cross section deformation: In order to penalize a possible cross section deformation of the beam, an additional term is added to the classical strain energy and can be varied by the penalization factors named penalty. See reference [15] for more details.

Examples: Find static and linearized dynamic applications of the beam element as well as nonlinear dynamic examples and buckling tests in reference [16].

| CMF | The definition of the elastic forces is based on a continuum mechanics based formulation for a St.Venant Kirchhoff material using the relation between the nonlinear Green-Lagrange strain tensor and the second Piola-Kirchhoff stress tensor. The beam is defined as any other solid finite element and the volume integration can be chosen via the variables order_axial and order_crosssectional in this modus. Using the parameter perform_reduced_integration, the standard integration order is reduced, in order to reduce stiffening effects or computation time. |
| SMF | The definition of the elastic forces is based on a structural mechanics based formulation based on Reissner’s nonlinear rod theory including generalized strain measures, namely the axial strain, the shear strains, the torsional strain, and the bending strains. The integration along the beam axis is performed as follows: two Lobatto integration points are used for the integration of the elastic forces covering cross section deformation and one Gauss point is used for the integration of the terms accounting for axial deformation, bending, shear and torsion. |
3.2. ELEMENT

Figure 3.17: The geometric description of the elements is based on a position vector $r^{(i)}$ and two slope vectors $r_{\xi}^{(i)}$ and $r_{\eta}^{(i)}$ in the $i$-th node. These vectors are defined on a scaled and straight reference element, given in coordinates $(\xi, \eta, \zeta)$.

Figure 3.18: Different configurations of the finite beam element: (a) scaled straight reference element and (b) the reference element depicted in the global coordinate system.

Data objects of ANCFBeamShear3DQuadratic:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;ANCFBeamShear3D Quadratic&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Element&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics RGB color   | vector | [0.1, 0.1, 0.8] red, green, blue color of element, range = 0..1, use default color: [-1,-1,-1] |
| Graphics geom elements| vector | [] Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty |
| Graphics use alternative shape| bool | 0 Graphical representation of element with geom-objects that are attached to the element |
| Graphics show element | bool | 1 Flag to draw element |

**ShearBeam**

| ShearBeam straight beam | bool | 0 is straight beam in reference configuration |
| ShearBeam beam formulation | integer | 4 2 = CMF, 4 = SMF |
| ShearBeam calc linear | bool | 0 Linearized strain computation in cont. mech. formulation (CMF) |
| ShearBeam modes | integer | R 3 number of nodes |
| ShearBeam integration order axial | integer | 3 Axial integration order |
| ShearBeam integration order cross section | integer | 2 Cross section integration order, takes effect only in cont. mech. formulation (CMF) |
| ShearBeam ip number per disc quadrant | integer | 1 Number of integration points per disc quadrant in angular direction, required if cross_section_type of Beam3DProperties is circular or tubular |
| ShearBeam penalty kappa | vector | [1, 1, 1] Penalty term for kappa [kappa1,kappa2,kappa3] |
| ShearBeam penalty gamma | vector | [1, 1, 1] Penalty term for gamma [gamma1,gamma2,gamma3] |
| ShearBeam penalty E | vector | [1, 1, 1] Penalty term for green lagrange strains (E) [Eyy,Ezz,Eyz] |

**Geometry**

| Geometry body dimensions | vector | [1, 0.1, 0.1] Dimensions of the beam. [L_x (length), L_y (width), L_z (height)] |
| Geometry node number 1 | integer | 1 Global number of node 1 (left), node must already exist |
| Geometry node number 2 | integer | 2 Global number of node 2 (right), node must already exist |
| Geometry node number 3 | integer | 3 Global number of node 3 (middle), node must already exist |

**Physics**

| Physics material number | integer | 1 Material number which contains the main material properties of the beam |

**Initialization**

| Initialization node1 reference position | vector | [0, 0, 0] Position of node 1 (left) in reference configuration |
| Initialization node1 reference slope 2 | vector | [0, 0, 0] Slope vector 2 of node 1 (left) in reference configuration |
| Initialization node1 reference slope 3 | vector | [0, 0, 0] Slope vector 3 of node 1 (left) in reference configuration |
3.2. ELEMENT

Initialization.
node2_reference_position vector [0, 0, 0] position of node 2 (right) in reference configuration.
Initialization.
node2_reference_slope_2 vector [0, 0, 0] slope vector 2 of node 2 (right) in reference configuration.
Initialization.
node2_reference_slope_3 vector [0, 0, 0] slope vector 3 of node 2 (right) in reference configuration.
Initialization.
node3_reference_position vector [0, 0, 0] position of node 3 (middle) in reference configuration.
Initialization.
node3_reference_slope_2 vector [0, 0, 0] slope vector 2 of node 3 (middle) in reference configuration.
Initialization.
node3_reference_slope_3 vector [0, 0, 0] slope vector 3 of node 3 (middle) in reference configuration.

RotorDynamics
angular_velocity double 0 Element is rotating with angular_velocity around axis.

Observable FieldVariables:
The following values can be measured with a FieldVariableElementSensor, [3.9.1]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>bryant_angle</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>rotations_312</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>angular_velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td></td>
</tr>
<tr>
<td>beam_curvature</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_torsion</td>
<td></td>
</tr>
<tr>
<td>beam_moment_bending</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_moment_torsional</td>
<td></td>
</tr>
<tr>
<td>beam_shear</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_axial_extension</td>
<td></td>
</tr>
<tr>
<td>beam_force_transversal</td>
<td></td>
</tr>
<tr>
<td>beam_force_axial</td>
<td></td>
</tr>
</tbody>
</table>

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.kinetic_energy</td>
<td>force in the rope</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>length of the rope</td>
</tr>
</tbody>
</table>
Suitable Connectors:

The following connectors can be used to constrain the element:
PointJoint, CoordinateConstraint, VelocityCoordinateConstraint, Multi-CoordConstraint, SlidingPointJoint, SlidingPrismaticJoint, Rope3D, FrictionConstraint, Contact1D, PlaneConstraint, GenericBodyJoint, RevoluteJoint, PrismaticJoint, UniversalJoint, RigidJoint, CylindricalJoint, SpringDamperActuator, RigidLink, RotatorySpringDamperActuator

Example

see file ANCFBeamShear3DQuadratic.txt

```plaintext
my_material
{
  material_type= "Beam3DProperties"
  cross_section_type= 1
  cross_section_size= [0.1, 0.1]
  EA= 20000
  EIy= 16.66666666666667
  EIz= 16.66666666666667
  Gkxy= 7692.307692307694
  Gkxz= 7692.307692307694
  GJkx= 10.81538461538462
  RhoA= 72
  RhoIy= 0.12
  RhoIz= 0.06
  density= 7200
}

nMaterial = AddBeamProperties(my_material)

node
{
  node_type = "Node3DS2S3"
  Geometry.reference_position = [0,0,0]
  Geometry.reference_slope2 = [0,1,0]
  Geometry.reference_slope3 = [0,0,1]
}

nNode1 = AddNode(node)

node.Geometry.reference_position = [1,0,0]

nNode2 = AddNode(node)

node.Geometry.reference_position = [0.5,0,0]

nNode3 = AddNode(node)

my_beam
{
  element_type = "ANCFBeamShear3DQuadratic"
  Physics.material_number = nMaterial
}
3.2. ELEMENT

\begin{verbatim}
ShearBeam.beamformulation = 4
Geometry.node_number1 = nNode1
Geometry.node_number2 = nNode2
Geometry.node_number3 = nNode3

nElement = AddElement(my_beam)

ViewingOptions.FiniteElements.Nodes.show = 1
ViewingOptions.FiniteElements.Nodes.node_size = 0.05
\end{verbatim}

3.2.14 ANCFBeam3DTorsion

Short description

ANCFBeam3DTorsion is a Bernoulli-Euler beam finite element in ANCF (Absolute Nodal Coordinate Formulation) capable of large axial, bending, and torsional deformations.

Degrees of freedom

The element affects 14 degrees of freedom (generalized coordinates) in total, which are 7 degrees of freedom per node, i.e., at each node we have: the axial displacement \( u = r - r_0 \), the change of the axial slope \( u' = r' - r'_0 \), and the change of the torsional angle \( \theta - \theta_0 \). Each quantity with index 0 here conforms to the reference configuration. The element wise ordering of the degrees of freedom is displayed in Fig. 3.20.

Nodes

The element operates with two Nodes of type Node3DS1rot1, each of which are located at either tip of the beam element. The integer values Geometry.node_number1 and Geometry.node_number2 refer to the index of the nodes in the multibody system. Each of these Nodes is instantiated by the user with a position and a rotation (kardan angles), and provides a frame \((e_1, e_2, e_3)\) (which is measured in the global frame of the multibody system) for the instantiation of the beam element: At each node, the slope of the beam axis \( r' \) is identical with \( e_1 \), and the director is defined as \( e_3 \).

Geometry

The geometry of the element is defined by the nodal values for axial position \( r \), the axial slope vector \( r' \), and the torsional angle \( \theta \) of the cross section around the beam axis, see Fig. 3.20. This angle is measured with respect to a reference direction in the global frame (director). Between the nodal values, the axial position is interpolated cubically, the axial slope is interpolated quadratically, and the torsional angle of the cross section (around the beam axis) as well as the director are interpolated linearly.

Equations

Variation of the strain energy:

\[
\delta \Pi = \int_{-L/2}^{L/2} \left( EA \delta \varepsilon + GJ_k \delta k_1 + EI_2 \delta k_2 + EI_3 \delta k_3 \right) d\xi. \quad (3.11)
\]
Considering viscous material damping, we replace \( \varepsilon \to \varepsilon^E \) and \( \kappa_i \to \kappa_i^E \) for \( i \in \{1, 2, 3\} \) with

\[
\varepsilon^E = \varepsilon + \varepsilon^D, \quad \varepsilon^D = c_K \dot{\varepsilon} = c_K \left( \frac{\partial \varepsilon}{\partial \mathbf{q}} \right)^T \dot{\mathbf{q}},
\]

\[
\kappa_i^E = \kappa_i + \kappa_i^D, \quad \kappa_i^D = c_K \dot{\kappa}_i = c_K \left( \frac{\partial \kappa_i}{\partial \mathbf{q}} \right)^T \dot{\mathbf{q}},
\]

resulting in

\[
\delta \Pi = \int_{-L/2}^{L/2} \left( EA (\varepsilon + c_K \dot{\varepsilon}) \delta \varepsilon + GJ (\kappa_1 + c_K \dot{\kappa}_1) \delta \kappa_1 + E I_y (\kappa_2 + c_K \dot{\kappa}_2) \delta \kappa_2 + E I_z (\kappa_3 + c_K \dot{\kappa}_3) \delta \kappa_3 \right) d\xi.
\]

Additional notes

For details on the element, such as the definition of the elastic forces and the kinetic terms, see [17, 18].

Figure 3.19: The geometry of the element is defined by nodal values for (a) the axial position, (b) the axial slope vector, and (c) the torsional angle of the cross section around the beam axis. This angle is measured with respect to a reference direction in the global frame (director). Between the nodal values, the axial position is interpolated cubically, the axial slope is interpolated quadratically, and the torsional angle of the cross section (around the beam axis) as well as the director are interpolated linearly.
3.2. ELEMENT

Figure 3.20: Ordering of the generalized coordinates.

Data objects of ANCFBeam3DTorsion:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;ANCFBeam3DTorsion&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;ANCFBeam3DTorsion&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[ ]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2,...' or empty</td>
</tr>
</tbody>
</table>

Graphics

| Graphics.RGB_color   | vector      | [0.1, 0.1, 0.8] | [red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.geom_elements | vector      | [ ]            | Set Geometric elements to represent body 'geoelem1, geoelem2,...' or empty   |
| Graphics.use_alternative_shape | bool | 0              | Graphical representation of element with geom-objects that are attached to the element |
| Graphics.show_element | bool        | 1              | Flag to draw element                                                        |

Geometry

| Geometry.node_number1 | integer     | 1            | global number of node 1 (left), node must already exist                     |
| Geometry.node_number2 | integer     | 2            | global number of node 2 (right), node must already exist                    |
| Geometry.update_directors | bool | 0            | update directors during calculation                                          |

Computation

| Computation.kinematic_computation_mode | integer | 0            | 0 .. exact kinematic terms + 5th order gaussian integration (slow), 1 .. exact terms + 1st order lobatto integration (fast), 2 .. constant mass matrix approximation (fastest) |
| Computation.IntegrationOrder.mass     | integer  | 4            | integration order for mass terms                                            |
| Computation.IntegrationOrder.axial_strain | integer | 9            | integration order for work of axial strain                                  |
| Computation.IntegrationOrder.curvature | integer  | 5            | integration order for work of curvature                                     |

Physics
Observable FieldVariables:

The following values can be measured with a FieldVariableElementSensor, \[\text{3.9.1}\]. The sensor needs 2 informations: the field_variable itself and the component. For more information see section \[\text{3.1}\].

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity_local_basis</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_axial_extension</td>
<td></td>
</tr>
<tr>
<td>beam_force_axial</td>
<td></td>
</tr>
<tr>
<td>beam_curvature</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_moment</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>beam_torsion</td>
<td></td>
</tr>
<tr>
<td>beam_moment_torsional</td>
<td></td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section \[\text{3.1}\].

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-28</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-14</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-14</td>
</tr>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which have no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-6</td>
</tr>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element: PointJoint, \[\text{3.3.1}\] CoordinateConstraint, \[\text{3.3.2}\] VelocityCoordinateConstraint, \[\text{3.3.3}\] MultiCoordConstraint, \[\text{3.3.4}\] Rope3D, \[\text{3.3.7}\] FrictionConstraint, \[\text{3.3.8}\] Contact1D, \[\text{3.3.9}\] PlaneConstraint, \[\text{3.3.10}\] GenericBodyJoint, \[\text{3.3.11}\] RevoluteJoint, \[\text{3.3.12}\] PrismaticJoint, \[\text{3.3.13}\] UniversalJoint, \[\text{3.3.14}\] RigidJoint, \[\text{3.3.15}\] CylindricalJoint, \[\text{3.3.16}\] SpringDamperActuator, \[\text{3.3.17}\] RigidLink, \[\text{3.3.18}\].
3.2. ELEMENT

Example

see file ANCFBeam3DTorsion.txt

node
{
    node_type= "Node3DS1rot1"
    Geometry
    {
        reference_position= [0, 0, 0]
        reference_rot_angles= [0, 0, 0]
    }
}

nNode1 = AddNode(node)

node.Geometry.reference_position = [1,0,0]
nNode2 = AddNode(node)

beamproperties
{
    material_type= "Beam3DProperties"
    cross_section_type= 1
    cross_section_size= [0.1, 0.1]
    EA= 2100000000
    EIy= 1750000
    EIz= 1750000
    GJkx= 2692307.692307693
}

nBeamProperties = AddBeamProperties(beamproperties)

element
{
    element_type= "ANCFBeam3DTorsion"
    loads= [1]
    Physics
    {
        material_number= nBeamProperties
    }
    Geometry
    {
        node_number1= nNode1
        node_number2= nNode2
    }
}

AddElement(element)

3.2.15 Hexahedral

Short description

The Hexahedral is a 3D element with 8 or 20 nodes, 6 faces with 4 nodes each. The local coordinates ranges \([-1,+1]\) for \(\{x,y,z\}\).
order 1 nodes

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(-1,-1,-1)</td>
</tr>
<tr>
<td>2</td>
<td>(+1,-1,-1)</td>
</tr>
<tr>
<td>3</td>
<td>(-1,+1,-1)</td>
</tr>
<tr>
<td>4</td>
<td>(+1,+1,-1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(-1,-1,1)</td>
</tr>
<tr>
<td>6</td>
<td>(+1,-1,1)</td>
</tr>
<tr>
<td>7</td>
<td>(-1,+1,1)</td>
</tr>
<tr>
<td>8</td>
<td>(+1,+1,1)</td>
</tr>
</tbody>
</table>

additional order 2 nodes

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>(0,-1,-1)</td>
</tr>
<tr>
<td>10</td>
<td>(0,+1,-1)</td>
</tr>
<tr>
<td>11</td>
<td>(0,-1,+1)</td>
</tr>
<tr>
<td>12</td>
<td>(0,+1,+1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>(-1,0,-1)</td>
</tr>
<tr>
<td>14</td>
<td>(+1,0,-1)</td>
</tr>
<tr>
<td>15</td>
<td>(-1,0,+1)</td>
</tr>
<tr>
<td>16</td>
<td>(+1,0,+1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>(-1,-1,0)</td>
</tr>
<tr>
<td>18</td>
<td>(+1,-1,0)</td>
</tr>
<tr>
<td>19</td>
<td>(-1,+1,0)</td>
</tr>
<tr>
<td>20</td>
<td>(+1,+1,0)</td>
</tr>
</tbody>
</table>

node numbers of the element faces - counter-clockwise from outside

<table>
<thead>
<tr>
<th>FaceNr</th>
<th>O1</th>
<th>O2</th>
<th>FaceCoordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,5,7,3</td>
<td>1,5,7,3,17,15,19,13</td>
<td>x' = (z+1)/2, y' = (y+1)/2</td>
</tr>
<tr>
<td>2</td>
<td>2,4,8,6</td>
<td>2,4,8,6,14,20,16,18</td>
<td>x' = (y+1)/2, y' = (z+1)/2</td>
</tr>
<tr>
<td>3</td>
<td>1,2,6,5</td>
<td>1,2,6,5,9,18,11,17</td>
<td>x' = (x+1)/2, y' = (z+1)/2</td>
</tr>
<tr>
<td>4</td>
<td>3,7,8,4</td>
<td>3,7,8,4,19,12,20,10</td>
<td>x' = (z+1)/2, y' = (x+1)/2</td>
</tr>
<tr>
<td>5</td>
<td>3,4,2,1</td>
<td>3,4,2,1,10,14,9,13</td>
<td>x' = (x+1)/2, y' = (1-y)/2</td>
</tr>
<tr>
<td>6</td>
<td>5,6,8,7</td>
<td>5,6,8,7,11,16,12,15</td>
<td>x' = (x+1)/2, y' = (y+1)/2</td>
</tr>
</tbody>
</table>

Degrees of freedom

3 degrees of freedoms per node (24 or 60), arranged by node, \{N_1x, N_1y, N_1z, N_2x, \ldots N_{20}z\}

Limitations

The Hexahedral has only position DOFs
Data objects of Hexahedral:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Hexahedral&quot;</td>
<td>specification of element type. Once the element is added to the mesh, you MUST NOT change this type anymore!</td>
</tr>
</tbody>
</table>
### CHAPTER 3. HOTINT REFERENCE MANUAL

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Element&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>int</td>
<td>integer number of the element in the mdb</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td>R 1 set loads attached to this element: 'nr_load1, nr_load2,...' or empty</td>
</tr>
</tbody>
</table>

#### Graphics

| Graphics.RGB_color | vector  | [0.1, 0.1, 0.8] [red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.geom_elements | vector | [] Set Geometric elements to represent body 'geomelem1, geomelem2,...' or empty |
| Graphics.use_alternative_shape | bool   | 0 Graphical representation of element with geometrical objects that are attached to the element |
| Graphics.show_element | bool   | 1 Flag to draw element |

#### FiniteElement

| FiniteElement.Node_list | vector  | [1, 2, 3, 4, 5, 6, 7, 8] nodes of the element |
| FiniteElement.Body_index | int     | 1 index of the domain |
| FiniteElement.material_number | int     | 1 material number which contains the main material properties of the finite element |
| FiniteElement.GeometricNonlinearityStatus | int    | 1 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |

#### RotorDynamics

| RotorDynamics.angular_velocity | vector  | [0, 0, 0] Element is rotating with angular_velocity around axis_position. |
| RotorDynamics.axis_position   | vector  | [0, 0, 0] Element is rotating with angular_velocity around axis_position. |

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, [3.9.1](#). The sensor needs 2 informations: the field_variable itself and the component. For more information see section [3.1](#).

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude, x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_cauchy</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>total_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>logarithmic_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1](#).

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal_volume</td>
<td>volume of an element</td>
</tr>
</tbody>
</table>
3.2. ELEMENT

<table>
<thead>
<tr>
<th>Internal.potential_energy</th>
<th>potential (strain) energy of an element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element:
- PointJoint, 3.3.1
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- Rope3D, 3.3.7
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10
- SpringDamperActuator, 3.3.17
- RigidLink, 3.3.18

3.2.16 Tetrahedral

Short description

The Tetrahedral is a 3D element with 4 or 10 nodes, 4 faces with 3 nodes each. The local coordinates range $[0, 1]$ for $x$, $[0, 1 - x]$ for $y$ and $[0, 1 - x - y]$ for $z$.

<table>
<thead>
<tr>
<th>order 1 nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Number</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>local Coord</td>
<td>(0, 0, 0)</td>
<td>(1, 0, 0)</td>
<td>(0, 1, 0)</td>
<td>(0, 0, 1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>additional order 2 nodes</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Number</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>local Coord</td>
<td>(1/2, 0, 0)</td>
<td>(1/2, 1/2, 0)</td>
<td>(0, 1/2, 0)</td>
</tr>
</tbody>
</table>

| Node Number               | 8 | 9 | 10 |
| local Coord               | (1/2, 0, 1/2) | (0, 1/2, 1/2) | (0, 0, 1/2) |

Node numbers of the element faces - counter-clockwise from outside

<table>
<thead>
<tr>
<th>FaceNr</th>
<th>O1</th>
<th>O2</th>
<th>FaceCoordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.32</td>
<td>1.32, 2.76.5</td>
<td>$x' = y$, $y' = x$</td>
</tr>
<tr>
<td>2</td>
<td>1.2.4</td>
<td>1.2.4, 5.8.10</td>
<td>$x' = x$, $y' = z$</td>
</tr>
<tr>
<td>3</td>
<td>1.4.3</td>
<td>1.4.3, 10.9.7</td>
<td>$x' = x$, $y' = y$</td>
</tr>
<tr>
<td>4</td>
<td>2.3.4</td>
<td>2.3.4, 6.9.8</td>
<td>$x' = y$, $y' = z$</td>
</tr>
</tbody>
</table>

Degrees of freedom

3 degrees of freedoms per node (12 or 30), arranged by node, \( \{N1_x, N1_y, N1_z, N2_x, \ldots N10_z\} \)

Limitations

The Tetrahedral has only position DOFs
Figure 3.23: Order 1 Tetrahedral
3.2. ELEMENT

Figure 3.24: Order 2 Tetrahedral

Data objects of Tetrahedral:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Tetrahedral&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Element&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.1, 0.1, 0.8]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
</tbody>
</table>

FiniteElement

| FiniteElement.Node_list | vector | [1, 2, 3, 4] | nodes of the element            |
| FiniteElement.Body_index | integer | 1          | index of the domain            |
| FiniteElement.material_number | integer | 1          | material number which contains the main material properties of the finite element |
| FiniteElement.GeometricNonlinearityStatus | integer | 1          | 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |

RotorDynamics

| RotorDynamics.angular_velocity | vector | [0, 0, 0] | Element is rotating with angular_velocity around axis_position. |
CHAPTER 3. HOTINT REFERENCE MANUAL

<table>
<thead>
<tr>
<th>RotorDynamics.</th>
<th>axis_position</th>
<th>vector</th>
<th>[0, 0, 0]</th>
<th>Element is rotating with angular_velocity around axis_position.</th>
</tr>
</thead>
</table>

Observable FieldVariables:

The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude, x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_cauchy</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>total_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>logarithmic_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal_volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal_potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal_kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

Suitable Connectors:

The following connectors can be used to constrain the element:

PointJoint, 3.3.1, CoordinateConstraint, 3.3.2, VelocityCoordinateConstraint, 3.3.3, MultiCoordConstraint, 3.3.4, Rope3D, 3.3.7, FrictionConstraint, 3.3.8, Contact1D, 3.3.9, PlaneConstraint, 3.3.10, SpringDamperActuator, 3.3.17, RigidLink, 3.3.18

3.2.17 Prism

Short description

The Prism is a 3D element with 6 or 15 nodes, 5 faces total, 3 faces with 4 nodes and 2 faces (base and top) with 3 noxes.
The local coordinates ranges [0, 1] for x, [0, 1 − x] for y and [0, 1] for z.

order 1 nodes
3.2. ELEMENT

<table>
<thead>
<tr>
<th>Node Number</th>
<th>local Coord</th>
<th>Node Number</th>
<th>local Coord</th>
<th>Node Number</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0, 0, 0)</td>
<td>2</td>
<td>(1, 0, 0)</td>
<td>3</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>4</td>
<td>(0, 0, 1)</td>
<td>5</td>
<td>(1, 0, 1)</td>
<td>6</td>
<td>(0, 1, 1)</td>
</tr>
</tbody>
</table>

additional order 2 nodes

<table>
<thead>
<tr>
<th>Node Number</th>
<th>local Coord</th>
<th>Node Number</th>
<th>local Coord</th>
<th>Node Number</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>(1/2, 0, 0)</td>
<td>8</td>
<td>(1/2, 1, 0)</td>
<td>9</td>
<td>(0, 1/2, 0)</td>
</tr>
<tr>
<td>10</td>
<td>(0, 0, 1/2)</td>
<td>11</td>
<td>(1, 0, 1/2)</td>
<td>12</td>
<td>(0, 1, 1/2)</td>
</tr>
<tr>
<td>13</td>
<td>(1/2, 0, 1)</td>
<td>14</td>
<td>(1/2, 1, 1)</td>
<td>15</td>
<td>(0, 1/2, 1)</td>
</tr>
</tbody>
</table>

node numbers of the element faces - counter-clockwise from outside

<table>
<thead>
<tr>
<th>FaceNr</th>
<th>O1</th>
<th>O2</th>
<th>FaceCoordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 5, 4</td>
<td>1, 2, 5, 4, 7, 11, 12, 10</td>
<td>x’ = x, y’ = z</td>
</tr>
<tr>
<td>2</td>
<td>1, 4, 6, 3</td>
<td>1, 4, 6, 3, 10, 15, 12, 9</td>
<td>x’ = z, y’ = y</td>
</tr>
<tr>
<td>3</td>
<td>2, 3, 6, 5</td>
<td>2, 3, 6, 5, 8, 12, 14, 11</td>
<td>x’ = y, y’ = z</td>
</tr>
<tr>
<td>4</td>
<td>1, 3, 2</td>
<td>1, 3, 2, 9, 8, 7</td>
<td>x’ = y, y’ = x</td>
</tr>
<tr>
<td>5</td>
<td>4, 5, 6</td>
<td>4, 5, 6, 13, 14, 15</td>
<td>x’ = x, y’ = y</td>
</tr>
</tbody>
</table>

Degrees of freedom

3 degrees of freedoms per node (15 or 45), arranged by node, \{N_{1x}, N_{1y}, N_{1z}, N_{2x}, ... N_{15z}\}

Limitations

The Prism has only position DOFs
Figure 3.25: Order 1 Prism
3.2. ELEMENT

Figure 3.26: Order 2 Prism

Data objects of Prism:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;Prism&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Element&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td>[]</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.1, 0.1, 0.8]</td>
<td>red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[]</td>
<td>Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0</td>
<td>Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw element</td>
</tr>
</tbody>
</table>

FiniteElement

| FiniteElement.Node_list | vector | [1, 2, 3, 4] | nodes of the element                                                         |
| FiniteElement.Body_index | integer | 1 | index of the domain                                                          |
| FiniteElement.material_number | integer | 1 | material number which contains the main material properties of the finite element |
| FiniteElement.GeometricNonlinearityStatus | integer | 1 | 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |
**RotorDynamics**

<table>
<thead>
<tr>
<th>Field</th>
<th>Possible Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotorDynamics. angular_velocity</td>
<td>vector [0, 0, 0] Element is rotating with angular velocity around axis_position.</td>
</tr>
<tr>
<td>RotorDynamics. axis_position</td>
<td>vector [0, 0, 0] Element is rotating with angular velocity around axis_position.</td>
</tr>
</tbody>
</table>

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor, 3.9.1. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude, x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_cauchy</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>total_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>logarithmic_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value_name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.volume</td>
<td>volume of an element</td>
</tr>
<tr>
<td>Internal.potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal.kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element: PointJoint, 3.3.1, CoordinateConstraint, 3.3.2, VelocityCoordinateConstraint, 3.3.3, MultiCoordConstraint, 3.3.4, Rope3D, 3.3.7, FrictionConstraint, 3.3.8, Contact1D, 3.3.9, PlaneConstraint, 3.3.10, SpringDamperActuator, 3.3.17, RigidLink, 3.3.18

### 3.2.18 Pyramid

**Short description**

The Pyramid is a 3D element with 5 or 13 nodes, 5 faces total, one face (base) with 4 nodes and 4 faces with 3 nodes.

The local coordinates ranges \[0, 1\] for x and y and \[0, 1 - \text{max}(x, y)\] for z.
3.2. ELEMENT

order 1 nodes

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(1, 0, 0)</td>
</tr>
<tr>
<td>3</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>4</td>
<td>(1, 1, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(0, 0, 1)</td>
</tr>
</tbody>
</table>

additional order 2 nodes

<table>
<thead>
<tr>
<th>NodeNumber</th>
<th>local Coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(1/2, 0, 0)</td>
</tr>
<tr>
<td>7</td>
<td>(1/2, 1, 0)</td>
</tr>
<tr>
<td>8</td>
<td>(0, 1/2, 0)</td>
</tr>
<tr>
<td>9</td>
<td>(1, 1/2, 0)</td>
</tr>
<tr>
<td>10</td>
<td>(0, 0, 1/2)</td>
</tr>
<tr>
<td>11</td>
<td>(1/2, 0, 1/2)</td>
</tr>
<tr>
<td>12</td>
<td>(0, 1/2, 1/2)</td>
</tr>
<tr>
<td>13</td>
<td>(1, 1/2, 1/2)</td>
</tr>
</tbody>
</table>

node numbers of the element faces - counter-clockwise from outside

<table>
<thead>
<tr>
<th>FaceNr</th>
<th>O 1</th>
<th>O2</th>
<th>FaceCoordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,2,3</td>
<td>1,2,4,3, 6,9,7,8</td>
<td>x' = x, y' = y</td>
</tr>
<tr>
<td>2</td>
<td>1,2,5</td>
<td>1,3,5, 6,11,10</td>
<td>x' = x, y' = z</td>
</tr>
<tr>
<td>3</td>
<td>2,4,5</td>
<td>2,4,5, 9,13,11</td>
<td>x' = y, y' = z</td>
</tr>
<tr>
<td>4</td>
<td>3,5,4</td>
<td>3,5,4, 12,13,7</td>
<td>x' = z, y' = x</td>
</tr>
<tr>
<td>5</td>
<td>1,5,3</td>
<td>1,5,3, 10,12,8</td>
<td>x' = z, y' = y</td>
</tr>
</tbody>
</table>

Degrees of freedom

3 degrees of freedoms per node (15 or 45), arranged by node, \{N1_x, N1_y, N1_z, N2_x,...N15_z\}

Limitations

The Prism has only position DOFs
Data objects of Pyramid:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>

Figure 3.27: Order 1 Pyramid

Figure 3.28: Order 2 Pyramid
### 3.2. ELEMENT

<table>
<thead>
<tr>
<th>element_type</th>
<th>string</th>
<th>&quot;Pyramid&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Element&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 1 number of the element in the mbs</td>
</tr>
<tr>
<td>loads</td>
<td>vector</td>
<td>Set loads attached to this element: 'nr_load1, nr_load2, ...' or empty</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.RGB_color</th>
<th>vector</th>
<th>[0.1, 0.1, 0.8] red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.geom_elements</td>
<td>vector</td>
<td>[] Set Geometric elements to represent body 'geomelem1, geomelem2, ...' or empty</td>
</tr>
<tr>
<td>Graphics.use_alternative_shape</td>
<td>bool</td>
<td>0 Graphical representation of element with geom-objects that are attached to the element</td>
</tr>
<tr>
<td>Graphics.show_element</td>
<td>bool</td>
<td>1 Flag to draw element</td>
</tr>
</tbody>
</table>

**FiniteElement**

<table>
<thead>
<tr>
<th>FiniteElement.Node_list</th>
<th>vector</th>
<th>[1, 2, 3, 4] nodes of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiniteElement.Body_index</td>
<td>integer</td>
<td>1 index of the domain</td>
</tr>
<tr>
<td>FiniteElement.material_number</td>
<td>integer</td>
<td>1 material number which contains the main material properties of the finite element</td>
</tr>
<tr>
<td>FiniteElement.GeometricNonlinearityStatus</td>
<td>integer</td>
<td>1 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
</tbody>
</table>

**RotorDynamics**

<table>
<thead>
<tr>
<th>RotorDynamics.angular_velocity</th>
<th>vector</th>
<th>[0, 0, 0] Element is rotating with angular_velocity around axis_position.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotorDynamics.axis_position</td>
<td>vector</td>
<td>[0, 0, 0] Element is rotating with angular_velocity around axis_position.</td>
</tr>
</tbody>
</table>

**Observable FieldVariables:**

The following values can be measured with a FieldVariableElementSensor. The sensor needs 2 informations: the field_variable itself and the component. For more information see section 3.1

<table>
<thead>
<tr>
<th>field_variable</th>
<th>possible components</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>displacement</td>
<td>x, y, z, magnitude</td>
</tr>
<tr>
<td>velocity</td>
<td>x, y, z, magnitude, x, y, z, magnitude</td>
</tr>
<tr>
<td>stress</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_cauchy</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>stress_mises</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>total_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
<tr>
<td>logarithmic_strain</td>
<td>xx, xy, xz, yy, yz, zz, magnitude</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Internal volume</th>
<th>volume of an element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal_potential_energy</td>
<td>potential (strain) energy of an element</td>
</tr>
<tr>
<td>Internal_kinetic_energy</td>
<td>kinetic energy of an element</td>
</tr>
</tbody>
</table>

**Suitable Connectors:**

The following connectors can be used to constrain the element:

- PointJoint, 3.3.1
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- Rope3D, 3.3.7
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10
- SpringDamperActuator, 3.3.17
- RigidLink, 3.3.18
3.3 Connector

These connectors are available:

- PointJoint, 3.3.1
- CoordinateConstraint, 3.3.2
- VelocityCoordinateConstraint, 3.3.3
- MultiCoordConstraint, 3.3.4
- SlidingPointJoint, 3.3.5
- SlidingPrismaticJoint, 3.3.6
- Rope3D, 3.3.7
- FrictionConstraint, 3.3.8
- Contact1D, 3.3.9
- PlaneConstraint, 3.3.10
- GenericBodyJoint, 3.3.11
- RevoluteJoint, 3.3.12
- PrismaticJoint, 3.3.13
- UniversalJoint, 3.3.14
- RigidJoint, 3.3.15
- CylindricalJoint, 3.3.16
- SpringDamperActuator, 3.3.17
- RigidLink, 3.3.18
- RotatorySpringDamperActuator, 3.3.19
- SpringDamperActuator2D, 3.3.20
- PointJoint2D, 3.3.21

Note:
In HOTINT several classes are treated as 'elements'. Connectors and control elements are also 'elements', and can therefore be edited and deleted in the GUI with the menu items of the elements.
In the script language the command AddConnector has to be used for the connectors in the list above and also for control elements.
3.3.1 PointJoint

Short description

The PointJoint constrains two elements at a local position or node each. If only one element is specified (second element 0), a ground PointJoint is realized. It is possible to constrain just some of the directions. If the first body is a rigid body then the constraint forces are applied as follows:

**Connecting element to element:**
The constraint forces are applied on both bodies at the position of the connection point of the second body.

**Connecting element to ground:**
The constraint forces are applied on the body

- at the position of the connection point on ground if the formulation is penalty and use_local_coordinate_system=1 and
- at the position of the connection point of the element otherwise.

If the first element is a flexible body or a point mass the forces are applied differently. See *Limitations*.

Equations

**Lagrange formulation:**
The constraint equations are  
\[ C = A^T (x_1 - x_2) = 0 \]  
or on velocity level  \[ C = \dot{A}^T (x_1 - x_2) + A^T (v_1 - v_2) = 0 \]  
where each equation corresponds to a constrained direction.

The meaning of the variables is as follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>position of connection point on body 1 in global coordinates</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>position of connection point on body 2 in global coordinates, or if constraint connects element to ground then connection point of ground in global coordinates</td>
</tr>
<tr>
<td>( v_1 )</td>
<td>time derivative of ( x_1 )</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>time derivative of ( x_2 )</td>
</tr>
<tr>
<td>( A )</td>
<td>rotation matrix from local joint coordinates to global coordinates. If Geometry.use_local_coordinate_system = 1 and Geometry.use_joint_local_frame = 1, then ( A = QJ ). If Geometry.use_local_coordinate_system = 1 and Geometry.use_joint_local_frame = 0, then ( A = Q ). If Geometry.use_local_coordinate_system = 0 and Geometry.use_joint_local_frame = 1, then ( A = J ). If Geometry.use_local_coordinate_system = 0 and Geometry.use_joint_local_frame = 0, then ( A = I ).</td>
</tr>
<tr>
<td>( Q )</td>
<td>rotation matrix from local coordinate system of body 1 to global coordinates</td>
</tr>
<tr>
<td>( J )</td>
<td>joint local frame</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

**Penalty formulation:**
The spring force is given by

\[ f_s = A \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix} A^T \cdot (x_2 - x_1) \]

and the damper force by

\[ f_d = d \cdot (v_2 - v_1). \]

The resulting constraint force is then given by

\[ f = f_s + f_d. \]

Where

- \( f_s \) constraint force due to stiffness
- \( f_d \) constraint force due to damping,
- \( f \) constraint force
- \( k_x \) stiffness in (local or global) \( x \)-direction
- \( k_y \) stiffness in (local or global) \( y \)-direction
- \( k_z \) stiffness in (local or global) \( z \)-direction
- \( d \) damping coefficient

**Limitations**

In general the constraint forces act on the first body not at the position of the connection point of the first body.

If the first body is a rigid body, then the force acting on body 1 is shifted to the connection point of the first body. The moment induced by shifting the force is compensated by a moment in the opposite direction.

If the first body is a point mass, we cannot apply a force outside it’s position.

If the first body is a flexible body, applying a force outside the connection point is at least very questionable.

Hence the PointJoint has the following limitations:

- It is not possible to use the PointJoint in Lagrangian formulation if not all directions are constrained and the first body is a point mass or a flexible body.

- When using the PointJoint in penalty formulation with body 1 being a point joint or a flexible body, the constraint force acting on body 1 is applied at the position of the connection point of body 1. If the constraint forces are not collinear to connection points, e.g. if the stiffness is unisotropic or if a damping is set, then the shifting the force induces a moment. This moment is not compensated and thus the law of angular momentum is broken. Therefore a PointJoint in penalty formulation should only be used with a point masses or a flexible bodies being the first element, if you expect small displacements or the constraint forces to be (nearly) collinear to the connection points.

Another limitations, which applies for all kinds of bodies, is that it is not possible in Lagrange formulation to constraint just some global directions.
Description of the different modi
3.3. CONNECTOR

<table>
<thead>
<tr>
<th>element to ground</th>
<th>Position2.element_number AND Position2.node_number have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>element to element</td>
<td>Position2.element_number and/or Position2.node_number must not be equal to 0</td>
</tr>
<tr>
<td>Lagrange</td>
<td>If Physics.use_penalty_formulation = 0, then no stiffness and no damping parameters are used.</td>
</tr>
</tbody>
</table>

![Diagram of penalty forces acting](image)

Figure 3.29: The penalty forces acting on the first body (dotted) act on the position of $x_2$. The forces acting on the first body are shifted to $x_1$ (dashed) and a moment (dashed) is applied to compensate the induced moment.

### Data objects of PointJoint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;PointJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;PointJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 2</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>red, green, blue color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>-1</td>
<td>drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used.</td>
</tr>
<tr>
<td>Graphics.draw_size_joint_local_frame</td>
<td>double</td>
<td>0</td>
<td>drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn.</td>
</tr>
<tr>
<td>Geometry.use_joint_local_frame</td>
<td>bool</td>
<td>0</td>
<td>Use a special joint local frame</td>
</tr>
<tr>
<td>Geometry.joint_local_frame</td>
<td>matrix</td>
<td>[0, 0, 0; 0, 0, 0; 0, 0, 0]</td>
<td>Rotates the local or global coordinate system. Just used if use_joint_local_frame == 1</td>
</tr>
<tr>
<td>Geometry.use_local_coordinate_system</td>
<td>bool</td>
<td>0</td>
<td>0=use global coordinates, 1=use local coordinate system of Body 1</td>
</tr>
<tr>
<td>Physics</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Physics

<table>
<thead>
<tr>
<th>use_penalty_formulation</th>
<th>bool</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = use Lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Physics.Penalty

<table>
<thead>
<tr>
<th>spring_stiness</th>
<th>double</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>General or penalty stiffness parameter</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>spring_stiness_vector</th>
<th>vector</th>
<th>[0, 0, 0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>penalty stiffness parameter ([k_x, k_y, k_z]). Just used if scalar spring_stiness == 0, otherwise (k_x = k_y = k_z = \text{spring_stiness})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>damping</th>
<th>double</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damping coefficient for viscous damping ((F = dv)), applied in all constrained directions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Physics.Lagrange

<table>
<thead>
<tr>
<th>constrained_directions</th>
<th>vector</th>
<th>[1, 1, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x, y, z])...(1 = constrained, 0 = free), can be defined as local or global directions (see Geometry)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Position1

<table>
<thead>
<tr>
<th>element_number</th>
<th>integer</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of constrained element</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>position</th>
<th>vector</th>
<th>[0, 0, 0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local position. Only used if node_number == 0!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>node_number</th>
<th>integer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local or global (if element_number == 0) node number.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Position2

<table>
<thead>
<tr>
<th>element_number</th>
<th>integer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of constrained element</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>position</th>
<th>vector</th>
<th>[0, 0, 0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local or global (if element_number == 0) position. Only used if node_number == 0!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>node_number</th>
<th>integer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local or global (if element_number == 0) node number.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>Degrees of freedom (or generalized unknowns) of the element. Range: 1-3</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>Algebraic variables of the element. Range: 1-3</td>
</tr>
<tr>
<td>Connector.force</td>
<td>Force applied to the kinematic pairs due to the connector. Range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
</tbody>
</table>

### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness</td>
<td>Set the stiffness coefficient</td>
</tr>
<tr>
<td>Connector.damping</td>
<td>Set the damping coefficient</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

Example
see file PointJointShort.txt

\[ l = 1 \text{ m} \]
\[ g = 9.81 \text{ m/s}^2 \]

```
gravLoad
{
    load_type = "Gravity"
    direction = 3 \text{ z - direction}
    gravity_constant = g
}
nLoad = AddLoad(gravLoad)
```

```
rigidBody
{
    element_type = "Rigid3D"
    loads = [nLoad]
    Graphics.body_dimensions = [l, 0.05, 0.05]
}
nRigid = AddElement(rigidBody)
```

```
pointJoint
{
    element_type = "PointJoint"
    Position1
    {
        element_number = nRigid \text{ number of constrained element}
        position = [-l/2, 0, 0] \text{ local position}
    }
    Position2.position = [-l/2, 0, 0]
}
AddConnector(pointJoint)
```

3.3.2 CoordinateConstraint

Short description
The CoordinateConstraint constrains two elements by constraining a single coordinate of each element, e.g. the x-displacement of two different elements. If the second element number is zero, a groundjoint can be realized. The CoordinateConstraint uses the lagrange multiplier formulation by default, which means that there is no constraint violation at all. For static problems, the lagrange multiplier constraint formulation is applied directly, by adding the kinematical conditions to the nonlinear system equations. In dynamic (time dependent) simulations, the constraint is solved on the position (displacement) level with index 3 solvers and on the velocity level with index 2 solvers. Alternatively, the penalty formulation can be used, which means that a certain (very high) spring stiffness is used instead of lagrange multipliers. Thus, no additional equation is added, however, the system equations may become unsolvable stiff (ill conditioned) in case of static problems; for dynamical problems, the very high stiffness might lead to high-frequency oscillations, inaccurate solutions or no convergence.
Equations

Lagrange formulation:

position constraint (index 3 solver)
2 elements (coordinate to coordinate): \( C = k (q_{el1}^{i1} - q_{el2}^{i1}) - (q_{el1}^{j2} - q_{el2}^{j2}) - d = 0 \)
1 element (coordinate to ground): \( C = k (q_{el1}^{i1} - q_{el2}^{i1}) - d = 0 \)

velocity constraint - index reduction (index 2 solver)
2 elements (coordinate to coordinate): \( C = k \dot{q}_{el1}^{i1} - \dot{q}_{el2}^{j2} = 0 \)
1 element (coordinate to ground): \( C = \dot{q}_{el1}^{i1} = 0 \)

Langrange multiplier
\[
\frac{\partial C}{\partial q_{el1}^{i1}}^T = [0 \ldots 0, k, 0 \ldots 0] \ldots \text{with } k \text{ at index i}
\]
\[
\frac{\partial C}{\partial q_{el2}^{j2}}^T = [0 \ldots 0, -1, 0 \ldots 0] \ldots \text{with } -1 \text{ at index j}
\]

Penalty formulation:

2 elements (coordinate to coordinate): \( f = SP \left(k (q_{el1}^{i1} - q_{el2}^{i1}) - (q_{el1}^{j2} - q_{el2}^{j2}) - d\right) + DP \left(k \dot{q}_{el1}^{i1} - \dot{q}_{el2}^{j2}\right) \)
1 element (coordinate to ground): \( f = SP \left(k (q_{el1}^{i1} - q_{el2}^{i1}) - d\right) + DP k \dot{q}_{el1}^{i1} \)

Description:
\( k \) ... coordinate gain factor
\( d \) ... coordinate offset (for index 2 solvers not used)
\( q_{el1}^{i1} \) ... \( i^{th} \) coordinate of element 1
\( q_{el1}^{i1} (t = 0) \ldots \) \( i^{th} \) coordinate of element 1 at initialization
\( q_{el2}^{j2} \) ... \( j^{th} \) coordinate of element 2
\( q_{el2}^{j2} (t = 0) \ldots \) \( j^{th} \) coordinate of element 2 at initialization
\( SP \) ... spring stiffness
\( DP \) ... damping
\( C \) ... Lagrange equation
\( f \) ... force vector (penalty formulation)

Description of the different modi

| coordinate to ground | Coordinate2.element_number AND Coordinate2.local_coordinate have to be equal to 0 |
| coordinate to coordinate | Coordinate2.element_number AND/or Coordinate2.local_coordinate must not be equal to 0 |
| Lagrange | For Physics.use_penalty_formulation = 0 no stiffness parameter is used. |
| relative or absolute to initial values | Only important for max index 3 solvers. If relative_to_initial_values is set to 1: Equation above is used. If set to 0: Simplified equation is used (\( q_{el1}^{i1} = q_{el2}^{j2} = 0 \)). |

Data objects of CoordinateConstraint:
### 3.3. CONNECTOR

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;CoordinateConstraint&quot;</td>
<td>specification of element type. Once the element is added to the nbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;CoordinateConstraint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the nbs</td>
</tr>
</tbody>
</table>

#### Graphics

| Graphics:RGB_color         | vector       | [0.3, 0.8, 0.3] | [red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.show_connector    | bool         | 1             | Flag to draw connector                                                  |
| Graphics.draw_size         | double       | 0.1           | General drawing size of constraint                                       |

#### Physics

| Physics.use_penalty_formulation | bool | 0 | 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |

- **Physics.Penalty**
  - damping: double 0 damping coefficient $D_p$ for viscous damping
  - spring_stiffness: double 0 general or penalty stiffness parameter $S_p$

- **coord_offset**: double 0 coordinate offset $d$, see documentation section equation
- **coord_gain_factor**: double 1 coordinate gain factor $k$, see documentation section equation
- **relative_to_initial_values**: bool 1 flag $== 1$: full equation is used, see documentation; flag $== 0$: the init state values $q_i0$ and $q_j0$ are neglected.

- **Coordinate1**
  - element_number: integer 0 element number for coordinate 1
  - local_coordinate: integer 0 Local coordinate of element 1 to be constrained

- **Coordinate2**
  - element_number: integer 0 element number for coordinate 2; for ground joint, set element number to zero
  - local_coordinate: integer 0 Local coordinate of element 2 to be constrained

#### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint generalized_force</td>
<td>force acting on the generalized coordinates</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint coordinate_difference</td>
<td>difference between the coordinates</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint coordinate_offset</td>
<td>coordinate offset for CoordinateConstraint (w.r.t. ground or between two element coordinates); offset is ignored for Index 2 (setting of time integration) velocity level constraint</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint gain_factor</td>
<td>coordinate gain factor for CoordinateConstraint</td>
</tr>
</tbody>
</table>
Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.CoordinateConstraint.coordinate_offset</td>
<td>coordinate offset for CoordinateConstraint (w.r.t. ground or between two element coordinates); offset is ignored for Index 2 (setting of time integration) velocity level constraint</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.gain_factor</td>
<td>coordinate gain factor for CoordinateConstraint</td>
</tr>
</tbody>
</table>

Example

see file CoordinateConstraint.txt

l = 1 % m

rigidBody
{
    element_type= "Rigid3D"
    Graphics.body_dimensions= [1, 0.05, 0.05]
}
nRigid = AddElement(rigidBody)

coordinateConstraint
{
    element_type= "CoordinateConstraint"
    Coordinate1
    {
        element_number= nRigid %element number for coordinate 1
        local_coordinate= 1 %local coordinate of element 1
    }
}
AddConnector(coordinateConstraint)

3.3.3 VelocityCoordinateConstraint

Short description

Similar to CoordinateConstraint. Lagrangian constraint implemented for index 3 and index 2 solvers. A penalty formulation is also implemented.

Equations

Lagrange formulation:

velocity constraint (index 2 and 3 solvers)
2 elements (coordinate to coordinate): \( C = k (\dot{q}_{1}^{el1} - \dot{q}_{1,0}^{el1}) - (\dot{q}_{2,0}^{el2} - \dot{q}_{2}^{el2}) - d = 0 \),
1 element (coordinate to ground): \( C = k (\dot{q}_{1}^{el1} - \dot{q}_{1,0}^{el1}) - d = 0 \)
3.3. CONNECTOR

Langrange multiplier

\[
\begin{align*}
\frac{\partial C}{\partial \dot{q}^i} & = [0 \ldots 0, k, 0 \ldots 0] \ldots \text{with } k \text{ at index } i \\
\frac{\partial C}{\partial \dot{q}^j} & = [0 \ldots 0, -1, 0 \ldots 0] \ldots \text{with } -1 \text{ at index } j
\end{align*}
\]

Penalty formulation:

2 elements (coordinate to coordinate): \( f = S_P \left( k (\dot{q}^{el1}_i - \dot{q}^{el1}_{i,0}) - (\dot{q}^{el2}_j - \dot{q}^{el2}_{j,0}) - d \right) \)

1 element (coordinate to ground): \( f = S_P \left( k (\dot{q}^{el1}_i - \dot{q}^{el1}_{i,0}) - d \right) \)

Description:

\( k \) ... coordinate velocity gain factor

\( d \) ... coordinate velocity offset

\( \dot{q}^{el1}_i \) ... \( i^{th} \) coordinate velocity of element 1

\( \dot{q}^{el1}_{i,0} = \dot{q}^{el1}_i (t = 0) \) ... \( i^{th} \) coordinate velocity of element 1 at initialization

\( \dot{q}^{el2}_j \) ... \( j^{th} \) coordinate velocity of element 2

\( \dot{q}^{el2}_{j,0} = \dot{q}^{el2}_j (t = 0) \) ... \( j^{th} \) coordinate velocity of element 2 at initialization

\( S_P \) ... spring stiffness

\( C \) ... Lagrange equation

\( f \) ... force vector (penalty formulation)

Description of the different modi

<table>
<thead>
<tr>
<th>coordinate to ground</th>
<th>Coordinate2.element_number AND Coordinate2.local_coordinate have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordinate to coordinate</td>
<td>Coordinate2.element_number and/or Coordinate2.local_coordinate must not be equal to 0</td>
</tr>
</tbody>
</table>
| relative or absolute to initial values | If relative_to_initial_values is set to 1: Equation above is used. 
If set to 0: Simplified equation is used \((\dot{q}^{el1}_i = \dot{q}^{el2}_{j,0} = 0)\). |

Data objects of VelocityCoordinateConstraint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;VelocityCoordinateConstraint&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;VelocityCoordinateConstraint&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics.RGB_color    | vector       | [0.3, 0.8, 0.3] | [red, green, blue] color of element, range = 0.1, use default color: [-1, -1, -1] |
| Graphics.show_connector | bool        | 1              | Flag to draw connector                          |
| Graphics.draw_size    | double       | 0.1            | General drawing size of constraint              |

**Physics**

| Physics.use_penalty_formulation | bool | 0 | 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |
### Physics Penalty

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spring_stiffness</td>
<td>double</td>
<td>0</td>
<td>general or penalty stiffness parameter Sp</td>
</tr>
<tr>
<td>coord_offset</td>
<td>double</td>
<td>0</td>
<td>coordinate offset d, see documentation section equation</td>
</tr>
<tr>
<td>coord_gain_factor</td>
<td>double</td>
<td>1</td>
<td>coordinate gain factor k, see documentation section equation</td>
</tr>
<tr>
<td>relative_to_initial_values</td>
<td>bool</td>
<td>1</td>
<td>flag == 1: full equation is used, see documentation; flag == 0: the init state derivatives (d(q_i)/dt) and (d(q_j)/dt) are neglected.</td>
</tr>
</tbody>
</table>

### Coordinate 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>0</td>
<td>element number for coordinate 1</td>
</tr>
<tr>
<td>local_coordinate</td>
<td>integer</td>
<td>0</td>
<td>Local coordinate of element 1 to be constrained</td>
</tr>
</tbody>
</table>

### Coordinate 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>0</td>
<td>element number for coordinate 2; for ground joint, set element number to zero</td>
</tr>
<tr>
<td>local_coordinate</td>
<td>integer</td>
<td>0</td>
<td>Local coordinate of element 2 to be constrained</td>
</tr>
</tbody>
</table>

#### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.generalized_force</td>
<td>force acting on the generalized coordinates</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.coordinate_difference</td>
<td>difference between the coordinates</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.coordinate_offset</td>
<td>coordinate offset for CoordinateConstraint (w.r.t. ground or between two element coordinates); offset is ignored for Index 2 (setting of time integration) velocity level constraint</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.gain_factor</td>
<td>coordinate gain factor for CoordinateConstraint</td>
</tr>
</tbody>
</table>

#### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.CoordinateConstraint.coordinate_offset</td>
<td>coordinate offset for CoordinateConstraint (w.r.t. ground or between two element coordinates); offset is ignored for Index 2 (setting of time integration) velocity level constraint</td>
</tr>
<tr>
<td>Connector.CoordinateConstraint.gain_factor</td>
<td>coordinate gain factor for CoordinateConstraint</td>
</tr>
</tbody>
</table>
Example

see file VelocityCoordinateConstraint.txt

1 = 1 % m

rigidBody
{
  element_type= "Rigid3D"
  Graphics.body_dimensions= [1, 0.05, 0.05]
}
nRigid = AddElement(rigidBody)

cordinateConstraint
{
  element_type= "VelocityCoordinateConstraint"
  Coordinate1
  {
    element_number= nRigid %element number for coordinate 1
    local_coordinate= 1 %local coordinate of element 1
  }
}
AddConnector(cordinateConstraint)

3.3.4 MultiCoordConstraint

Short description

The MultiCoordConstraint is an extension of Co ordinateConstraint and constrains more than
two elements. Only the lagrange multiplier formulation is implemented and no penalty formu-
lation.

Equations

position constraint (index 3 solver)
\( C = k_1 (q^{el\_1} - q^{el\_1,0}) - k_2 (q^{el\_2} - q^{el\_2,0}) - k_3... - d = 0 \)

velocity constraint - index reduction (index 2 solver)
\( C = k_1 \dot{q}^{el\_1} - k_2 \dot{q}^{el\_2} - k_3... = 0 \)

Langrange multiplier

first element:
\( \frac{\partial C}{\partial q_{el\_1}}^T = [0 ... 0, k_1, 0 ... 0] \) ... with \( k_1 \) at index \( c_1 \)

i-th element:
\( \frac{\partial C}{\partial q_{el\_i}}^T = [0 ... 0, -k_i, 0 ... 0] \) ... with \( -k_i \) at index \( c_i \)

Description:

\( k_1, k_2, ..., k_i \) ... coordinate gain factors

\( d \) ... coordinate offset (for index 2 solvers not used)

\( q^{el\_j} \) ... \( c_i \) coordinate of element \( j \)

\( q^{el\_j,0} = q^{el\_j} (t = 0) \) ... \( c_i \) coordinate of element \( j \) at initialization

\( C \) ... Lagrange equation
Description of the different modi

| relative or absolute to initial values | Only important for max index 3 solvers. If relative_to_initial_values is set to 1: Equation above is used. If set to 0: Simplified equation is used \( q_{i0}^1 = q_{20}^2 = \ldots = 0 \). |

Figure 3.30: 3 point masses are constrained to each other with MultiCoordConstraints to obtain the same behaviour as a rigid body, see the provided example code.

Data objects of MultiCoordConstraint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;MultiCoordConstraint&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;MultiCoordConstraint&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2 number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.RGB_color</th>
<th>vector</th>
<th>[0.3, 0.8, 0.3]</th>
<th>red, green, blue color of element, range = 0..1, use default color: [-1,-1,-1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>0.1</td>
<td>General drawing size of constraint</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td></td>
<td>element numbers to constrain</td>
</tr>
<tr>
<td>local_coordinates</td>
<td>vector</td>
<td></td>
<td>local coordinates of elements to be constrained</td>
</tr>
<tr>
<td>coord_gain_factors</td>
<td>vector</td>
<td></td>
<td>coordinate gain factor k for each element, see documentation section equation</td>
</tr>
<tr>
<td>coord_offset</td>
<td>double</td>
<td>0</td>
<td>coordinate offset d, see documentation section equation</td>
</tr>
<tr>
<td>relative_to_initial_values</td>
<td>bool</td>
<td>1</td>
<td>flag == 1: full equation is used, see documentation; flag == 0: the init state values q10 and q10 are neglected</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
</tbody>
</table>

Example
see file MultiCoordinateConstraint.txt

\[
\text{r} = 0.1 \quad \% \text{[m]} \text{ distance between point masses} \\
\text{m\_center} = 2 \quad \% \text{[kg]} \text{ mass of point mass in the center} \\
\text{m\_outer} = 0.5 \quad \% \text{[kg]} \text{ mass of point mass at the outer edges}
\]

\[
\begin{align*}
\text{Force.load\_type} &= \text{"ForceVector2D"} \\
\text{Force.force\_vector} &= [0,1] \\
\text{Force.position} &= [0,0] \\
\text{nLoad} &= \text{AddLoad(Force)} \\
\text{PointMass.element\_type} &= \text{"Mass2D"} \\
\text{PointMass.loads} &= [] \quad \% \text{no load at left mass} \\
\text{PointMass.Graphics.radius} &= \text{r/10} \\
\text{PointMass.Initialization.initial\_position} &= [0,0] \\
\text{PointMass.Physics.mass} &= \text{m\_outer} \\
\text{nE\_mLeft} &= \text{AddElement(PointMass)} \\
\text{PointMass.loads} &= \{\text{nLoad}\} \quad \% \text{force vector acting on right mass} \\
\text{PointMass.Initialization.initial\_position} &= [2*r,0] \\
\text{nE\_mRight} &= \text{AddElement(PointMass)} \\
\text{PointMass.loads} &= [] \quad \% \text{no load at the center mass} \\
\text{PointMass.Initialization.initial\_position} &= [r,0] \\
\text{PointMass.Physics.mass} &= \text{m\_center} \\
\text{nE\_mCenter} &= \text{AddElement(PointMass)} \\
\text{MultiCC} \\
\{ \\
\text{element\_type} &= \text{"MultiCoordConstraint"} \\
\text{Graphics.draw\_size} &= \text{r/10} \\
\text{element\_numbers} &= \{\text{nE\_mCenter},\text{nE\_mLeft},\text{nE\_mRight}\} \\
\text{local\_coordinates} &= [1,1,1] \quad \% \text{constrain x-directions} \\
\text{coord\_gain\_factors} &= [1,0.5,0.5] \quad \% \text{x of center mass is average of outer masses} \\
\} \\
\text{AddConnector(MultiCC)} \\
\text{MultiCC.local\_coordinates} &= [2,2,2] \quad \% \text{constrain y-directions} \\
\text{AddConnector(MultiCC)}
\]
ConstLength
{
  element_type= "SpringDamperActuator2D"
  Graphics.show_connector = 0
  Physics.spring_length= r  % keep the distance constant
  Physics.Linear.spring_stiffness= 1000  % high stiffness
  Physics.Linear.damping= 10  % sufficient damping
  Position1.element_number= nE_mCenter  % center - left
  Position2.element_number= nE_mLeft
}

AddConnector(ConstLength)

ConstLength.Position2.element_number= nE_mRight  % center - right
AddConnector(ConstLength)

% compare with rigid body formulation (reference solution) ================
Force.name= "Load for Rigid"
Force.position= [r, 0]  % local position of load is different
nLoadRigid = AddLoad(Force)

RigidBody.element_type= "Rigid2D"
RigidBody.loads= [nLoadRigid]
RigidBody.Graphics.body_dimensions= [2*r, r/10, r/10]
RigidBody.Graphics.RGB_color = [0.1,0.8,0.8]
RigidBody.Physics.moment_of_inertia= 2*m_outer*r*r  % [I_ZZ]
RigidBody.Physics.mass= 2*m_outer+m_center  % total mass of the body in kg
RigidBody.Initialization.initial_position= [r, 0]
AddElement(RigidBody)

3.3.5 SlidingPointJoint

Short description

This joint enables sliding of a fixed point of a body i along the x-axis of another body j. Both body i and body j can be flexible or rigid. Body j can contain more than one elements. No rotations are constrained at all. Only a Lagrangian formulation is implemented, the penalty formulation is not implemented yet. A MaxIndex 2 and 3 formulation exists.

Degrees of freedom

The vector of the DOF contains the sliding parameter s, its time derivative \( \dot{s} \) and the vector of the Lagrangian parameters \( \lambda = [\lambda_1 \lambda_2 \lambda_3]^T \). The Lagrange parameters \( \lambda_1 \) to \( \lambda_3 \) are representing the sliding forces in the global coordinate system.

\[
q = \begin{bmatrix} s & \dot{s} & \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}^T
\]  

(3.15)

Equations

positions:

\[
x^i = \begin{bmatrix} x_1^i & x_2^i & x_3^i \end{bmatrix}^T
\]  

(3.16)
3.3. CONNECTOR

\[
x^j = \begin{bmatrix} x_1^j = s & x_2^j & x_3^j \end{bmatrix}^T
\]

(3.17)

constraint equation - position level

\[
C = \begin{bmatrix} r^i(x^i) - r^j(x^j) \\
\frac{\partial r^i(x^i)}{\partial x_1^j} \lambda \\
\frac{\partial r^j(x^j)}{\partial x_1^j} \lambda \\
\frac{\partial r^j(x^j)}{\partial x_1^j} \lambda \end{bmatrix} = 0
\]

(3.18)

The first three constraints restrict the motion of the sliding point on body \(i\) and \(j\). The fourth constraint equation ensures, that there is no force in the sliding direction.

constraint equation - velocity level:

\[
C = \begin{bmatrix} \frac{\partial r^i(x^i)}{\partial t} - \frac{r^j(x^j)}{\partial t} - \frac{r^j(x^j)}{\partial x_1^j} \dot{s} \\
\frac{\partial r^i(x^i)}{\partial x_1^j} \lambda \\
\frac{\partial r^j(x^j)}{\partial x_1^j} \lambda \\
\frac{\partial r^j(x^j)}{\partial x_1^j} \lambda \end{bmatrix} = 0
\]

(3.19)

To obtain the constraints for velocity level, the first three equations are differentiated with respect to time. The sliding parameter \(s\) is also a function of time. The fourth constraint equation is equal to the position level equation.

Description of the different modi

<table>
<thead>
<tr>
<th>Sliding along a single body</th>
<th>The vector Geometry.element_numbers is equal to ([en1, en2]). Index Geometry.elemin1 must be 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sliding along more than one body</td>
<td>Geometry.element_numbers has to be set to ([en1, en2_1, en2_2, ..., en2_n]). Geometry.elemin1 is the body (j) index of the element in initial configuration, e.g. for (en2_2) the elemin1 is 2.</td>
</tr>
</tbody>
</table>

Figure 3.31: SlidingPointJoint

Data objects of SlidingPointJoint:
### CHAPTER 3. HOTINT REFERENCE MANUAL

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;SlidingPointJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;SlidingPointJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th></th>
<th>type</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td></td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>-1</td>
<td></td>
<td>Drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used.</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th></th>
<th>type</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.elemin</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>Index of the initial sliding body.</td>
</tr>
<tr>
<td>Geometry.position_1</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Vector from the center of body number 1 (en1) to the sliding point in the local body 1 coordinate system.</td>
</tr>
<tr>
<td>Geometry.position_2</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Vector from the center of the first body of en2 array to the sliding point in the local body 2 coordinate system.</td>
</tr>
<tr>
<td>Geometry.element_numbers</td>
<td>vector</td>
<td></td>
<td>[1, 2]</td>
<td>Element numbers: [en1,en2_1,en2_2,...,en2_n].</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-5</td>
</tr>
<tr>
<td>Internal.first_order_variable</td>
<td>first order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-4</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3 corresponds to force in global x-y-z direction</td>
</tr>
</tbody>
</table>

### Example

see file SlidingPointJoint.txt

```
l = 1 %m
k = 0.02 %nominal value k=1; decreased stiffness for demonstration!

load % define the load
{
   load_type = "ForceVector3D"
   position = [1/2,0,0]
   force_vector = [0,0,1000] % magnitude and direction
}
```
nLoad=AddLoad(load)

material
{
  material_type = "Beam3DProperties"
  cross_section_type = 1 % rectangular cross section
  cross_section_size = [0.05,0.1]
  density = 7850 \( \text{kg/m}^3 \)
  EA = 2100000000\( \text{k N} \)
  EIy = 1750000\( \text{k Nm}^2 \)
  E Iz = 1750000\( \text{k Nm}^2 \)
  GAky = 800000000\( \text{k N} \)
  GAKz = 800000000\( \text{k N} \)
  GJkx = 500000000\( \text{k Nm}^2 \)
  RhoA = 78.5 \( \text{kg/m}^2 \)
  RhoIx = 0.1 \( \text{kgm} \)
  RhoIy = 0.1 \( \text{kgm} \)
  RhoIz = 0.1 \( \text{kgm} \)
}

nMaterial = AddBeamProperties(material)

node
{
  node_type = "Node3DRxyz"
}

n1 = AddNode(node)

node.Geometry.reference_position = [l/2,0,0]

n2 = AddNode(node)

node.Geometry.reference_position = [l,0,0]

n3 = AddNode(node)

node.Geometry.reference_position = [3*l/4,0,0]
node.Geometry.reference_rot_angles = [0,-\( \text{Pi}/2,0 \)] % bryant angles

n4 = AddNode(node)

node.Geometry.reference_position = [3*l/4,0,l]

n5 = AddNode(node)

beam
{
  element_type= "LinearBeam3D"
  Physics.material_number = nMaterial
  Geometry.node_1 = n1
  Geometry.node_2 = n2
}

nBeam12 = AddElement(beam)

beam.Geometry.node_1 = n2
beam.Geometry.node_2 = n3
nBeam23 = AddElement(beam)
beam.loads = [nLoad]
beam.Geometry.node_1 = n4
beam.Geometry.node_2 = n5
nBeam45 = AddElement(beam)

slidingJoint
{
  element_type = "SlidingPointJoint"
  Geometry
  {
    elemind = 2  %number of the initial sliding body (2nd body).
    position_1 = [-1/2, 0, 0]
    %vector from the center of body number 1 (en1) to the sliding point
    %in the local body 1 coordinate system.
    position_2 = [1/2, 0, 0]  %vector from the center of the first body
    %of en2 array to the sliding point in the local body 2 coordinate system.
    element_numbers = [nBeam45, nBeam12, nBeam23]
    %Element numbers: [en1, en2_1, en2_2].
  }
}
AddConnector(slidingJoint)

rigidJoint
{
  element_type = "RigidJoint"
  Position1
  {
    element_number = nBeam12  %constrained element
    position = [-1/4, 0, 0]  %local position.
  }
}
AddConnector(rigidJoint)

3.3.6 SlidingPrismaticJoint

Short description

This joint enables sliding of a fixed point of a body i along the x-axis of another body j. Both body i and body j can be flexible or rigid. Body j can contain more than one element. The difference to the SlidingPointJoint is that the relative rotation between the bodies is also constrained. A Lagrangian formulation is used for both stiff and springy constrained rotation. For the position constraint only a stiff formulation exists. A penalty formulation is not implemented yet. There is a MaxIndex 2 and 3 formulation implemented.

Degrees of freedom

The vector of the DOF contains the sliding parameter $s$, its time derivative $\dot{s}$ and the vector of the Lagrangian parameters $\lambda = [\lambda_1 \lambda_2 \lambda_3]^T$. The Lagrange parameters $\lambda_1$ to $\lambda_3$ are representing the sliding forces in the global coordinate system. The three Lagrangian parameters $\lambda_4$ to $\lambda_6$ are the sliding moments about the global coordinate system axes.

$$q = \begin{bmatrix} s & \dot{s} & \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \end{bmatrix}^T$$ (3.20)
3.3. CONNECTOR

Equations

At initialization the unit vectors of the global coordinate system are transformed to the local coordinate system of each body and the vectors \( v_i^1, v_i^2 \) and \( v_i^3 \) for body \( i \) and \( v_j^1, v_j^2 \) and \( v_j^3 \) for body \( j \) are obtained. The vectors are fixed in the body coordinate system. The position vectors are the same as for the SlidingPointJoint.

**constraint equation - position level (stiff connection)**

\[
C = \begin{bmatrix}
    r^i(x^i) - r^j(x^j) \\
    \frac{\partial r^j(x^j)}{\partial x_1^i} \lambda \\
    v_j^2 v_j^3 \\
    v_j^3 v_i^1 \\
    v_j^3 v_i^1
\end{bmatrix} = 0 \quad (3.21)
\]

**constraint equation - position level (springy connection)**

\[
C = \begin{bmatrix}
    r^i(x^i) - r^j(x^j) \\
    \frac{\partial r^j(x^j)}{\partial x_1^i} \lambda \\
    v_j^2 v_j^3 k_1 + (\dot{v}_j^2 v_j^3 + v_j^2 \dot{v}_j^3) d_1 + \lambda_4 \\
    v_j^3 v_i^1 k_1 + (\dot{v}_j^3 v_i^1 + v_j^3 \dot{v}_i^1) d_2 + \lambda_5 \\
    -v_j^2 v_i^1 k_1 - (\dot{v}_j^2 v_i^1 + v_j^2 \dot{v}_i^1) d_3 + \lambda_6
\end{bmatrix} = 0 , \quad (3.22)
\]

**constraint equation - velocity level (stiff connection)**

\[
C = \begin{bmatrix}
    \frac{\partial r^i(x^i)}{\partial t} - \frac{r^i(x^i)}{\partial t} - \frac{r^j(x^j)}{\partial x_1^i} \dot{s} \\
    \frac{\partial r^j(x^j)}{\partial x_1^i} \lambda \\
    \dot{v}_2 v_3 v_1 + v_2 \dot{v}_3 \\
    \dot{v}_3 v_1 + v_3 \dot{v}_1 \\
    \dot{v}_2 v_1 + v_2 \dot{v}_1
\end{bmatrix} = 0 \quad (3.23)
\]

**constraint equation - velocity level (springy connection)**

\[
C = \begin{bmatrix}
    \frac{\partial r^i(x^i)}{\partial t} - \frac{r^i(x^i)}{\partial t} - \frac{r^j(x^j)}{\partial x_1^i} \dot{s} \\
    \frac{\partial r^j(x^j)}{\partial x_1^i} \lambda \\
    v_j^2 v_j^3 k_1 + (\dot{v}_j^2 v_j^3 + v_j^2 \dot{v}_j^3) d_1 + \lambda_4 \\
    v_j^3 v_i^1 k_1 + (\dot{v}_j^3 v_i^1 + v_j^3 \dot{v}_i^1) d_2 + \lambda_5 \\
    -v_j^2 v_i^1 k_1 - (\dot{v}_j^2 v_i^1 + v_j^2 \dot{v}_i^1) d_3 + \lambda_6
\end{bmatrix} = 0 \quad (3.24)
\]

Description of the different modi
sliding along a single body

The vector Geometry.element_numbers is equal to $[en_1, en_2]$. Index Geometry.elemind must be 1.

sliding along more than one body

Geometry.element_numbers has to be set to $[en_1, en_{2_1}, en_{2_2}, ..., en_{2_n}]$. Geometry.elemind is the body $j$ index of the element in initial configuration, e.g. for $en_{2_2}$ the elemind is 2.

stiff constrained rotation

Physics.use_penalty_formulation is set to 0.

springy constrained rotation

Physics.use_penalty_formulation is set to 1. The values for stiffness and damping must be set in Physics.Penalty folder.

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;SlidingPrismaticJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;SlidingPrismaticJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.3, 0.8, 0.3]</td>
<td>red, green, blue color of element, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.drawing_size</td>
<td>double</td>
<td></td>
<td>-1</td>
<td>Drawing dimensions of constraint. If set to -1, then global_drawing_scalar_size is used.</td>
</tr>
<tr>
<td>Geometry.position_1</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Vector from the center of body number 1 (en1) to the sliding point in the local body 1 coordinate system.</td>
</tr>
<tr>
<td>Geometry.position_2</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Vector from the center of the first body of en2 array to the sliding point in the local body 2 coordinate system.</td>
</tr>
<tr>
<td>Geometry.element_numbers</td>
<td>vector</td>
<td></td>
<td>[1, 2]</td>
<td>Element numbers: [en1, en2_1, en2_2, ..., en2_n].</td>
</tr>
<tr>
<td>Geometry.elemind</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>Index of the initial sliding body.</td>
</tr>
<tr>
<td>Physics.use_penalty_formulation</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physics.Penalty</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Penalty.k1</td>
<td>double</td>
<td></td>
<td>1e+005</td>
<td>Stiffness for rotation about global x - axis.</td>
</tr>
<tr>
<td>Physics.Penalty.k2</td>
<td>double</td>
<td></td>
<td>1e+005</td>
<td>Stiffness for rotation about global y - axis.</td>
</tr>
<tr>
<td>Physics.Penalty.k3</td>
<td>double</td>
<td></td>
<td>1e+005</td>
<td>Stiffness for rotation about global z - axis.</td>
</tr>
<tr>
<td>Physics.Penalty.d1</td>
<td>double</td>
<td></td>
<td>100</td>
<td>Damping of rotation about global x - axis.</td>
</tr>
<tr>
<td>Physics.Penalty.d2</td>
<td>double</td>
<td></td>
<td>100</td>
<td>Damping of rotation about global y - axis.</td>
</tr>
<tr>
<td>Physics.Penalty.d3</td>
<td>double</td>
<td></td>
<td>100</td>
<td>Damping of rotation about global z - axis.</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section 3.1
3.3. CONNECTOR

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-8</td>
</tr>
<tr>
<td>Internal.first_order_variable</td>
<td>first order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td>Connector.SlidingPrismaticJoint.sliding_parameter</td>
<td>internal sliding parameter s</td>
</tr>
<tr>
<td>Connector.SlidingPrismaticJoint.sliding_parameter_p</td>
<td>internal time derivative of sliding parameter s</td>
</tr>
</tbody>
</table>

Example

see file SlidingPrismaticJoint.txt

... copy this part from "SlidingPointJoint" example

slidingJoint
{
  element_type = "SlidingPrismaticJoint"
  Geometry
  {
    elemind = 2 %number of the initial sliding body (2nd body).
    position_1 = [-1/2, 0, 0] %vector from the center of body number 1 (en1) to the sliding point
    %in the local body 1 coordinate system.
    position_2 = [1/2, 0, 0] %vector from the center of the first body
    %of en2 array to the sliding point in the local body 2 coordinate system.
    element_numbers = [nBeam45, nBeam12,nBeam23] %Element numbers: [en1, en2_1, en2_2].
  }
}
AddConnector(slidingJoint)

... copy this part from "SlidingPointJoint" example

3.3.7 Rope3D

Short description

Elastic rope that is always under tension and can be fixed to multiple bodies and ground. There are 2 different kinds of suspensions points. Suspension points fixed on the ground are defined with the element number 0 and the global position. Suspension points on bodies are defined with the element number and the corresponding local position.

Limitations

The rope is assumed to be straight between 2 suspension points. No negative forces can be transmitted by a rope. The computation of the time derivative of the length of the rope is just an approximation. Therefore the damping of the rope may be represented slightly incorrect.
Data objects of Rope3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Rope3D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Rope3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics.RGB_color | vector  | [0.3, 0.8, 0.3] | red, green, blue color of element, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.show_connector | bool   | 1           | Flag to draw connector                                                      |
| Graphics.draw_size | double  | -1          | drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used. |

**Geometry**

| Geometry.rope_length | double  | R | 1 | initial length l0 of rope (computed automatically) |
| Geometry.element_numbers | vector  | [0, 0] | element numbers of the suspension points |
| Geometry.positions   | matrix   | [0, 0, 0; 1, 0, 0] | (local) positions of the suspension points |

**Physics**

| Physics.Penalty.damping | double  | 0 | damping coefficient for viscous damping (F = d*v), applied in all constrained directions |
| Physics.Penalty.rope_stiness | double  | 0 | [N] stiffness parameter c of the rope, F = c * (l-l0) / l0 |
| Physics.Penalty.spring_stiness | double  | R | 0 | total stiffness c1 of the rope F = c1 * (l-l0) |

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.Rope.coiled_length</td>
<td>(additional) length of the rope that is provided by a coil. length = rope_length + coiled_length</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

<table>
<thead>
<tr>
<th>Internal data variable</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-3</td>
<td></td>
</tr>
</tbody>
</table>

| Connector.Rope.coiled_length | (additional) length of the rope that is provided by a coil. length = rope_length + coiled_length |
| Connector.Rope.force | force in the rope |
| Connector.Rope.rope_length | length of the rope |

Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.Rope.coiled_length</td>
<td>(additional) length of the rope that is provided by a coil. length = rope_length + coiled_length</td>
</tr>
<tr>
<td>Connector.Rope.coiled_length</td>
<td>(additional) length of the rope that is provided by a coil. length = rope_length + coiled_length</td>
</tr>
</tbody>
</table>

Example

see file Rope3D.txt

Mass
{
  element_type= "Mass3D"
  Physics.mass= 1
  Initialization.initial_position= [0.5, 0.8, 0]
}
nMass = AddElement(Mass)

rope
{
  element_type= "Rope3D"
  name= "Rope3D" %name of the element
  Graphics.draw_size = 0.03
  Physics
  {
    Penalty
    {
      rope_stiffness= 1e3
      damping= 10
    }
  }
  Geometry
  {
    element_numbers= [0, 0, nMass, 0] %element numbers of the suspension points
    positions= [0, 0.5, 0; 0, 1, 0; 0,0,0; 1,1,0]
  }
}
nRope = AddConnector(rope)
3.3.8 FrictionConstraint

Short description

The FrictionConstraint is acting on an arbitrary coordinate, including rotations. It can be used to connect two elements to each other or one element to ground. Up to a specified threshold of the force, the constraint is sticking, which is either realized by a spring-damper formulation (penalty formulation) or with an algebraic equation (lagrange formulation). Above this threshold, a constant friction force is applied during the sliding phase. Alternatively sticking can be switched off and a coulomb friction force, with a transition region for very small velocities, can be applied.

Equations

Lagrange formulation:

\[ \text{sticking:} \]

position constraint (index 3 solver)
2 elements (coordinate to coordinate): \( C = \hat{q}_i^{el1} - \hat{q}_j^{el2} - x_0 = 0 \)
1 element (coordinate to ground): \( C = \hat{q}_i^{el1} - x_0 = 0 \)
velocity constraint - index reduction (index 2 solver)
2 elements (coordinate to coordinate): \( C = \dot{q}_i^{el1} - \dot{q}_j^{el2} = 0 \)
1 element (coordinate to ground): \( C = \dot{q}_i^{el1} = 0 \)

\[ \text{sliding:} \quad C = \lambda - \mu_{\text{kin}} F_n = 0 \]

Penalty formulation:

\[ \text{sticking:} \]

2 elements (coordinate to coordinate): \( F_{st} = c (\hat{q}_i^{el1} - \hat{q}_j^{el2} - x_0) + d (\dot{\hat{q}}_i^{el1} - \dot{\hat{q}}_j^{el2}) \)
1 element (coordinate to ground): \( F_{st} = c (\hat{q}_i^{el1} - x_0) + d \dot{\hat{q}}_i^{el1} \)

\[ \text{sliding:} \quad F_{sl} = \mu_{\text{kin}} F_n \]

Description:

\( \hat{q}_i^{el1} \ldots i^{th} \) coordinate of element 1
\( \hat{q}_j^{el2} \ldots j^{th} \) coordinate of element 2
\( x_0 \ldots \) last sticking position (updated at every slide-stick transition)

Description of the different modi

<table>
<thead>
<tr>
<th>Modus</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>sticking</td>
<td>During sticking phase, the constraint is implemented as spring-damper, with the force ( F_{st} ), the spring stiffness ( c ) and the damping coefficient ( d ) or alternatively with one algebraic equation in lagrange mode.</td>
</tr>
<tr>
<td>sliding</td>
<td>During sliding phase, a constant friction force ( F_{sl} ) is applied. ( F_{sl} ) depends on the normal force ( F_n ). If the flag keep_sliding is active, then a transition region for small velocities is used.</td>
</tr>
</tbody>
</table>
Additional notes

The switching from sticking phase to sliding phase is done automatically, as soon as $F_{st} > \mu_{st}F_n$. The switching to sticking phase is performed when the absolute value of the velocity $v$ is smaller than the specified velocity tolerance.

If the solver does not converge close to the switching points, set the solver option SolverOptions.Discontinuous.ignore_max_iterations = 1.

If you are using index 2 solver it is advised to use RadauIIA and not LobattoIIIA. LobattoIIIA may lead to oscillations of the friction force and therefore unwanted stick-slip transitions.

If you are using the FrictionConstraint in order to constrain rotations, problems may occur when the change of the angle is discontinuous, e.g. if it exceeds $\pi/2$.

![Diagram of friction forces](image)

Figure 3.33: FrictionConstraint with friction forces $F_{st}$ and $F_{sl}$, with sticking (left figure, keep_sliding = 0) and without sticking (right figure, keep_sliding = 1).

Data objects of FrictionConstraint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td></td>
<td>&quot;FrictionConstraint&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td></td>
<td>&quot;FrictionConstraint&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td></td>
<td>-1</td>
<td>Drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used.</td>
</tr>
<tr>
<td>Physics.use_penalty_formulation</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</td>
</tr>
<tr>
<td>Physics.normal_force</td>
<td>double</td>
<td></td>
<td></td>
<td>constant normal force $F_n$</td>
</tr>
<tr>
<td>Physics.velocity_tolerance</td>
<td>double</td>
<td></td>
<td>1e-005</td>
<td>If velocity is below this value, sticking starts, or if 'keep sliding' is active, the transition region is used.</td>
</tr>
<tr>
<td>Physics.fr_coeff_st</td>
<td>double</td>
<td></td>
<td>0.15</td>
<td>static friction coefficient, used to determine the threshold when sliding starts.</td>
</tr>
<tr>
<td>Physics.fr_coeff_kin</td>
<td>double</td>
<td></td>
<td>0.1</td>
<td>kinematic friction coefficient, used to calculate the constant force during sliding phase.</td>
</tr>
<tr>
<td>Physics.keep_sliding</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>The constraint will never go to modus 'stick'.</td>
</tr>
</tbody>
</table>

Physics.Penalty
### Physics.Penalty

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spring_stiffness</td>
<td>double</td>
<td>0</td>
<td>spring stiffness c, only used during sticking phase!</td>
</tr>
<tr>
<td>damping</td>
<td>double</td>
<td>0</td>
<td>damping coefficient d for viscous damping, only used during sticking phase!</td>
</tr>
</tbody>
</table>

### Initialization

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_sliding_velocity</td>
<td>double</td>
<td>0</td>
<td>Initial (relative) sliding velocity between the two kinematic pairs. If absolute value is smaller than 'velocity tolerance' then the constraint starts with 'sticking'.</td>
</tr>
</tbody>
</table>

### Coordinate1

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>0</td>
<td>element number for coordinate 1</td>
</tr>
<tr>
<td>local_coordinate</td>
<td>integer</td>
<td>1</td>
<td>Local coordinate of element 1 to be constrained</td>
</tr>
</tbody>
</table>

### Coordinate2

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>0</td>
<td>element number for coordinate 2; for ground joint, set element number to zero</td>
</tr>
<tr>
<td>local_coordinate</td>
<td>integer</td>
<td>1</td>
<td>Local coordinate of element 2 to be constrained</td>
</tr>
</tbody>
</table>

### Observables special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-4</td>
</tr>
<tr>
<td>Connector.FrictionConstraint.sticking</td>
<td>1 if sticking, 0 if sliding</td>
</tr>
<tr>
<td>Connector.FrictionConstraint.force_forward</td>
<td>force, applied to the kinematic pairs due to the constraint</td>
</tr>
<tr>
<td>Connector.FrictionConstraint.force_forward_abs</td>
<td>absolute value of the force, applied to the kinematic pairs due to the constraint</td>
</tr>
<tr>
<td>Connector.FrictionConstraint.force_normal</td>
<td>normal force F_n. Sliding force F_sl is directly proportional to F_n.</td>
</tr>
</tbody>
</table>

### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.FrictionConstraint.force_normal</td>
<td>normal force F_n. Sliding force F_sl is directly proportional to F_n.</td>
</tr>
</tbody>
</table>
Example

see file FrictionConstraint.txt

force
{
    load_type = "ForceVector3D"
    force_vector= [1, 0, 0]
    load_function_type= 1 %time dependency of the load: 1..MathFunction
    MathFunction
    {
        piecewise_mode= 1 %modus for piecewise interpolation: 1=linear
        piecewise_points= [0, 0.08, 0.081, 0.2] %supporting points
        piecewise_values= [0, 50, -50, 0] %values at supporting points
    }
}

nLoad=AddLoad(force)

test_mass
{
    element_type = "Mass3D"
    Physics.mass = 1
    loads= [nLoad]
}

nMass = AddElement(test_mass)

friction
{
    element_type= "FrictionConstraint"
    name= "FrictionConstraint"
    Physics
    {
        normal_force= 10
        fr_coeff_st= 0.15
        fr_coeff_kin= 0.1
    }
    Coordinate1
    {
        element_number= nMass %element number for coordinate 1
        local_coordinate= 1 %Local coordinate of element 1 to be constrained
    }
}

nFriction=AddConnector(friction)

sensfriction
{
    name = "sticking"
    sensor_type= "ElementSensor"
    element_number= nFriction
    value= "Connector.FrictionConstraint.sticking"
}

AddSensor(sensfriction)

sensfriction.name="friction_force"
sensfriction.value = "Connector.FrictionConstraint.force_forward"
nSensFriction = AddSensor(sensfriction)

SolverOptions {
  end_time = 0.2
  TimeInt.max_step_size = 1e-5
  Newton.relative_accuracy = 1
  Newton.use_modified_newton = 1
  Linalg.use_sparse_solver = 1
  Discontinuous.ignore_max_iterations = 1
}
ViewingOptions.Loads.show_loads = 1

3.3.9 Contact1D

Short description
Contact1D realizes a contact formulation between two elements or one element and ground. Only one coordinate (direction) is considered per element.

Geometry
Figure 3.34 shows the meaning of the values local coordinate and position in the case of a ground constraint. The only direction which is considered is that defined by Coordinate1.local coordinate. Figure 3.34 shows the case for 2 elements. The value Physics.direction, dir in the following equations, is used to define how the elements are located w.r.t. each other.
ATTENTION: Be careful when using coordinates which do not represent a position!

Equations
Some general definitions:

\[ pos = coordinate + localposition \] (3.25)
\[ u = \text{dir}(pos_1 - pos_2) \] (3.26)
\[ v = \text{dir}(vel_1 - vel_2) \] (3.27)

Mode 1:
if \( u \geq 0 \):
\[ F = 0 \] (3.28)
else:
\[ F = \text{dir}(cu + dv) \] (3.29)

Description of the different modi

<table>
<thead>
<tr>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penalty Formulation with spring and damper. The bodies will penetrate slightly according to the spring stiffness. Results may depend on chosen step size!</td>
<td>Lagrange Formulation (not implemented yet)</td>
</tr>
</tbody>
</table>
3.3. **CONNECTOR**

![Diagram of gravity and coordinate positions](image1)

**Figure 3.34:** Description of the geometry options in the case of a ground constraint.

![Diagram of two elements](image2)

**Figure 3.35:** Description of the geometry options in the case of 2 elements.

**Data objects of Contact1D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;Contact1D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Contact1D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

| Graphics.show_connector    | bool      | 1 | Flag to draw connector |
| Graphics.draw_size         | double    | -1 | Drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used. |

**Physics**

| Physics.direction          | double    | 1 | Direction of the contact: +1 if the first body is on top, or else -1 |
| Physics.mode               | integer   | 1 | mode of computation |

**Physics.Model**

| Physics.Model.spring_stiffness | double | 0 | spring stiffness c |
| Physics.Model.damping        | double | 0 | damping coefficient d for viscous damping |

**Coordinate1**

| Coordinate1.local_coordinate | integer | 1 | Local coordinate of element 1 to be constrained |
| Coordinate1.position         | double  | 0 | Local position at which contact occurs |
| Coordinate1.element_number   | integer | 1 | element number for coordinate 1; set to zero if you use nodal coordinates! |
CHAPTER 3. HOTINT REFERENCE MANUAL

<table>
<thead>
<tr>
<th>Coordinate1. node_number</th>
<th>integer</th>
<th>0</th>
<th>just used if element number = 0; node number for coordinate 1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Coordinate2</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate2. local_coordinate</td>
<td>integer</td>
<td>1</td>
<td>Local coordinate of element 2 to be constrained (not used if ground constraint)</td>
</tr>
<tr>
<td>Coordinate2.position</td>
<td>double</td>
<td>0</td>
<td>Local (or global if ground) position at which contact occurs</td>
</tr>
<tr>
<td>Coordinate2. element_number</td>
<td>integer</td>
<td>0</td>
<td>element number for coordinate 2; for ground joint or nodal coordinates, set element number to zero</td>
</tr>
<tr>
<td>Coordinate2. node_number</td>
<td>integer</td>
<td>0</td>
<td>just used if element number = 0; node number for coordinate 2; for ground joint, set node number to zero</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section [3.1]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Connector.Contact.force</td>
<td>the force applied to the coordinates due to the contact</td>
</tr>
</tbody>
</table>

Example

see file Contact1D.txt

load.load_type= "Gravity"
load.gravity_constant= -9.81
nLoad = AddLoad(load)

r = 0.1
mass  % define point mass
{
element_type= "Mass2D"
loads= [nLoad]
Initialization.initial_position= [1,0]
Physics.mass= 1
Graphics.radius = r
}
nElem1 = AddElement(mass)

contact  % add contact
{
element_type= "Contact1D"
Graphics.draw_size = 0.01
Physics
3.3. CONNECTOR

{  
  mode= 1 % mode of computation  
  Model1.spring_stiffness= 1e6 % spring stiffness c  
  Model1.damping= 5e2 % damping coefficient d for viscous damping  
}

Coordinate1

{  
  local_coordinate= 1 % coord 1 of element 1 is x-direction!  
  position= -r % offset in x-direction  
  element_number= nElem1 % element number for coordinate 1  
}
 % ground constraint without offset: no entries for Coordinate 2 needed

AddConnector(contact)

SolverOptions.Discontinuous.absolute_accuracy = 0.001  
SolverOptions.end_time = 2

3.3.10 PlaneConstraint

Short description

PlaneConstraint forces a material point (given by global node number, or by element index and local node number or local position) to reside in a given plane. The plane is defined by its unit normal (Geometry.Plane.normal) and an arbitrary point on the plane (Geometry.Plane.ground). If the material point is defined by a global node number, then setting penalty formulation is mandatory!

Data objects of PlaneConstraint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;PlaneConstraint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;PlaneConstraint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

Graphics

| Graphics.RGB_color | vector          | [0.3, 0.8, 0.3] | red, green, blue | color of element, range = 0..1, use default color: [-1,-1,-1] |
| Graphics.show_connector | bool             | 1              | Flag to draw connector |

Geometry

<table>
<thead>
<tr>
<th>Geometry.use_local_coordinate_system</th>
<th>bool</th>
<th>0</th>
<th>0=use global coordinates, 1=use local coordinate system of Body 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.Plane.normal</td>
<td>vector</td>
<td>[0, 0, 1]</td>
<td>normal of plane</td>
</tr>
<tr>
<td>Geometry.Plane.ground</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>arbitrary position on plane</td>
</tr>
</tbody>
</table>

Physics

<table>
<thead>
<tr>
<th>Physics.use_penalty_formulation</th>
<th>bool</th>
<th>0</th>
<th>0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Penalty.spring_stiffness</td>
<td>double</td>
<td>0</td>
<td>general or penalty stiffness parameter</td>
</tr>
</tbody>
</table>

Position
### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-6</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector</td>
</tr>
</tbody>
</table>

### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness</td>
<td>Set the stiffness coefficient</td>
</tr>
<tr>
<td>Connector.damping</td>
<td>Set the damping coefficient</td>
</tr>
<tr>
<td>Connector.Geometry.Plane.normal</td>
<td>Set the damping coefficient</td>
</tr>
<tr>
<td>Connector.Geometry.Plane.ground</td>
<td>Set the damping coefficient</td>
</tr>
</tbody>
</table>

### Example

see file PlaneConstraintStaticsShort.txt

```
HOTINT_data_file_version="1.3.36"
SolverOptions.do_static_computation = 1
SolverOptions.start_time = 0
SolverOptions.end_time = 1
SolverOptions.Linalg.use_sparse_solver = 1
SolverOptions.Newton.use_modified_newton = 1

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2e11
    Solid.poisson_ratio = 0.3
```
3.3. CONNECTOR

}\n\texttt{mnr} = \texttt{AddMaterial(blockmaterial)}

\texttt{eps} = 1e-6
\texttt{a} = 1
\texttt{N elems} = 2

\texttt{meshparameters.mesh\_type} = "SolidMesh"
\texttt{meshparameters.mesh\_name} = "msh"
\texttt{msh} = \texttt{GenerateNewMesh(meshparameters)}

\texttt{blockparameters}
{
\hspace{1em} \texttt{component\_type} = "Block"
\hspace{1em} \texttt{Generation}
\hspace{2em} \{\n\hspace{3em} \texttt{P1} = [0,0,0]
\hspace{3em} \texttt{P2} = [a,0,0]
\hspace{3em} \texttt{P3} = [0,a,0]
\hspace{3em} \texttt{P4} = [a,a,0]
\hspace{3em} \texttt{P5} = [0,0,a]
\hspace{3em} \texttt{P6} = [a,0,a]
\hspace{3em} \texttt{P7} = [0,a,a]
\hspace{3em} \texttt{P8} = [a,a,a]
\hspace{3em} \texttt{discretization} = [\texttt{N elems},\texttt{N elems},\texttt{N elems}]
\hspace{3em} \texttt{Material\_number} = \texttt{mnr}
\hspace{2em} \}
\}

\texttt{msh}\texttt{.GenerateBlock(blockparameters)}
\texttt{lin2quadparams.name} = "muh"
\texttt{lin2quadparams.Generation.GeometricNonlinearityStatus} = 2
\texttt{msh}\texttt{.Linear2Quadratic(1,lin2quadparams)}
\texttt{msh}\texttt{.AddMeshToMBS(1)}

\texttt{%%% BOUNDARY CONDITIONS (plane constraints only)}

\texttt{%%% common definitions \textit{==}}
\texttt{u_y} = 0.1
\texttt{p_x} = a
\texttt{p_y} = \texttt{u_y}

\texttt{n_{xy\_bot}} = [0,0,-1]
\texttt{n_{xy\_top}} = [0,0,1]
\texttt{n_{yz\_bot}} = [-1,0,0]
\texttt{n_{yz\_top}} = [1,0,0]
\texttt{n_{xz\_bot}} = [p_y, -p_x, 0]
\texttt{n_{xz\_top}} = [-p_y, p_x, 0]
\texttt{p_{bot}} = [0,0,0]
\texttt{p_{top}} = [a, a + u_y, a]
planeconstr.element_type = "PlaneConstraint"
node_set.set_type = "GlobalNodeSet"

% === xz bottom === plane constraint ===
boxparams.P1 = [-eps,-eps,-eps]
boxparams.P2 = [a+eps,eps,a+eps]

node_set.set_name = "XZBotNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_bot
planeconstr.Geometry.Plane.normal = n_xz_bot
planeconstr.Graphics.RGB_color = [0.3, 0.8, 0.3]

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

% === xz top === plane constraint ===
boxparams.P1 = [-eps,a-eps,-eps]
boxparams.P2 = [a+eps,a+eps,a+eps]

node_set.set_name = "XZTopNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_top
planeconstr.Geometry.Plane.normal = n_xz_top

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

% === yz bottom === plane constraint ===
boxparams.P1 = [-eps,-eps,-eps]
boxparams.P2 = [eps,a+eps,a+eps]

node_set.set_name = "YZBotNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_bot
planeconstr.Geometry.Plane.normal = n_yz_bot
3.3. CONNECTOR

planeconstr.Graphics.RGB_color = [0.8, 0.3, 0.3]

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

% === yz top === plane constraint ===

boxparams.P1 = [a-eps,-eps,-eps]
boxparams.P2 = [a+eps,a+eps,a+eps]

node_set.set_name = "YZTopNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_top
planeconstr.Geometry.Plane.normal = n_yz_top

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

% === xy bottom === plane constraint ===

boxparams.P1 = [-eps,-eps,-eps]
boxparams.P2 = [a+eps,a+eps,a+eps]

node_set.set_name = "XYBotNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_bot
planeconstr.Geometry.Plane.normal = n_xy_bot
planeconstr.Graphics.RGB_color = [0.3, 0.3, 0.8]

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

% === xy top === plane constraint ===

boxparams.P1 = [-eps,-eps,a-eps]
boxparams.P2 = [a+eps,a+eps,a+eps]
node_set.set_name = "XYTopNodes"
node_set.global_node_numbers = msh.GetNodesInBox(boxparams)
n_node_set = AddSet(node_set)
node_set_data = AccessSet(n_node_set)
planeconstr.Geometry.Plane.ground = p_top
planeconstr.Geometry.Plane.normal = n_xy_top

for(k_node=1,k_node<=cols(node_set_data.local_node_numbers),k_node=k_node+1)
{
    planeconstr.Position.node_number = node_set_data.local_node_numbers[k_node]
    planeconstr.Position.element_number = node_set_data.element_numbers[k_node]
    AddConnector(planeconstr)
}

3.3.11 GenericBodyJoint

Short description
The GenericBodyJoint constrains two elements at a local position each. If only one element is specified (second element 0), a ground GenericBodyJoint is realized. A penalty and Lagrange formulation is available.
The constraint forces and moments are applied as follows:

Connecting element to element:
The constraint forces and moments are applied on both elements at the position of the connection point of the second element.

Connecting element to ground:
The constraint forces are applied at the position of the connection point of the element.

Equations
Lagrange equations:
The constraint equations for translation are
\[ C_{\text{trans}} = A^T (x_1 - x_2) = 0 \]
Each equation in \( C_{\text{trans}} \) corresponds to a constrained direction. Hence only those equations corresponding to the constrained directions are considered. If all directions are constrained, \( C_{\text{trans}} \) simplifies to
\[ C_{\text{trans}} = x_1 - x_2 = 0 \]
since \( A^T (x_1 - x_2) = 0 \iff x_1 - x_2 = 0 \).
If all rotations are constrained, then the constraint equations for rotation are
\[ C_{\text{rot}} = \begin{pmatrix} e_j^i \cdot e_z^i \\ e_z^i \cdot e_j^i \\ e_z^i \cdot e_y^i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} . \]
If the rotation about the \( x \)-axis is not constrained, then
\[ C_{\text{rot}} = \begin{pmatrix} e_j^i \cdot e_z^i \\ e_z^i \cdot e_y^i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \]
If the rotation about the $y$-axis is not constrained, then
\[
C_{rot} = \left( \begin{array}{c} e_y^i \cdot e_y^i \\ e_y^j \cdot e_z^j \end{array} \right) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right).
\]

If the rotation about the $z$-axis is not constrained, then
\[
C_{rot} = \left( \begin{array}{c} e_z^j \cdot e_y^i \\ e_z^j \cdot e_x^i \end{array} \right) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right).
\]

Where
\[
\begin{align*}
&x_1 \quad \text{position of connection point on body 1 in global coordinates} \\
&x_2 \quad \text{position of connection point on body 2 in global coordinates, or if constraint connects element to ground then connection point of ground in global coordinates} \\
&v_1 \quad \text{time derivative of } x_1 \\
&v_2 \quad \text{time derivative of } x_2 \\
&A \quad \text{rotation matrix from local joint coordinates to global coordinates.} \\
&B \quad B = Q_i J^* \\
&Q_i \quad \text{rotation matrix from local coordinate system of body 1 to global coordinates} \\
&Q_j \quad \text{rotation matrix from local coordinate system of body 2 to global coordinates} \\
&J \quad \text{joint local frame} \\
&J^* \quad J^* = Q_j^T \big|_{t=0} Q_i \big|_{t=0} J \\
&e_x^i, e_y^i, e_z^i \quad \left( \begin{array}{c} e_x^i \\ e_y^i \\ e_z^i \end{array} \right) = A \\
&e_x^j, e_y^j, e_z^j \quad \left( \begin{array}{c} e_x^j \\ e_y^j \\ e_z^j \end{array} \right) = B
\end{align*}
\]

**Penalty equations:**

The stiffness and damping force is given by
\[
f = AK_{\text{trans}} A^T u + AD_{\text{trans}} A^T v
\]

The stiffness and damping moment is given by
\[
m = K_{\text{rot}} \varphi + D_{\text{rot}} \omega
\]

Where
\[
\begin{align*}
f & \quad \text{constraint force due to stiffness and damping} \\
m & \quad \text{constraint moment due to stiffness and damping} \\
K_{\text{trans}} & \quad \text{stiffness matrix for translation} \\
D_{\text{trans}} & \quad \text{damping matrix for translation} \\
K_{\text{rot}} & \quad \text{stiffness matrix for rotation} \\
D_{\text{rot}} & \quad \text{damping matrix for rotation} \\
\varphi & \quad \text{relative angles between body 1 and body 2 or absolute angles of body 1 if body 1 is connected to ground} \\
\omega & \quad \text{relative angular velocities between body 1 and body 2 or absolute angular velocities of body 1 if body 1 is connected to ground.}
\end{align*}
\]
If all rotations are constrained, linearized angles
\[
\varphi = \begin{pmatrix}
\varphi_x \\
\varphi_y \\
\varphi_z
\end{pmatrix} = \begin{pmatrix}
-e^j_y \cdot e^i_z \\
e^j_z \cdot e^i_z \\
-e^j_x \cdot e^i_y
\end{pmatrix}.
\]
and linearized angular velocities are used
\[
\omega = \begin{pmatrix}
\omega_x \\
\omega_y \\
\omega_z
\end{pmatrix} = \begin{pmatrix}
-\dot{e}^j_y \cdot e^i_z - e^j_y \cdot \dot{e}^i_z \\
\dot{e}^j_z \cdot e^i_z + e^j_z \cdot \dot{e}^i_z \\
\dot{e}^j_x \cdot e^i_y - e^j_x \cdot \dot{e}^i_y
\end{pmatrix}.
\]

**Limitations**

It is strongly recommended to prefer the Lagrangian method for free rotation instead of penalty formulation to avoid simulation problems.

The constraint forces have to act for both bodies at the same position. This means, that if the constraint is in penalty mode, or if not all directions are constrained, the constraint forces need to be applied outside the connection point of at least one body. In case of the GenericBodyJoint, the constraint forces are applied at the connection position of the second element if two elements are constrained, or if one element is constrained, the constraint forces are applied at the connection position of the element.

So if connecting two elements with a GenericBodyJoint, the constraint forces have to be applied outside the first bodies connection point. For rigid bodies this is equivalent to applying the force at the connection point and applying a moment which compensates the moment induced by the shifting of the force. Applying forces on flexible bodies outside the connection point gives various problems, like what happens if the force is outside the body, etc.

Therefore flexible bodies are treated like rigid bodies and the force is applied to the connection point and a moment is applied, also on the position of the connection point, which compensates the moment induced by shifting the force.

If you need a constraint which allows the sliding of an element on a flexible body please use a SlidingPointJoint (3.3.5) or a SlidingPrismaticJoint (3.3.6).

**Description of the different modi**

<table>
<thead>
<tr>
<th>element to ground</th>
<th>Position2.element_number AND Position2.node_number must have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>element to element</td>
<td>Position2.element_number AND/OR Position2.node_number must not be equal to 0</td>
</tr>
<tr>
<td>Lagrange</td>
<td>Physics.use_penalty_formulation must be set to 0. Set the vector of constrained directions in Physics.Lagrange.constrained_directions ([x, y, z], 1 = constrained, 0 = free). The directions are w.r.t the local body 1 joint coordinate system. Set the vector of constrained rotations in Physics.Lagrange.constrained_rotations ([\phi_x, \phi_y, \phi_z], 1 = constrained, 0 = free). The rotations are about the axes of local body 1 joint coordinate system.</td>
</tr>
</tbody>
</table>
Penalty

Physics.use_penalty_formulation must be set to 1. In Physics.Penalty.stiffness_matrix and Physics.Penalty.damping_matrix all parameters for translational stiffness and damping w.r.t. local body 1 coordinate system can be set. In Physics.Penalty.stiffness_matrix_rotation and Physics.Penalty.damping_matrix_rotation all parameters for rotational stiffness and damping w.r.t. local body 1 coordinate system can be set.

Figure 3.36: GenericBodyJoint

Figure 3.37: The constraint forces act for both bodies on the position of $x_2$. The force acting on the first body is shifted to $x_1$ (dashed) and a moment is applied to compensate the induced moment through shifting.

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;GenericBodyJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;GenericBodyJoint&quot;</td>
<td>name of the element</td>
</tr>
</tbody>
</table>
### Graphics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.draw_size_joint_local_frame</td>
<td>double</td>
<td>Drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn.</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>Cone size for standard joint drawing</td>
</tr>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3] [red, green, blue] first color of constraint, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3] [red, green, blue] second color of constraint, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
</tbody>
</table>

### Geometry

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.joint_local_frame</td>
<td>matrix</td>
<td>R [1, 0, 0; 0, 1, 0; 0, 0, 1]</td>
</tr>
<tr>
<td>Geometry.joint_local_frame_in_bryant_angles</td>
<td>vector</td>
<td>Prerotate joint coordinate system w.r.t. local coordinate system of body 1 [phi x, phi y, phi z]. Rot. sequence: JA0i=A(phi z)A(phi y)A(phi x)</td>
</tr>
</tbody>
</table>

### Physics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.use_penalty_formulation</td>
<td>bool</td>
<td>0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</td>
</tr>
<tr>
<td>Physics.Penalty.stiffness_matrix</td>
<td>matrix</td>
<td>3x3 matrix with stiffness parameters</td>
</tr>
<tr>
<td>Physics.Penalty.damping_matrix</td>
<td>matrix</td>
<td>3x3 matrix with damping parameters</td>
</tr>
<tr>
<td>Physics.Penalty.stiffness_matrix_rotation</td>
<td>matrix</td>
<td>3x3 matrix with stiffness parameters for rotation</td>
</tr>
<tr>
<td>Physics.Penalty.damping_matrix_rotation</td>
<td>matrix</td>
<td>3x3 matrix with damping parameters for rotation</td>
</tr>
</tbody>
</table>

### Position1

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position1.element_number</td>
<td>integer</td>
<td>1 Number of constrained element</td>
</tr>
<tr>
<td>Position1.position</td>
<td>vector</td>
<td>[0, 0, 0] local position. Only used if node_number == 0!</td>
</tr>
</tbody>
</table>

### Position2

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position2.element_number</td>
<td>integer</td>
<td>0 Number of constrained element</td>
</tr>
<tr>
<td>Position2.position</td>
<td>vector</td>
<td>[0, 0, 0] local or global (if element_number == 0) position. Only used if node_number == 0!</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction.</td>
</tr>
<tr>
<td>Connector.moment</td>
<td>internal global moment of connector</td>
</tr>
<tr>
<td>Connector.force_local</td>
<td>internal local force of connector (joint coordinate system J Ai)</td>
</tr>
<tr>
<td>Connector.moment_local</td>
<td>internal local moment of connector (joint coordinate system J Ai)</td>
</tr>
<tr>
<td>Connector.displacement</td>
<td>displacement between the joint coordinate systems J Ai and J Aj expressed in coordinate system J Ai</td>
</tr>
<tr>
<td>Connector.angle</td>
<td>Bryant angles between the joint coordinate systems J Ai and J Aj. All constrained components are zero.</td>
</tr>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.joint_bryant_angle</td>
<td>prescribe the angles of the joint coordinate system (for actuation, penalty formulation ONLY!)</td>
</tr>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Example

see file GenericBodyJointShort.txt

l = 1 m

rigidBody
{
  element_type = "Rigid3D"
  Graphics.body_dimensions = [1, 0.05, 0.05]
}

nRigid = AddElement(rigidBody)

genericBodyJoint
{
  element_type = "GenericBodyJoint"
  Position1
3.3.12 RevoluteJoint

Short description
The RevoluteJoint constrains all relative degrees of freedom between two bodies except the rotation about a local rotation axis. A penalty formulation exists, which replaces the exact lagrange constraint by an approximation with joint stiffness and damping. This constraint can be used together with a RotatorySpringDamperActuator (3.3.19). The RevoluteJoint is equivalent to a GenericBodyJoint (3.3.11) with all directions and rotations constrained except the rotation about the local x axis. The joint local frame is chosen such that the local x axis is the rotation axis. Please read also the documentation of GenericBodyJoint for details and limitations.

The constraint forces and moments are applied as follows:

- **Connecting element to element:**
  The constraint forces and moments are applied on both elements at the position of the connection point of the second element.

- **Connecting element to ground:**
  The constraint forces are applied on the position of the connection point of the element.

Limitations

"In penalty formulation the constraints damps the relative velocity of the two connection points in global coordinates, hence if the penalty stiffness is low and the forces high, then a damping of the rotation is possible.

![Figure 3.38: RevoluteJoint](image_url)
### 3.3. CONNECTOR

| element_type | string | "RevoluteJoin" specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore! |
| name | string | "RevoluteJoin" name of the element |
| element_number | integer | R 2 number of the element in the mbs |

#### Graphics

| Graphics.show_connector | bool | 1 Flag to draw connector |
| Graphics.draw_size_joint_local_frame | double | 0 drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn. |
| Graphics.draw_size | double | -1 cone size for standard joint drawing |
| Graphics.color_body1 | vector | [0.3, 0.8, 0.3] [red, green, blue] first color of constraint, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.color_body2 | vector | [0.7, 0.8, 0.3] [red, green, blue] second color of constraint, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.standard_joint_drawing | bool | 1 flag for drawing mode; 1 == draw constraint element; 0 == show constrained directions and rotations; |
| Graphics.diameter | double | -1 diameter of the revolute joint (for drawing) |
| Graphics.axis_length | double | -1 axis length of the revolute joint (for drawing) |

#### Physics

| Physics.use_penalty_formulation | bool | 0 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |
| Physics.Penalty.damping | double | 100 damping parameter used for translation and rotation |
| Physics.Penalty.stiffness | double | 1e+006 stiffness parameter used for translation and rotation |

#### Physics.Lagrange

| Physics.Lagrange.constrained_directions | vector | R [1, 1, 1] constrained directions cannot be changed |
| Physics.Lagrange.constrained_rotations | vector | R [0, 1, 1] constrained rotations cannot be changed |
| Physics.rotation_axis | vector | [1, 0, 0] local rotation axis w.r.t body 1 coordinate system |

#### Position 1

| Position1.element_number | integer | 1 Number of constrained element |
| Position1.position | vector | [0, 0, 0] local position. Only used if node_number == 0! |

#### Position 2

| Position2.element_number | integer | 0 Number of constrained element |
| Position2.position | vector | [0, 0, 0] local or global (if element_number == 0) position. Only used if node_number == 0! |

---

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-5</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-5</td>
</tr>
</tbody>
</table>
Connector.force force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction

Connector.moment internal global moment of connector

Connector.force_local internal local force of connector (joint coordinate system JAi)

Connector.moment_local internal local moment of connector (joint coordinate system JAi)

Connector.displacement displacement between the joint coordinate systems JAi and JAj expressed in coordinate system JAi

Connector.angle Bryant angles between the joint coordinate systems JAi and JAj. All constrained components are zero.

Connector.stiffness_matrix stiffness matrix

Connector.damping_matrix damping matrix

Connector.stiffness_matrix_rotation stiffness matrix for rotation

Connector.stiffness_matrix_damping damping matrix for rotation

Connector.local_position_1 local position on element 1

Connector.local_position_2 local position on element 2

Controllable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Example
see file RevoluteJointShort.txt

```plaintext
l = 1 % m
g = 9.81 % m/s^2
g
load_type = "Gravity"
direction = 3 % z - direction
gravity_constant = g
}
gravLoad = AddLoad(gravLoad)

rigidBody
{
element_type= "Rigid3D"
loads= [nLoad]
```
Connectors

Graphics.body_dimensions= [1, 0.05, 0.05]
}

nRigid = AddElement(rigidBody)

revoluteJoint
{
  element_type= "RevoluteJoint"
  Physics.rotation_axis= [0, 1, 0] %local rotation axis
  Position 1
  {
    element_number= nRigid %number of constrained element
    position= [-1/2, 0, 0] %local position
  }
}

AddConnector(revoluteJoint)

### 3.3.13 PrismaticJoint

**Short description**

The PrismaticJoint constrains all relative degrees of freedom between two bodies except the translation along a local sliding axis. A penalty formulation exists, which replaces the exact Lagrange constraint by an approximation with joint stiffness and damping.

The PrismaticJoint is equivalent to a GenericBodyJoint (3.3.11) with all directions and rotations constrained except the translation about the local x axis. The joint local frame is chosen such that the local x axis is the sliding axis. Please read also the documentation of GenericBodyJoint for details and limitations.

If the first body is a flexible body, then you might consider using the SlidingPrismaticJoint (3.3.6).

The constraint forces and moments are applied as follows:

- **Connecting element to element:**
  The constraint forces and moments are applied on both elements at the position of the connection point of the second element.

- **Connecting element to ground:**
  The constraint forces are applied on the position of the connection point of the element.

![Figure 3.39: PrismaticJoint](image)

**Data objects of PrismaticJoint:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;PrismaticJoint&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;PrismaticJoint&quot; name of the element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 2 number of the element in the mbs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1 Flag to draw connector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.draw_size_joint_local_frame</td>
<td>double</td>
<td>0 drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>-1 cone size for standard joint drawing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3] [red, green, blue] first color of constraint, range = 0..1, use default color:[-1,-1,-1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3] [red, green, blue] second color of constraint, range = 0..1, use default color:[-1,-1,-1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.standard_joint_drawing</td>
<td>bool</td>
<td>1 flag for drawing mode; 1 == draw constraint nicely; 0 == show constrained directions and rotations;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.rail_length</td>
<td>double</td>
<td>-1 length of the prismatic joint (for drawing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.joint_cube_size</td>
<td>vector</td>
<td>[-1, -1, -1] cube dimension of prismatic joint (for drawing); [lx (in sl. dir.),ly (normal to sl. dir.),lz (normal to sl. dir.)]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.use_penalty_formulation</td>
<td>bool</td>
<td>0 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.Penalty.damping</td>
<td>double</td>
<td>100 damping parameter used for translation and rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.Penalty.stiffness</td>
<td>double</td>
<td>1e+006 stiffness parameter used for translation and rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.Lagrange</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.Lagrange.constrained_directions</td>
<td>vector</td>
<td>R [0, 1, 1] constrained directions cannot be changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.Lagrange.constrained_rotations</td>
<td>vector</td>
<td>R [1, 1, 1] constrained rotations cannot be changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.sliding_direction</td>
<td>vector</td>
<td>[1, 0, 0] local sliding direction w.r.t body 1 coordinate system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position1.element_number</td>
<td>integer</td>
<td>1 Number of constrained element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position1.position</td>
<td>vector</td>
<td>[0, 0, 0] local position. Only used if node_number == 0!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position2.element_number</td>
<td>integer</td>
<td>0 Number of constrained element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position2.position</td>
<td>vector</td>
<td>[0, 0, 0] local or global (if element_number == 0) position. Only used if node_number == 0!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Observable special values:
For more information see section 3.1
3.3. CONNECTOR

<table>
<thead>
<tr>
<th><strong>Internal.DOF</strong></th>
<th>degrees of freedom (or generalized unknowns) of the element. range: 1-5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal.algebraic_variable</strong></td>
<td>algebraic variables of the element. range: 1-5</td>
</tr>
<tr>
<td><strong>Connector.force</strong></td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td><strong>Connector.moment</strong></td>
<td>internal global moment of connector</td>
</tr>
<tr>
<td><strong>Connector.force_local</strong></td>
<td>internal local force of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td><strong>Connector.moment_local</strong></td>
<td>internal local moment of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td><strong>Connector.displacement</strong></td>
<td>displacement between the joint coordinate systems JAi and JAj expressed in coordinate system JAi</td>
</tr>
<tr>
<td><strong>Connector.angle</strong></td>
<td>bryant angles between the joint coordinate systems JAi and JAj. All constrained components are zero.</td>
</tr>
<tr>
<td><strong>Connector.stiffness_matrix</strong></td>
<td>stiffness matrix</td>
</tr>
<tr>
<td><strong>Connector.damping_matrix</strong></td>
<td>damping matrix</td>
</tr>
<tr>
<td><strong>Connector.stiffness_matrix_rotation</strong></td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td><strong>Connector.stiffness_matrix_damping</strong></td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td><strong>Connector.local_position_1</strong></td>
<td>local position on element 1</td>
</tr>
<tr>
<td><strong>Connector.local_position_2</strong></td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Controllable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Example
see file PrismaticJointShort.txt

```plaintext
l = 1 % m

force
{
    load_type = "ForceVector3D"
    force_vector = [10,10,10]
}

nForce = AddLoad(force)

rigidBody
{
    element_type= "Rigid3D"
}```
loads = [nForce]
Graphics.body_dimensions = [1, 0.05, 0.05]
}
nRigid = AddElement(rigidBody)

prismaticJoint
{
   element_type = "PrismaticJoint"
   Physics.sliding_direction = [1,0,0]
   Position1
   {
      element_number = nRigid %number of constrained element
      position = [-l/2, 0, 0] %local position
   }
   Position2.position = [-l/2, 0, 0]
}
AddConnector(prismaticJoint)

3.3.14 UniversalJoint

Short description
The UniversalJoint constrains the local position of two elements and keeps two axes, one on each body, perpendicular to each other.

Degrees of freedom
The vector of the DOF contains the Lagrangian parameters \( \lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]^T \), where \( \lambda_1, \lambda_2, \lambda_3 \) are measures for the violation of the displacement condition and \( \lambda_4 \) is a measure for the violation of the orthogonality condition of the two axes.

Geometry
For this constraint one needs to specify the axes of the cross and the directions in which the hinges are drawn. The direction of the hinge and the axis connected to this hinge have to be given in the local coordinate system of the respective body. See figure 3.41

Equations
The positions and axes are given in local coordinates of body 1 respectively body 2. However the calculations are done internally in global coordinates.
Let
\[
\begin{align*}
\mathbf{x}^i &= \begin{bmatrix} x^i_1 & x^i_2 & x^i_3 \end{bmatrix}^T \\
\mathbf{x}^j &= \begin{bmatrix} x^j_1 & x^j_2 & x^j_3 \end{bmatrix}^T \\
\mathbf{a}^i &= \begin{bmatrix} a^i_1 & a^i_2 & a^i_3 \end{bmatrix}^T
\end{align*}
\]
be the position (in global coordinates) where the joint is connected to the first body and let
be the position (in global coordinates) where the joint is connected to the second body.
Let
be the axis (in global coordinates) connected to the first body and let

\[ \mathbf{a}^j = \begin{bmatrix} a_1^j & a_2^j & a_3^j \end{bmatrix}^T \]

be the axis (in global coordinates) connected to the second body. Then the constraint equations at position level are

\[ \mathbf{C} = \begin{bmatrix} \mathbf{x}^i - \mathbf{x}^j \\ \mathbf{a}^iT \cdot \mathbf{a}^j \end{bmatrix} = 0. \]

The first three constraints restrict the position of the connection points of body 1 and 2. The fourth equation ensures that the two axes of the cross are perpendicular to each other.

The constraint equations at velocity level are

\[ \mathbf{C} = \begin{bmatrix} \frac{\partial \mathbf{x}^i}{\partial t} - \frac{\partial \mathbf{x}^j}{\partial t} \\ \frac{\partial \mathbf{a}^i}{\partial t} \cdot \frac{\partial \mathbf{a}^j}{\partial t} \end{bmatrix} = 0. \]

**Limitations**

No penalty formulation is available.
Data objects of UniversalJoint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;UniversalJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;UniversalJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>red, green, blue color of the hinge connected to the first body, range = 0..1</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3]</td>
<td>red, green, blue color of the hinge connected to the first body, range = 0..1</td>
</tr>
<tr>
<td>Graphics.color_cross</td>
<td>vector</td>
<td>[0.2, 0.2, 0.2]</td>
<td>red, green, blue color of the cross shaft</td>
</tr>
<tr>
<td>Graphics.draw_length</td>
<td>double</td>
<td>-1</td>
<td>length of the universal joint (for drawing)</td>
</tr>
<tr>
<td>Graphics.draw_width</td>
<td>double</td>
<td>-1</td>
<td>width of the universal joint (for drawing)</td>
</tr>
<tr>
<td>Graphics.draw_direction_1</td>
<td>vector</td>
<td>[1, 0, 0]</td>
<td>direction from body 1 to joint (for drawing)</td>
</tr>
<tr>
<td>Graphics.draw_direction_2</td>
<td>vector</td>
<td>[-1, 0, 0]</td>
<td>direction from body 2 to joint (for drawing)</td>
</tr>
</tbody>
</table>

**Position1**

<table>
<thead>
<tr>
<th>Position1.element_number</th>
<th>integer</th>
<th>1</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position1.position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local position</td>
</tr>
<tr>
<td>Position1.axis</td>
<td>vector</td>
<td>[0, 1, 0]</td>
<td>the axis of the cross connected to body 1 in local coordinates</td>
</tr>
</tbody>
</table>

**Position2**

<table>
<thead>
<tr>
<th>Position2.element_number</th>
<th>integer</th>
<th>0</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position2.position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local or global (if element_number == 0) position</td>
</tr>
<tr>
<td>Position2.axis</td>
<td>vector</td>
<td>[0, 0, 1]</td>
<td>the axis of the cross connected to body 2 in local coordinates</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-4</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-4</td>
</tr>
</tbody>
</table>

Example
see file UniversalJoint.txt

rotor
{
  element_type= "Rigid3D"
  name= "rotor1"
  Graphics
  {
    body_dimensions= [1, 0.1, 0.1]
  }
  Physics
  {
    moment_of_inertia= [sqr(0.05)*0.5, 0, 0
                        0, 1/12*(3*sqr(0.05)+1), 0
                        0, 0, 1/12*(3*sqr(0.05)+1)]
    volume= sqr(0.05)*Pi
    mass= 1
  }
  Initialization
  {
    initial_rotation = [0, 0, 0]
    initial_position = [0, 0, 0]
    initial_angular_velocity= [0, 0, 0] %Angular velocity vector in global coordinates: [ang_X, ang_Y, ang_Z]
  }
}nRotor1 = AddElement(rotor)

rotor.name = "rotor2"
rotor.Initialization.initial_rotation = [0, 0, pi/4]
rotor.Initialization.initial_position = [0.5+0.5*sqrt(0.5), 0.5*sqrt(0.5), 0]
nRotor2 = AddElement(rotor)

universalJoint
{
  element_type= "UniversalJoint"
  name= "UniversalJoint"
  Graphics
  {
    show_connector= 1
    color_body1 = [0.3, 0.8, 0.3]
3.3.15 RigidJoint

Short description

The RigidJoint constrains the position and relative angles of an element at a specified local position. If only one element is specified, a ground joint is realized. A penalty formulation exists, which replaces the exact lagrange constraint by an approximation with joint stiffness and damping. The RigidJoint is equivalent to a GenericBodyJoint (3.3.11) with all directions and rotations constrained. Please read also the documentation of GenericBodyJoint for details and limitations.

The constraint forces and moments are applied as follows:

**Connecting element to element:**
The constraint forces and moments are applied on both elements at the position of the connection point of the second element.

**Connecting element to ground:**
The constraint forces are applied on the position of the connection point of the element.
3.3. CONNECTOR

Data objects of RigidJoint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;RigidJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;RigidJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>0</td>
<td>drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn.</td>
</tr>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] first color of constraint, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3]</td>
<td>[red, green, blue] second color of constraint, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.standard_joint_drawing</td>
<td>bool</td>
<td>1</td>
<td>flag for drawing mode; 1 == draw constraint element; 0 == show constrained directions and rotations;</td>
</tr>
<tr>
<td>Graphics.cube_length</td>
<td>double</td>
<td>-1</td>
<td>rigid joint dimension (for drawing)</td>
</tr>
</tbody>
</table>

**Physics**

| Physics.use_penalty_formulation | bool  | 0 | 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |
| Physics.Penalty.damping       | double | 100 | damping parameter used for translation and rotation |
| Physics.Penalty.stiffness      | double | 1e+006 |

**Physics.Lagrange**

| Physics.Lagrange.constrained_directions | vector | R | [1, 1, 1] | constrained directions cannot be changed |
| Physics.Lagrange.constrained_rotations | vector | R | [1, 1, 1] | constrained rotations cannot be changed |

**Position1**

| Position1.element_number | integer | 1 | Number of constrained element |
| Position1.position       | vector  | [0, 0, 0] | local position. Only used if node_number == 0! |

**Position2**

| Position2.element_number | integer | 0 | Number of constrained element |
| Position2.position       | vector  | [0, 0, 0] | local or global (if element_number == 0) position. Only used if node_number == 0! |
Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-6</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-6</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td>Connector.moment</td>
<td>internal global moment of connector</td>
</tr>
<tr>
<td>Connector.force_local</td>
<td>internal local force of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.moment_local</td>
<td>internal local moment of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.displacement</td>
<td>displacement between the joint coordinate systems JAi and JAj expressed in coordinate system JAi</td>
</tr>
<tr>
<td>Connector.angle</td>
<td>Bryant angles between the joint coordinate systems JAi and JAj. All constrained components are zero.</td>
</tr>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.damping_matrix_rotation</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Controllable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.damping_matrix_rotation</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

Example
see file RigidJointShort.txt

l = 1 % m
g = 9.81 % m/s^2

gravLoad
{
    load_type = "Gravity"
3.3. \textit{CONNECTOR}

\begin{verbatim}
    direction = 3 \, \% \, z - direction
    gravity_constant = g
\end{verbatim}

\texttt{nLoad} = AddLoad(gravLoad)

\texttt{rigidBody}
\{
    \texttt{element_type} = "Rigid3D"
    \texttt{loads} = [\texttt{nLoad}]
    \texttt{Graphics.body_dimensions} = [1, 0.05, 0.05]
\}
\texttt{nRigid} = AddElement(rigidBody)

\texttt{rigidJoint}
\{
    \texttt{element_type} = "RigidJoint"
    \texttt{Position1}
    \{
        \texttt{element_number} = \texttt{nRigid} \, \% \, \text{number of constrained element}
        \texttt{position} = [-1/2, 0, 0] \, \% \, \text{local position}
    \}
    \texttt{Position2.position} = [-1/2, 0, 0]
\}
AddConnector(rigidJoint)

\section*{3.3.16 \textit{CylindricalJoint}}

\textbf{Short description}

The \textit{CylindricalJoint} constrains like the \textit{RevoluteJoint}, but allows additionally translation along the rotational axis. A penalty formulation exists, which replaces the exact Lagrange constraint by a approximation with joint stiffness and damping.
The \textit{CylindricalJoint} is equivalent to a \textit{GenericBodyJoint} (3.3.11) with all directions and rotations constrained except the translation and rotation about the local x axis. The joint local frame is chosen such that the local x axis is the rotation and sliding axis. Please read also the documentation of \textit{GenericBodyJoint} for details and limitations.
The constraint forces and moments are applied as follows:
\textbf{Connecting element to element:}
The constraint forces and moments are applied on both elements at the position of the connection point of the second element.
\textbf{Connecting element to ground:}
The constraint forces are applied on the position of the connection point of the element.
### Data objects of CylindricalJoint:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;CylindricalJoint&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;CylindricalJoint&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

| Graphics.show_connector            | bool     |   | 1                  | Flag to draw connector                                                     |
| Graphics.draw_size_joint_local_frame| double   |   | 0                  | drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn. |
| Graphics.draw_size                | double   |   | -1                 | cone size for standard joint drawing                                       |
| Graphics.color_body1              | vector   |   | [0.3, 0.8, 0.3]    | [red, green, blue] first color of constraint, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.color_body2              | vector   |   | [0.7, 0.8, 0.3]    | [red, green, blue] second color of constraint, range = 0..1, use default color:[-1,-1,-1] |
| Graphics.standard_joint_drawing  | bool     |   | 1                  | flag for drawing mode; 1 == draw constraint element; 0 == show constrained directions and rotations; |
| Graphics.joint_cylinder_size      | vector   |   | [-1, -1]           | cylinder dimension of cylindrical joint (for drawing); [lx (cyl. length, in sl. dir.),d (cylinder diameter)] |
| Graphics.axis_length              | double   |   | -1                 | axis length of the revolute joint (for drawing)                            |

#### Physics

| Physics.use_penalty_formulation   | bool     |   | 0                  | 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |
| Physics.Penalty.damping          | double   |   | 100                | damping parameter used for translation and rotation                        |
| Physics.Penalty.stiffness        | double   |   | 1e+006             | stiffness parameter used for translation and rotation                       |

#### Physics.Lagrange

| Physics.Lagrange.constrained_directions | vector   | R | [0, 1, 1] | constrained directions cannot be changed                                   |
| Physics.Lagrange.constrained_rotations | vector   | R | [0, 1, 1] | constrained rotations cannot be changed                                    |
| Physics.rotation_sliding_axis      | vector   |   | [1, 0, 0] | local rotation/sliding axis w.r.t body 1 coordinate system                 |
### 3.3. CONNECTOR

<table>
<thead>
<tr>
<th>Position 1</th>
<th>element_number</th>
<th>integer</th>
<th>1</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 1</td>
<td>position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local position. Only used if node_number == 0!</td>
</tr>
</tbody>
</table>

**Position 2**

<table>
<thead>
<tr>
<th>Position 2</th>
<th>element_number</th>
<th>integer</th>
<th>0</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 2</td>
<td>position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local or global (if element_number == 0) position. Only used if node_number == 0!</td>
</tr>
</tbody>
</table>

#### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-4</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-4</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td>Connector.moment</td>
<td>internal global moment of connector</td>
</tr>
<tr>
<td>Connector.force_local</td>
<td>internal local force of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.moment_local</td>
<td>internal local moment of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.displacement</td>
<td>displacement between the joint coordinate systems JAi and JAj expressed in coordinate system JAi</td>
</tr>
<tr>
<td>Connector.angle</td>
<td>bryant angles between the joint coordinate systems JAi and JAj. All constrained components are zero.</td>
</tr>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>

#### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
</tbody>
</table>
Example

see file CylindricalJointShort.txt

l = 1 % m

force
{
    load_type = "ForceVector3D"
    force_vector = [10,10,10]
}
nForce = AddLoad(force)

rigidBody
{
    element_type= "Rigid3D"
    loads= [nForce]
    Graphics.body_dimensions= [1, 0.05, 0.05]
}
nRigid = AddElement(rigidBody)

cylindricalJoint
{
    element_type= "CylindricalJoint"
    Physics.rotation_sliding_axis = [1,0,0]
    Position1
    {
        element_number= nRigid %number of constrained element
        position= [-l/2, 0, 0] %local position
    }
    Position2.position= [-l/2, 0, 0]
}
AddConnector(cylindricalJoint)

3.3.17 SpringDamper Actuator

Short description

The Spring-Damper-Actuator connects two points with a spring, a damper and a actor element, in which actuator force fa remains constant. The resultant force is applied in the connection line of these points. There are different modes available, how the spring and damper force is calculated. It is also possible to change the neutral spring length. This joint is realized in Penalty formulation only.

Equations

point positions: \( \mathbf{p}^{(1)} = \begin{bmatrix} p_x^{(1)} & p_y^{(1)} & p_z^{(1)} \end{bmatrix}^T \); \( \mathbf{p}^{(2)} = \begin{bmatrix} p_x^{(2)} & p_y^{(2)} & p_z^{(2)} \end{bmatrix}^T \).

point velocities: \( \dot{\mathbf{p}}^{(1)} = \begin{bmatrix} \dot{p}_x^{(1)} & \dot{p}_y^{(1)} & \dot{p}_z^{(1)} \end{bmatrix}^T \); \( \dot{\mathbf{p}}^{(2)} = \begin{bmatrix} \dot{p}_x^{(2)} & \dot{p}_y^{(2)} & \dot{p}_z^{(2)} \end{bmatrix}^T \).

spring length: \( l_0 \)
3.3. CONNECTOR

direction vector: \( \text{dir} = \frac{\mathbf{p}^{(1)} - \mathbf{p}^{(2)}}{\sqrt{\left(p^{(1)}_x - p^{(2)}_x\right)^2 + \left(p^{(1)}_y - p^{(2)}_y\right)^2 + \left(p^{(1)}_z - p^{(2)}_z\right)^2}} \)

spring elongation: \( \Delta x = l - l_0 = (\mathbf{p}^{(1)} - \mathbf{p}^{(2)})^T \text{dir} - l_0 \)

spring velocity: \( v = (\dot{\mathbf{p}}^{(1)} - \dot{\mathbf{p}}^{(2)})^T \text{dir} \)

resultant force (see section forcemode):
forcemode 0: \( f = k \Delta x + d v + f_a \) (a)
forcemode 1: \( f = k (\Delta x) \Delta x + d (v) v + f_a \) (b)
forcemode 2: \( f = f_k + f_d + f_a \) (c)
forcemode 3: \( f = f_k (\Delta x) + f_d (v) + f_a \) (d)

Limitations

If the 2 end points of the spring are the same point in the initial configuration, this may lead to problems! The direction of the spring can not be determined in that case!

Description of the different modi

<table>
<thead>
<tr>
<th>element to ground</th>
<th>Position2.element_number AND Position2.node_number have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>element to element</td>
<td>Position2.element_number AND/or Position2.node_number must not be equal to 0</td>
</tr>
<tr>
<td>forcemode</td>
<td><strong>Physics.forcemode</strong> = 0: Force is computed as (a) with constant stiffness and damping factors ( k ) and ( d ). The factors can be defined in the two fields in Physics.Linear. <strong>Physics.forcemode</strong> = 1: 2 MathFunctions are used to describe piecewise linear stiffness ( k (\Delta x) ) and damping ( d (v) ), see formula (b) and Physics.MathFunction. <strong>Physics.forcemode</strong> = 2: 2 IOElementDataModifiers describe the force (c) due to stiffness and damping. You should use this mode if full nonlinear behavior is required, e.g. ( f_k = f_k (t, l, v, ...) ) and ( f_d = d (t, l, v, ...) ). <strong>Physics.forcemode</strong> = 3: 2 MathFunctions are used to describe piecewise linear spring force ( f_k (\Delta x) ) and damping force ( f_d (v) ), see formula (d) and Physics.MathFunction.</td>
</tr>
<tr>
<td></td>
<td>modifier value names for forcemode == 2: ( f_k: \text{Connector.SpringDamperActuator.spring_force} ) ( f_d: \text{Connector.SpringDamperActuator.damper_force} )</td>
</tr>
<tr>
<td>spring length offset</td>
<td>It is possible to change the spring length ( l_0 ) (neutral length of the spring) during the simulation, e.g. for the usage of the SpringDamperActuator as a linear actuator. In standard mode the value in the field Physics.spring_length remains constant. This value can be modified by a IOElementDataModifier via 'Connector.SpringDamperActuator.spring_length_offset'.</td>
</tr>
<tr>
<td>additional actor force</td>
<td>In Physics.actor_force a constant offset force ( f_a ) can be added.</td>
</tr>
</tbody>
</table>
Figure 3.44: SpringDamperActuator

**Data objects of SpringDamperActuator:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;SpringDamperActuator&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;SpringDamperActuator&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td></td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] first color of constraint (spring), range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td></td>
<td>[0.7, 0.8, 0.3]</td>
<td>[red, green, blue] second color of constraint (damper), range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.spring_diameter</td>
<td>double</td>
<td>R</td>
<td>-1</td>
<td>spring diameter used for drawing only.</td>
</tr>
<tr>
<td>Graphics.spring_coils</td>
<td>double</td>
<td></td>
<td>10</td>
<td>spring coils used for drawing. If set to 0, then a cylinder with the value 'spring_diameter' as diameter is shown instead of the coils.</td>
</tr>
<tr>
<td>Graphics.damper_diameter</td>
<td>double</td>
<td></td>
<td>-1</td>
<td>damper diameter used for drawing only. If set to 0, then the damper is not shown. It's recommended to choose the value smaller then the spring diameter.</td>
</tr>
</tbody>
</table>

**Physics**

<table>
<thead>
<tr>
<th>Physics</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.spring_length</td>
<td>double</td>
<td></td>
<td>0</td>
<td>length of the spring in the initial configuration</td>
</tr>
<tr>
<td>Physics.actor_force</td>
<td>double</td>
<td></td>
<td>0</td>
<td>constant force acting on the spring</td>
</tr>
<tr>
<td>Physics.forcemode</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>defines how the spring and damper force is computed: 0..constant coefficient, 1..MathFunction (stiffness and damping), 2..spring and damper force prescribed by IOElementDataModifier, 3..MathFunction (spring force and damping force)</td>
</tr>
</tbody>
</table>

**Physics.Linear**

<table>
<thead>
<tr>
<th>Physics.Linear</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Linear.spring_stiffness</td>
<td>double</td>
<td></td>
<td>100</td>
<td>stiffness coefficient of the linear spring. Only used if forcemode is 0.</td>
</tr>
<tr>
<td>Physics.Linear.damping</td>
<td>double</td>
<td></td>
<td>1</td>
<td>damping coefficient for viscous damping. Only used if forcemode is 0.</td>
</tr>
</tbody>
</table>

**Physics.MathFunction**

<table>
<thead>
<tr>
<th>Physics)MathFunction</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physics.MathFunction</strong>. <strong>MathFunction_k</strong>. piecewise_mode</td>
<td>integer</td>
<td>-1</td>
<td>modulus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
<td></td>
</tr>
<tr>
<td><strong>Physics.MathFunction</strong>. <strong>MathFunction_k</strong>. piecewise_points</td>
<td>vector</td>
<td>[]</td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
<td></td>
</tr>
<tr>
<td><strong>Physics.MathFunction</strong>. <strong>MathFunction_k</strong>. piecewise_values</td>
<td>vector</td>
<td>[]</td>
<td>values at supporting points</td>
<td></td>
</tr>
<tr>
<td><strong>Physics.MathFunction</strong>. <strong>MathFunction_k</strong>. piecewise_diff_values</td>
<td>vector</td>
<td>[]</td>
<td>differential values at supporting points - for quadratic interpolation</td>
<td></td>
</tr>
</tbody>
</table>
| **Physics.MathFunction**. **MathFunction_k**. parsed_function | string | "" | string representing parsed function, e.g. 'A*sin(omega*t)'
| **Physics.MathFunction**. **MathFunction_k**. parsed_function_parameter | string | "" | string representing parameter of parsed function, e.g. 't' |

**Physics.MathFunction**. **MathFunction_d**

| **Physics.MathFunction**. **MathFunction_d**. piecewise_mode | integer | -1 | modulus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic |
| **Physics.MathFunction**. **MathFunction_d**. piecewise_points | vector | [] | supporting points (e.g. time or place) for piecewise interpolation |
| **Physics.MathFunction**. **MathFunction_d**. piecewise_values | vector | [] | values at supporting points |
| **Physics.MathFunction**. **MathFunction_d**. piecewise_diff_values | vector | [] | differential values at supporting points - for quadratic interpolation |
| **Physics.MathFunction**. **MathFunction_d**. parsed_function | string | "" | string representing parsed function, e.g. 'A*sin(omega*t)'
| **Physics.MathFunction**. **MathFunction_d**. parsed_function_parameter | string | "" | string representing parameter of parsed function, e.g. 't' |

**Physics.MathFunction**. **MathFunction_fk**

| **Physics.MathFunction**. **MathFunction_fk**. piecewise_mode | integer | -1 | modulus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic |
| **Physics.MathFunction**. **MathFunction_fk**. piecewise_points | vector | [] | supporting points (e.g. time or place) for piecewise interpolation |
| **Physics.MathFunction**. **MathFunction_fk**. piecewise_values | vector | [] | values at supporting points |
| **Physics.MathFunction**. **MathFunction_fk**. piecewise_diff_values | vector | [] | differential values at supporting points - for quadratic interpolation |
| **Physics.MathFunction**. **MathFunction_fk**. parsed_function | string | "" | string representing parsed function, e.g. 'A*sin(omega*t)'
<p>| <strong>Physics.MathFunction</strong>. <strong>MathFunction_fk</strong>. parsed_function_parameter | string | &quot;&quot; | string representing parameter of parsed function, e.g. 't' |</p>
<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
<th><strong>MathFunction_fd.</strong>&lt;br&gt;<strong>piecewise_mode</strong></th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physics.MathFunction.</strong></td>
<td><strong>MathFunction_fd.</strong>&lt;br&gt;<strong>piecewise_points</strong></td>
<td>vector</td>
<td>[ ]</td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
<tr>
<td><strong>Physics.MathFunction.</strong></td>
<td><strong>MathFunction_fd.</strong>&lt;br&gt;<strong>piecewise_values</strong></td>
<td>vector</td>
<td>[ ]</td>
<td>values at supporting points</td>
</tr>
<tr>
<td><strong>Physics.MathFunction.</strong></td>
<td><strong>MathFunction_fd.</strong>&lt;br&gt;<strong>piecewise_diff_values</strong></td>
<td>vector</td>
<td>[ ]</td>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td><strong>Physics.MathFunction.</strong></td>
<td><strong>MathFunction_fd.</strong>&lt;br&gt;<strong>parsed_function</strong></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td><strong>Physics.MathFunction.</strong></td>
<td><strong>MathFunction_fd.parsed_function_parameter</strong></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
</tbody>
</table>

**Position 1**

<table>
<thead>
<tr>
<th><strong>Position1.</strong></th>
<th><strong>element_number</strong></th>
<th>integer</th>
<th>1</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Position1.position</strong></td>
<td></td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local position. Only used if node_number == 0!</td>
</tr>
<tr>
<td><strong>Position1.node_number</strong></td>
<td></td>
<td>integer</td>
<td>0</td>
<td>local or global (if element_number == 0) node number.</td>
</tr>
</tbody>
</table>

**Position 2**

<table>
<thead>
<tr>
<th><strong>Position2.</strong></th>
<th><strong>element_number</strong></th>
<th>integer</th>
<th>0</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Position2.position</strong></td>
<td></td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local or global (if element_number == 0) position. Only used if node_number == 0!</td>
</tr>
<tr>
<td><strong>Position2.node_number</strong></td>
<td></td>
<td>integer</td>
<td>0</td>
<td>local or global (if element_number == 0) node number.</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1](#)

<table>
<thead>
<tr>
<th><strong>value name</strong></th>
<th><strong>description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-14</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Internal.second_order_variable.velocity</td>
<td>velocities of second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.force</td>
<td>internal resultant force of connector</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_length</td>
<td>actual spring length</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_elongation</td>
<td>elongation of spring</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_velocity</td>
<td>spring velocity</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

Controllable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.SpringDamperActuator.spring_length_offset</td>
<td>prescribe the neutral spring length</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_force</td>
<td>prescribe the stiffness force</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.damper_force</td>
<td>prescribe the damping force</td>
</tr>
</tbody>
</table>

Example
see file SpringDamperActuator.txt

\[ l = 0.5 \, \text{m} \]
\[ m = 10 \, \text{kg} \]
\[ g = 9.81 \, \text{m/s}^2 \]

```
gravLoad
{
  load_type = "Gravity"
  direction = 3 \, \text{z - direction}
  gravity_constant = g
}
nLoad = AddLoad(gravLoad)

mass
{
  element_type = "Mass3D" \, \text{specification of element type.}
  loads = [nLoad]
  Initialization.initial_position = [0, 0, l] \, \text{initial position}
  Physics.mass = m \, \text{total mass}
}
nMass = AddElement(mass)

springDamperActuator
{
  element_type = "SpringDamperActuator"
  Physics.forcemode = 2 \, \text{nonlinear spring}
  Position1.element_number = nMass \, \text{number of constrained element}
  Position2.element_number = 0 \, \text{number of constrained element}
}
nSpringDamperActuator = AddConnector(springDamperActuator)

disp
{
  sensor_type = "FVElementSensor"
  element_number = nMass
  field_variable = "displacement"
  component = "z"
}
nDisp = AddSensor(disp)
```
nonlinearStiffnessForce
{
    element_type = "IOMathFunction"
    Graphics
    {
        position = [0, 0] %reference drawing position
        draw_size = [20, 20, 0] %draw size
    }
    IOBlock
    {
        input_element_numbers = [nDisp] %element connected to input
        input_element_types = [2] %2=Sensor
        input_local_number = [1]
        MathFunction
        {
            piecewise_mode = 1 %modus for piecewise interpolation: 1=linear
            piecewise_points = [-0.3,-0.2,-0.15,0,0.15,0.2,0.3] %m, supporting points
            piecewise_values = [-5000,-300,-30,0,30,300,5000] %N, values at s. p.
        }
    }
}
nNonlinearStiffnessForce = AddElement(nonlinearStiffnessForce)

modifier_SDA
{
    element_type = "IOElementDataModifier"
    Graphics
    {
        position = [30, 0] %reference drawing position
        draw_size = [20, 20, 0] %draw size
    }
    IOBlock
    {
        input_element_numbers = [nNonlinearStiffnessForce] %element connected to input
        input_element_types = [1]
        input_local_number = [1]
        mod_variable_name = "Connector.SpringDamperActuator.spring_force" %modified element data
        mod_element_number = nSpringDamperActuator %modified constraint
    }
}
AddElement(modifier_SDA)

3.3.18 RigidLink

Short description

A rigid link is a rigid constraint element that provides a stiff connection between nodes or positions in the model. In standard mode the distance between the connected points remains constant. In extended mode it is possible to change the distance as a function of time or input. There is only a Lagrange formulation implemented.
3.3. CONNECTOR

Equations

point positions: \( \mathbf{p}^{(1)} = \begin{bmatrix} p_{x}^{(1)} & p_{y}^{(1)} & p_{z}^{(1)} \end{bmatrix}^T \); \( \mathbf{p}^{(2)} = \begin{bmatrix} p_{x}^{(2)} & p_{y}^{(2)} & p_{z}^{(2)} \end{bmatrix}^T \).

point velocities: \( \dot{\mathbf{p}}^{(1)} = \begin{bmatrix} \dot{p}_{x}^{(1)} & \dot{p}_{y}^{(1)} & \dot{p}_{z}^{(1)} \end{bmatrix}^T \); \( \dot{\mathbf{p}}^{(2)} = \begin{bmatrix} \dot{p}_{x}^{(2)} & \dot{p}_{y}^{(2)} & \dot{p}_{z}^{(2)} \end{bmatrix}^T \).

link length: \( l_0 \)

time derivative of link length: \( v \) (equates \( l_0 \))

direction vector: \( \mathbf{dir} = \frac{\mathbf{p}^{(1)} - \mathbf{p}^{(2)}}{\sqrt{(p_{x}^{(1)} - p_{x}^{(2)})^2 + (p_{y}^{(1)} - p_{y}^{(2)})^2 + (p_{z}^{(1)} - p_{z}^{(2)})^2}} \)

position constraint: \( \mathbf{C} = (\mathbf{p}^{(1)} - \mathbf{p}^{(2)})^T \mathbf{dir} - l_0 = 0 \) (a)

velocity constraint: \( \dot{\mathbf{C}} = (\dot{\mathbf{p}}^{(1)} - \dot{\mathbf{p}}^{(2)})^T \mathbf{dir} - v = 0 \) (b)

\( \frac{\partial \mathbf{C}}{\partial \mathbf{q}} = (\frac{\partial \mathbf{p}^{(1)}}{\partial \mathbf{q}} - \frac{\partial \mathbf{p}^{(2)}}{\partial \mathbf{q}})^T \mathbf{dir} \)

Limitations

For a position constraint (index 3 solver) with variable distance it is necessary to define the link length \( l_0 \) as a function of time. In this case the velocity input \( v \) (the derivative of the distance with respect to time) is not considered, see formula (a). Reverse conditions apply to the velocity constraint with formula (b).

Description of the different modi

<table>
<thead>
<tr>
<th>element to ground</th>
<th>Position2.element_number AND Position2.node_number have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>element to element</td>
<td>Position2.element_number and/or Position2.node_number must not be equal to 0</td>
</tr>
<tr>
<td>distance mode</td>
<td><strong>Physics.distance_mode = 0:</strong> The distance remains constant. The value can be defined in the field Physics.Constant.link_length. <strong>Physics.distance_mode = 1:</strong> A MathFunction is used to describe piecewise linear distance or velocity development over time ( t ), e.g. for a rigid link actuator. See Physics.MathFunction. <strong>Physics.distance_mode = 2:</strong> A IOElementDataModifier describes the developing distance or velocity over time ( t ), e.g. for a rigid link actuator. See section limitations.</td>
</tr>
</tbody>
</table>

Figure 3.45: RigidLink
### Data objects of RigidLink:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;RigidLink&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;RigidLink&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] first color of constraint, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3]</td>
<td>[red, green, blue] second color of constraint, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.cylinder1_diameter</td>
<td>double</td>
<td>-1</td>
<td>cylinder one diameter (drawing only).</td>
</tr>
<tr>
<td>Graphics.cylinder2_diameter</td>
<td>double</td>
<td>0</td>
<td>cylinder two diameter (drawing only). Only used if distance not constant = distance mode 1 or 2.</td>
</tr>
<tr>
<td>Graphics.cylinder1_length</td>
<td>double</td>
<td>0</td>
<td>cylinder one length (drawing only). Only used if distance not constant = distance mode 1 or 2.</td>
</tr>
</tbody>
</table>

#### Physics

<table>
<thead>
<tr>
<th>Physics.distance_mode</th>
<th>integer</th>
<th>0</th>
<th>defines the distance: 0. constant distance, 1.MathFunction, 2.IOElementDataModifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Constant</td>
<td>double</td>
<td>0</td>
<td>constant distance is used, when distance_mode = 0</td>
</tr>
</tbody>
</table>

#### Physics.MathFunction

### Physics.MathFunction.MathFunction_1

<table>
<thead>
<tr>
<th>Physics.MathFunction.MathFunction_1.piecewise_mode</th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.MathFunction.MathFunction_1.piecewise_points</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_1.piecewise_values</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_1.piecewise_diff_values</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_1.parsed_function</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_1.parsed_function_parameter</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
</tbody>
</table>

### Physics.MathFunction.MathFunction_v

<table>
<thead>
<tr>
<th>Physics.MathFunction.MathFunction_v.piecewise_mode</th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.MathFunction.MathFunction_v.piecewise_points</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_v.piecewise_values</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_v.piecewise_diff_values</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Position 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>1</td>
<td>Number of constrained element</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Local position. Only used if node_number == 0!</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>0</td>
<td>Local or global (if element_number == 0) node number.</td>
</tr>
</tbody>
</table>

### Position 2

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_number</td>
<td>integer</td>
<td>0</td>
<td>Number of constrained element</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Local or global (if element_number == 0) position. Only used if node_number == 0!</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>0</td>
<td>Local or global (if element_number == 0) node number.</td>
</tr>
</tbody>
</table>

### Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>Degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>Algebraic variables of the element. range: 1-1</td>
</tr>
</tbody>
</table>

### Controllable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.RigidLink.link_length</td>
<td>Distance between the connected points (l0)</td>
</tr>
<tr>
<td>Connector.RigidLink.link_velocity</td>
<td>Derivative of the distance with respect to time (v)</td>
</tr>
</tbody>
</table>

### Example
see file RigidLink.txt

```plaintext
l = 0.5 \% m
m = 10 \% kg
v = 0.1 \% m/s
g = 9.81 \% m/s^2
```

gravLoad
{

load_type = "Gravity"
direction = 3 \% z - direction
gravity_constant = g
}

nLoad = AddLoad(gravLoad)

mass
{
    element_type = "Mass3D"
    loads = [nLoad]
    Initialization.initial_position = [1, 0, 0] \%initial position
    Physics.mass = m \%total mass of point mass
}

nMass = AddElement(mass)

%link
rigidLink
{
    element_type = "RigidLink"
    Physics.distancemode = 2 \% link length by modifier
    Graphics
    {
        show_connector = 1
        cylinder1_diameter = 0.1
        cylinder2_diameter = 0.08
        cylinder1_length = 1/2
    }
    Position1.element_number = nMass \%number of constrained element
    Position2.element_number = 0 \%number of constrained element
}

nRigidLink = AddConnector(rigidLink)

time
{
    element_type = "IOTime"
    Graphics
    {
        position = [-30, 0] \%reference drawing position
        draw_size = [20, 20, 0] \%draw size
    }
}

nTime = AddElement(time)

vel
{
    element_type = "IOMathFunction"
    IOBlock
    {
        input_element_numbers = [nTime] \%element connected to input
        input_element_types = [1] \%1=IOElement
        input_local_number = [1] \%i-th number of output of previous IOelement
        MathFunction
3.3. CONNECTOR

```{ 
    piecewise_mode = 1 %modus for piecewise interpolation: 1=linear
    piecewise_points = [0,1,2,3] %supporting points
    piecewise_values = [1,1,2*1,2*1] %values at supporting points
}
}
nVel = AddElement(vel)
```

type modifier {
    element_type = "IOElementDataModifier"
    Graphics {
        position = [30, 0] %reference drawing position
        draw_size = [20, 20, 0] %draw size
    }
    IOBlock {
        input_element_numbers = [nVel] %element connected to input
        input_element_types = [1] %1=IOElement
        input_local_number = [1] %i-th number of output connected to this element
        mod_variable_name = "Connector.RigidLink.link_length" %variable name
        mod_element_number = nRigidLink %element number
    }
}
AddElement(modifier)

3.3.19 RotatorySpringDamperActuator

**Short description**

The RotatorySpringDamperActuator connects two elements with rotatory spring, damper and a constant actuator moment \( m_a \). Positive rotation around rotation axis according to right hand rule. There are different modes available, how the spring and damper moment is calculated. It is also possible to change the neutral spring angle. This joint is realized in Penalty formulation only.

**Equations**

- spring angular deflection \( \Delta \phi = \phi - \phi_0 \)
- spring angular velocity \( \omega \)

resultant moment (see section forcemode):
forcemode 0: \( m = k \Delta \phi + d \omega + m_a \) (a)
forcemode 1: \( m = k (\Delta \phi) \Delta \phi + d (\omega) \omega + m_a \) (b)
forcemode 2: \( m = m_k + m_d + m_a \) (c)

**Limitations**

The RotatorySpringDamperActuator should be used together with a RevoluteJoint to avoid useless simulation results. It is important to ensure that the relative angle of rotation between
the two bodies must never be greater than \( \pm \pi \). This has to be taken into account when using an offset angle \( \phi_0 \).

**Description of the different modi**

<table>
<thead>
<tr>
<th>element to ground</th>
<th>Position2.elemen_number AND Position2.node_number have to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>element to element</td>
<td>Position2.elemen_number and/or Position2.node_number must not be equal to 0</td>
</tr>
</tbody>
</table>
| forcemode         | **Physics.forcemode** = 0:  

Moment is computed as (a) with constant stiffness and damping factors \( k \) and \( d \). The factors can be defined in the two fields in Physics.Linear.  

**Physics.forcemode** = 1:  

A MathFunction is used to describe piecewise linear stiffness \( k(\Delta \phi) \) and damping \( d(\omega) \), see formula (b) and Physics.MathFunction.  

**Physics.forcemode** = 2:  

2 IOElementDataModifiers describe the moment (c) due to stiffness and damping. You should use this mode if full nonlinear behavior is required, e.g. \( m_k = m_k(t, \phi, \omega, ...) \) and \( m_d = d(t, \phi, \omega, ...) \).

| spring angle offset | It is possible to change the spring angle \( \phi_0 \) (neutral angle of the spring) during the simulation, e.g. for the usage of the RotatorySpringDamperActuator as a rotational actuator. In standard mode the offset remains constant. The value can be defined in the field Physics.spring_angle_offset. This offset can be modified by a IOElementDataModifier via 'Connector.RotatorySpringDamperActuator.angle_offset'. |
| additional actuator moment | In Physics.actuator_torque a constant offset moment \( m_a \) can be added. |

---

**Data objects of RotatorySpringDamperActuator:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;RotatorySpringDamperActuator&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
</tbody>
</table>
### 3.3. CONNECTOR

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name_of_the_element</td>
<td>string</td>
<td>&quot;RotatorySpringDamperActuator&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 2</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.color_body1</td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] first color of constraint, range 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.color_body2</td>
<td>vector</td>
<td>[0.7, 0.8, 0.3]</td>
<td>[red, green, blue] second color of constraint, range 0..1, use default color: [-1,-1,-1]</td>
</tr>
<tr>
<td>Graphics.spring_size</td>
<td>double</td>
<td>-1</td>
<td>radius of torsional spring. This parameter is used for drawing only.</td>
</tr>
<tr>
<td>Graphics.windings</td>
<td>double</td>
<td>10</td>
<td>number of windings of torsional spring. This parameter is used for drawing only.</td>
</tr>
<tr>
<td>Graphics.axis_radius</td>
<td>double</td>
<td>-1</td>
<td>radius of torsional spring axis (cylinder). This parameter is used for drawing only.</td>
</tr>
</tbody>
</table>

#### Physics

<table>
<thead>
<tr>
<th>Physics.spring_angle_offset</th>
<th>double</th>
<th>0</th>
<th>spring angle offset is used if constant spring_angle_offset is enabled. A positive offset equates a positive angle about the rotation axis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.actuator_torque</td>
<td>double</td>
<td>0</td>
<td>constant torque of an actuator. A positive torque is acting about the rotation axis in a positive sense.</td>
</tr>
<tr>
<td>Physics.rotation_axis</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>local axis of rotation w.r.t. body 1 coordinate system in initial configuration</td>
</tr>
<tr>
<td>Physics.forcemode</td>
<td>integer</td>
<td>0</td>
<td>defines how the spring and damper moment is computed: 0.constant coefficient, 1.MathFunction, 2.IOElementDataModifier</td>
</tr>
</tbody>
</table>

#### Physics.Linear

<table>
<thead>
<tr>
<th>Physics.Linear.spring_stiffness</th>
<th>double</th>
<th>100</th>
<th>stiffness parameter of the rotatory spring. Only used if forcemode is 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Linear.damping</td>
<td>double</td>
<td>1</td>
<td>damping coefficient for viscous damping. Only used if forcemode is 0.</td>
</tr>
</tbody>
</table>

#### Physics.MathFunction

<table>
<thead>
<tr>
<th>Physics.MathFunction.MathFunction_k.piecewise_mode</th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.MathFunction.MathFunction_k.piecewise_points</td>
<td>vector</td>
<td>[]</td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_k.piecewise_values</td>
<td>vector</td>
<td>[]</td>
<td>values at supporting points</td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_k.piecewise_diff_values</td>
<td>vector</td>
<td>[]</td>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_k.parsed_function</td>
<td>string</td>
<td>&quot;A<em>sin(omega</em>t)&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>Physics.MathFunction.MathFunction_k.parsed_function_parameter</td>
<td>string</td>
<td>&quot;A&quot;</td>
<td>string representing parameter of parsed function, e.g. 'A'</td>
</tr>
</tbody>
</table>

#### Physics.MathFunction.MathFunction_d

<table>
<thead>
<tr>
<th>Physics.MathFunction.MathFunction_d.piecewise_mode</th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
</table>
### Physics.MathFunction.
### MathFunction_d.
**piecewise_points**
- vector
- ||
  - supporting points (e.g., time or place) for piecewise interpolation
**piecewise_values**
- vector
- ||
  - values at supporting points
**piecewise_diff_values**
- vector
- ||
  - differential values at supporting points - for quadratic interpolation
**parsed_function**
- string
- ||
  - string representing parsed function, e.g., 'A*sin(omega*t)'
**parsed_function_parameter**
- string
- ||
  - string representing parameter of parsed function, e.g., 't'

### Position1
- **element_number**
  - integer
  - 1
  - Number of constrained element
- **position**
  - vector
  - [0, 0, 0]
  - local position. Only used if node_number == 0!

### Position2
- **element_number**
  - integer
  - 0
  - Number of constrained element
- **position**
  - vector
  - [0, 0, 0]
  - local or global (if element_number == 0) position. Only used if node_number == 0!

### Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-14</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-3, corresponds to force in global x-y-z direction</td>
</tr>
<tr>
<td>Connector.moment</td>
<td>internal global moment of connector</td>
</tr>
<tr>
<td>Connector.force_local</td>
<td>internal local force of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.moment_local</td>
<td>internal local moment of connector (joint coordinate system JAi)</td>
</tr>
<tr>
<td>Connector.displacement</td>
<td>displacement between the joint coordinate systems JAi and JAj expressed in coordinate system JAI</td>
</tr>
<tr>
<td>Connector.angle</td>
<td>bryant angles between the joint coordinate systems JAi and JAj. All constrained components are zero.</td>
</tr>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
<tr>
<td>Connector.RotatorySpringDamperActuator.moment</td>
<td>internal moment of connector</td>
</tr>
</tbody>
</table>
3.3. CONNECTOR

Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.stiffness_matrix</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>Connector.damping_matrix</td>
<td>damping matrix</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_rotation</td>
<td>stiffness matrix for rotation</td>
</tr>
<tr>
<td>Connector.stiffness_matrix_damping</td>
<td>damping matrix for rotation</td>
</tr>
<tr>
<td>Connector.local_position_1</td>
<td>local position on element 1</td>
</tr>
<tr>
<td>Connector.local_position_2</td>
<td>local position on element 2</td>
</tr>
<tr>
<td>Connector.RotatorySpringDamperActuator.angle_offset</td>
<td>prescribe the angle offset</td>
</tr>
<tr>
<td>Connector.RotatorySpringDamperActuator.spring_moment</td>
<td>prescribe the stiffness moment</td>
</tr>
<tr>
<td>Connector.RotatorySpringDamperActuator.damp_moment</td>
<td>prescribe the damping moment</td>
</tr>
</tbody>
</table>

Example

see file RotationalSpringDamperActuator.txt

\[
\begin{align*}
l &= 0.5 \text{ m} \\
r &= 0.05 \\
m &= 10 \text{ kg} \\
I_x &= m*r^2/2 \text{ kg*m}^2 \\
I_y &= m*l^2/3 \text{ kg*m}^2 \\
I_z &= I_y \\
\end{align*}
\]

force

\{
  \text{load_type} = "\text{ForceVector3D}" \\
  \text{force_vector} = [0,-100,0] \\
  \text{position} = [l/2,0,0] \\
  \text{local_force} = 1 \\
\}

nForce = AddLoad(force)

body

\{
  \text{element_type} = "\text{Rigid3D}" \\
  \text{loads} = [nForce] \\
  \text{Physics} \\
  \{
    \text{mass} = m \\
    \text{moment_of_inertia} = [I_x,0.,0.;I_y,0.;0.,I_z] \\
  \}
  \text{Graphics} \\
  \{
    \text{RGB\_color} = [0.,0.,1.] \text{ [red, green, blue]} \\
    \text{show\_element} = 1 \text{ [Flag to draw element]} \\
    \text{body\_dimensions} = [l,r,r] \\
  \}
\}
CHAPTER 3. HOTINT REFERENCE MANUAL

```plaintext
Initialization.initial_position = [l/2, 0, 0]
nBody = AddElement(body)
revoluteJoint
{
    element_type = "RevoluteJoint"
    Physics.rotation_axis = [0, 0, 1]
    Graphics.show_connector = 0

    Position1.element_number = nBody %number of constrained element
    Position1.position = [-l/2, 0, 0] %local position
    Position2.element_number = 0 %number of constrained element
}
nRevoluteJoint = AddConnector(revoluteJoint)
rotSpringDamperActuator
{
    element_type = "RotatorySpringDamperActuator"
    Physics
    {
        forcemode = 2 % force by nonlinear spring
        rotation_axis = [0, 0, 1]
    }
    Graphics
    {
        show_connector = 1
    }

    Position1.element_number = nBody %number of constrained element
    Position1.position = [-l/2, 0, 0] %local position
    Position2.element_number = 0 %number of constrained element
}
nRotSpringDamperActuator = AddConnector(rotSpringDamperActuator)
phi
{
    sensor_type = "ElementSensor"
    element_number = nRevoluteJoint
    value = "Connector.angle[1]" %about x-axis (joint coordinate system)
}
phi = AddSensor(phi)
nonlinearStiffnessMoment
{
    element_type = "IOMathFunction"
    Graphics
    {
        position = [0, 0] %reference drawing position
        draw_size = [20, 20, 0] %draw size
    }
    IGBlock
```
3.3. **CONNECTOR**

```{math}
\begin{align*}
\text{input_element_numbers} &= [\text{nPhi}] \quad \% \text{element connected to input} \\
\text{input_element_types} &= [2] \quad \% 2 = \text{Sensor} \\
\text{input_local_number} &= [1] \quad \% i\text{-th number of output} \\
\text{MathFunction}
\begin{align*}
\text{piecewise_mode} &= 1 \quad \% 1 = \text{linear} \\
\text{piecewise_points} &= [-1.2, -0.8, -0.5, 0, 0.5, 0.8, 1.2] \quad \% m, \text{supporting points} \\
\text{piecewise_values} &= [200, 100, 10, 0, -10, -100, -200] \quad \% N, \text{values at s. p.}
\end{align*}
\end{align*}
```

\text{nNonlinearStiffnessMoment} = \text{AddElement} (\text{nonlinearStiffnessMoment})

\text{modifier}

```{math}
\begin{align*}
\text{element_type} &= \"\text{IOElementDataModifier}\" \\
\text{Graphics}
\begin{align*}
\text{position} &= [30, 0] \quad \% \text{reference drawing position} \\
\text{draw_size} &= [20, 20, 0] \quad \% \text{draw size}
\end{align*}
\end{align*}
```

\text{IOBlock}

```{math}
\begin{align*}
\text{input_element_numbers} &= [\text{nNonlinearStiffnessMoment}] \quad \% \text{element connected to input} \\
\text{input_element_types} &= [1] \quad \% 1 = \text{IOElement} \\
\text{input_local_number} &= [1] \quad \% i\text{-th number of output} \\
\text{mod_variable_name} &= \"\text{Connector.RotatorySpringDamperActuator.spring_moment}\" \quad \% \text{variable name} \\
\text{mod_element_number} &= \text{nRotSpringDamperActuator} \quad \% \text{element number}
\end{align*}
```

\text{AddElement} (\text{modifier})

### 3.3.20 **SpringDamperActuator2D**

**Short description**

The SpringDamperActuator2D is a simplified version of the SpringDamperActuator for 2D elements. Nodes are not supported in the 2D version. Apart from that the constraint has the same functionality as the 3D version. See the SpringDamperActuator documentation for more information.

**Description of the different modi**

<table>
<thead>
<tr>
<th>Element to Ground</th>
<th>Position2.element_number has to be equal to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element to Element</td>
<td>Position2.element_number must not be equal to 0</td>
</tr>
<tr>
<td>Lagrange</td>
<td>If Physics.use_penalty_formulation = 0, then no stiffness and no damping parameters are used.</td>
</tr>
</tbody>
</table>

**Data objects of SpringDamperActuator2D:**
### Data Name | Type | R | Description
---|---|---|---
`element_type` | string | "SpringDamperActuator2D" | Specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!
`name` | string | "SpringDamperActuator2D" | Name of the element
`element_number` | integer | R | 2 | Number of the element in the mbs

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Graphics.show_connector</code></td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td><code>Graphics.color_body1</code></td>
<td>vector</td>
<td>[0.3, 0.8, 0.3]</td>
<td>First color of constraint, range = 0..1, use default color: [-1, -1, -1]</td>
</tr>
<tr>
<td><code>Graphics.color_body2</code></td>
<td>vector</td>
<td>[0.7, 0.8, 0.3]</td>
<td>Second color of constraint, range = 0..1, use default color: [-1, -1, -1]</td>
</tr>
<tr>
<td><code>Graphics.spring_diameter</code></td>
<td>double</td>
<td>-1</td>
<td>Spring diameter used for drawing only.</td>
</tr>
<tr>
<td><code>Graphics.spring_coils</code></td>
<td>double</td>
<td>10</td>
<td>Spring coils used for drawing. If set to 0, then a cylinder with the value 'spring_diameter' as diameter is shown instead of the coils.</td>
</tr>
<tr>
<td><code>Graphics.damper_diameter</code></td>
<td>double</td>
<td>-1</td>
<td>Damper diameter used for drawing only. If set to 0, then the damper is not shown. It's recommended to choose the value smaller than the spring diameter.</td>
</tr>
</tbody>
</table>

**Physics**

<table>
<thead>
<tr>
<th>Physics.attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Physics.spring_length</code></td>
<td>double</td>
<td>0</td>
<td>Length of the spring in the initial configuration</td>
</tr>
<tr>
<td><code>Physics.actor_force</code></td>
<td>double</td>
<td>0</td>
<td>Constant force acting on the spring</td>
</tr>
<tr>
<td><code>Physics.force_mode</code></td>
<td>integer</td>
<td>0</td>
<td>Defines how the spring and damper force is computed: 0=constant, 1=MathFunction, 2=IOElementDataModifier</td>
</tr>
</tbody>
</table>

**Physics.Linear**

<table>
<thead>
<tr>
<th>Physics.Linear.attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spring_stiffness</code></td>
<td>double</td>
<td>100</td>
<td>Stiffness coefficient of the linear spring. Only used if <code>force_mode</code> is 0.</td>
</tr>
<tr>
<td><code>damping</code></td>
<td>double</td>
<td>1</td>
<td>Damping coefficient for viscous damping. Only used if <code>force_mode</code> is 0.</td>
</tr>
</tbody>
</table>

**Physics.MathFunction**

<table>
<thead>
<tr>
<th>Physics.MathFunction.attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>piecewise_mode</code></td>
<td>integer</td>
<td>-1</td>
<td>Modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
</tr>
<tr>
<td><code>piecewise_points</code></td>
<td>vector</td>
<td>[]</td>
<td>Supporting points (e.g., time or place) for piecewise interpolation</td>
</tr>
<tr>
<td><code>piecewise_values</code></td>
<td>vector</td>
<td>[]</td>
<td>Values at supporting points</td>
</tr>
<tr>
<td><code>piecewise_diff_values</code></td>
<td>vector</td>
<td>[]</td>
<td>Differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td><code>parsed_function</code></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>String representing parsed function, e.g., 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td><code>parsed_function_parameter</code></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>String representing parameter of parsed function, e.g., 't'</td>
</tr>
</tbody>
</table>
### 3.3. CONNECTOR

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>piecewise_mode</strong></td>
</tr>
<tr>
<td>integer</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>modus for piecewise interp.: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>piecewise_points</strong></td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>[]</td>
</tr>
<tr>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>piecewise_values</strong></td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>[]</td>
</tr>
<tr>
<td>values at supporting points</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>piecewise_diff_values</strong></td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>[]</td>
</tr>
<tr>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>parsed_function</strong></td>
</tr>
<tr>
<td>string</td>
</tr>
<tr>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>string representing parsed function, e.g. ( A \sin(\omega \cdot t) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Physics.MathFunction.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MathFunction_d.</strong></td>
</tr>
<tr>
<td><strong>parsed_function_parameter</strong></td>
</tr>
<tr>
<td>string</td>
</tr>
<tr>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>string representing parameter of parsed function, e.g. ( t )</td>
</tr>
</tbody>
</table>

**Position 1**

<table>
<thead>
<tr>
<th><strong>Position1.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>element_number</strong></td>
</tr>
<tr>
<td>integer</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>Number of constrained element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Position1.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>position</strong></td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>[0, 0]</td>
</tr>
<tr>
<td>local position 1</td>
</tr>
</tbody>
</table>

**Position 2**

<table>
<thead>
<tr>
<th><strong>Position2.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>element_number</strong></td>
</tr>
<tr>
<td>integer</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>Number of constrained element (0 if ground joint)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Position2.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>position</strong></td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>[0, 0]</td>
</tr>
<tr>
<td>local or global position 2</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th><strong>value name</strong></th>
<th><strong>description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-14</td>
</tr>
<tr>
<td>Internal.second_order_variable</td>
<td>second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Internal.second_order_variable_velocity</td>
<td>velocities of second order variables of the element. range: 1-7</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.force</td>
<td>resultant force of connector</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-2, corresponds to force in global x-y direction</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_length</td>
<td>actual spring length</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_ elongation</td>
<td>elongation of spring</td>
</tr>
<tr>
<td>Connector.SpringDamperActuator.spring_velocity</td>
<td>magnitude of spring velocity</td>
</tr>
</tbody>
</table>

### Controllable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th><strong>value name</strong></th>
<th><strong>description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector.SpringDamperActuator.spring_length_off</td>
<td>prescribe the neutral spring length</td>
</tr>
</tbody>
</table>
Example
see file SpringDamperActuator2D.txt

mass
{  
  element_type = "Mass2D"  %specification of element type.
  Initialization.initial_position = [1, 0.5]  %initial position [x,y]
  Physics.mass = 1  %total mass of point mass
}
nMass = AddElement(mass)

sda
{  
  element_type = "SpringDamperActuator2D"  %specification of element type.
  Position1.element_number = nMass  %Number of constrained element
}
nSDA = AddConnector(sda)

3.3.21 PointJoint2D

Short description
The PointJoint2D is a simplified version of the PointJoint for 2D elements. It constrains two elements at local element positions. If only one element is specified (second element 0), a ground PointJoint is realized. It provides both Lagrangian and penalty formulation.

Description of the different modi

| element to ground | Position2.element_number has to be equal to 0 |
| element to element | Position2.element_number must not be equal to 0 |
| Lagrange | If Physics.use_penalty_formulation = 0, then no stiffness and no damping parameters are used. |

Data objects of PointJoint2D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;PointJoint2D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;PointJoint2D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>2</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.3, 0.8, 0.3]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
</tbody>
</table>
### 3.3. CONNECTOR

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.draw_size_joint_local_frame</td>
<td>double</td>
<td>-1</td>
<td>drawing dimensions of joint local frame. If set to -1, then global_draw_scalar_size is used. If set to 0, then no joint local frame is drawn.</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>double</td>
<td>-1</td>
<td>drawing dimensions of constraint. If set to -1, then global_draw_scalar_size is used.</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th>Geometry.use_joint_local_frame</th>
<th>bool</th>
<th>0</th>
<th>Use a special joint local frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.joint_local_frame</td>
<td>double</td>
<td>0</td>
<td>Prerotate stiffness vector w.r.t. global coordinate system or local coordinate system of body 1 with angle phi_z about the z axis. Just used if use_joint_local_frame == 1</td>
</tr>
<tr>
<td>Geometry.use_local_coordinate_system</td>
<td>bool</td>
<td>0</td>
<td>0 = use global coordinates, 1 = use local coordinate system of Body 1</td>
</tr>
</tbody>
</table>

#### Physics

| Physics.use_penalty_formulation | bool | 0 | 0 = use lagrange multipliers (index 3 DAE, exact), 1 = use penalty formulation (no additional equation added, approximate constraint) |

#### Physics.Penalty

<table>
<thead>
<tr>
<th>Physics.Penalty.spring_stiffness</th>
<th>double</th>
<th>1e+008</th>
<th>general or penalty stiffness parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics.Penalty.spring_stiffness_vector</td>
<td>vector</td>
<td>[0, 0]</td>
<td>penalty stiffness parameter [kx,ky]. Just used if scalar spring_stiffness == 0, otherwise kx=ky=spring_stiffness</td>
</tr>
<tr>
<td>Physics.Penalty.damping</td>
<td>double</td>
<td>1</td>
<td>damping coefficient for viscous damping (F = d*v), applied in all constrained directions</td>
</tr>
</tbody>
</table>

#### Physics.Lagrange

| Physics.Lagrange.constrained_directions | vector | [1, 1] | [x,y]...(1 = constrained, 0 = free), can be defined as local or global directions (see Geometry) |

#### Position 1

<table>
<thead>
<tr>
<th>Position1.element_number</th>
<th>integer</th>
<th>1</th>
<th>Number of constrained element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position1.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>local position 1</td>
</tr>
</tbody>
</table>

#### Position 2

<table>
<thead>
<tr>
<th>Position2.element_number</th>
<th>integer</th>
<th>0</th>
<th>Number of constrained element (0 if ground joint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position2.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>local or global position 2</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-2</td>
</tr>
<tr>
<td>Internal.algebraic_variable</td>
<td>algebraic variables of the element. range: 1-2</td>
</tr>
<tr>
<td>Connector.force</td>
<td>force applied to the kinematic pairs due to the connector. range: 1-2, corresponds to force in global x-y direction</td>
</tr>
</tbody>
</table>
Example

see file PointJoint2D.txt

grav
{
    load_type= "Gravity"  %specification of load type.
    direction= 1  %global direction of the gravity
    gravity_constant= 9.81  %use negative sign if necessary
}
nLoad = AddLoad(grav)

mass
{
    element_type= "Mass2D"  %specification of element type.
    loads= [nLoad]
    Initialization.initial_position= [1, 0.5]  %initial position [x,y]
    Physics.mass= 1  %total mass of point mass
}
nMass = AddElement(mass)

sda
{
    element_type= "PointJoint2D"  %specification of element type.
    Position1.element_number= nMass  %Number of constrained element
}
nSDA = AddConnector(sda)
3.4 CONTROL ELEMENTS

3.4 Control elements

These control elements are available:

- IODiscreteTransferFunction, 3.4.1
- IODigitalFilter, 3.4.2
- IORandomSource, 3.4.3
- IOLinearTransformation, 3.4.4
- IOQuantizer, 3.4.5
- IOContinuousTransferFunction, 3.4.6
- IOLinearODE, 3.4.7
- IOMathFunction, 3.4.8
- IOSaturate, 3.4.9
- IODEadZone, 3.4.10
- IPOProduct, 3.4.11
- IOTime, 3.4.12
- IOPulseGenerator, 3.4.13
- IOTimeWindow, 3.4.14
- IOStopComputation, 3.4.15
- IOElementDataModifier, 3.4.16
- IODisplay, 3.4.17
- IODGraph3D, 3.4.18
- IOMinMax, 3.4.19
- IOTCPIPBlock, 3.4.20
- IOX2C, 3.4.21
- IOLinearTransducer, 3.4.22

Control elements are connectors which have input- and/or output-ports.

Note:
In HOTINT several classes are treated as 'elements'. Connectors and control elements are also 'elements', and can therefore be edited and deleted in the GUI with the menu items of the elements.
In the script language the command AddConnector has to be used for connectors and also for the control elements in the list above.
3.4.1 IODiscreteTransferFunction

Short description
Discontinuous transfer function in z-space. It is a SISO (single input-single output) control element. Initial state is zero.

Equations
\[ y(z) = G(z)u(z) \]
\[ G(z) = \frac{\text{num}}{\text{den}} \]
user input:
\[ \text{num}(z) = num_1 + num_2z + num_3z^2 + ... + num_{n+1}z^n \]
\[ \text{den}(z) = den_1 + den_2z + den_3z^2 + ... + den_{n+1}z^n \]

Theoretical background: Realization of z-transfer function as time discrete state space model

\[
\begin{bmatrix}
  z_{k+1,1} \\
  \vdots \\
  z_{k+1,n}
\end{bmatrix} =
\begin{bmatrix}
  0 & . & 0 & -den_1 \\
  1 & 0 & . & -den_2 \\
  . & . & . & . \\
  0 & . & 1 & -den_n
\end{bmatrix}
\begin{bmatrix}
  z_{k,1} \\
  \vdots \\
  z_{k,n}
\end{bmatrix} +
\begin{bmatrix}
  num_1 - num_{n+1}den_1 \\
  \vdots \\
  num_n - num_{n+1}den_n
\end{bmatrix} u_k \tag{3.30}
\]

\[ y_k = z_{k,n} + num_{n+1}u_z; \tag{3.31} \]

![Figure 3.47: IODiscreteTransferFunction](image)

Data objects of IODiscreteTransferFunction:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IODiscreteTransferFunction&quot;</td>
<td>specification of element type. Once the element is added to the nbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IODiscreteTransferFunction&quot;</td>
<td>name of the element</td>
</tr>
</tbody>
</table>
### 3.4 CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>element_number</th>
<th>integer</th>
<th>\textbf{R}</th>
<th>1</th>
<th>number of the element in the mbs</th>
</tr>
</thead>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1=-90°, 2=-180°, 3=270°, 4=360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### IOBlock

<table>
<thead>
<tr>
<th>IOBlock.number_of_inputs</th>
<th>integer</th>
<th>\textbf{R}</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>\textbf{R}</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>\textbf{R}</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.discrete_time_step</td>
<td>double</td>
<td>0</td>
<td>Sample time $dT$</td>
<td></td>
</tr>
<tr>
<td>IOBlock.discrete_time_offset</td>
<td>double</td>
<td>0</td>
<td>Sample offset $T_k = k \cdot dT + \text{off}$</td>
<td></td>
</tr>
<tr>
<td>IOBlock.num_coeffs</td>
<td>vector</td>
<td>[1]</td>
<td>Coefficients of numerator polynomial of $z$-function</td>
<td></td>
</tr>
<tr>
<td>IOBlock.den_coeffs</td>
<td>vector</td>
<td>[1]</td>
<td>Coefficients of denominator polynomial of $z$-function</td>
<td></td>
</tr>
</tbody>
</table>

#### Observable special values:

For more information see section \[3.1\]

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-2</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

#### Example

see file ZTransferFunction.txt
3.4.2 IODigitalFilter

Short description
A digital (highpass or lowpass) filter of 2nd order.

Equations
The filter is fully defined by the following parameters:
- \( f_c \), cut off frequency in Hz
- \( f_s = \frac{1}{\Delta T} \), sampling frequency
- Q-factor, see remarks below

The Q-factor is influencing the damping of the filter, the following values are important:
- \( Q=0.5 \) critically damped
- \( Q<0.5 \) overdamped
- \( Q=\frac{1}{\sqrt{2}} \) Butterworth
- \( Q=\frac{1}{\sqrt{3}} \) Bessel

The coefficients of the 2nd order discrete transfer function are computed automatically in this IOBlock.
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>lowpass (default)</th>
<th>the flag highpass is set to 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>highpass</td>
<td>the flag highpass has to be set to 1</td>
</tr>
</tbody>
</table>

![Figure 3.48: IODigitalFilter](image)

**Data objects of IODigitalFilter:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IODigitalFilter&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IODigitalFilter&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation; 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>input_nodes_num</td>
<td>vector</td>
<td></td>
<td></td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>discrete_time_step</td>
<td>double</td>
<td></td>
<td>0.001</td>
<td>Sample time dT</td>
</tr>
<tr>
<td>highpass</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>1..highpass, 0..lowpass</td>
</tr>
<tr>
<td>&amp;</td>
<td>double</td>
<td>100</td>
<td></td>
<td>cut off frequency of filter in Hz</td>
</tr>
<tr>
<td>Q</td>
<td>double</td>
<td></td>
<td>0.707</td>
<td>Q-factor: Q = 0.5.critically damped, Q smaller than 0.5. overdamped</td>
</tr>
</tbody>
</table>
IOBlock.num_coeffs vector R [0.0675, 0.135, 0.0675] Coefficients of numerator polynomial of z-function

IOBlock.den_coeffs vector R [0.413, -1.14, 1] Coefficients of denominator polynomial of z-function

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-4</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example
see file IODigitalFilter.txt

Time.element_type= "IOTime"
nETime = AddElement(Time)

Signal
{
  element_type= "IOMathFunction"
  Graphics.position= [50, 0]
  IOBlock.input_element_numbers= [nETime]
  IOBlock.input_element_types= [1] % i=IOElement
  IOBlock.input_local_number= [1]
  IOBlock.MathFunction.parsed_function= "10*sin(2*pi*5*t)+2*sin(2*pi*300*t)"
  IOBlock.MathFunction.parsed_function_parameter= "t"
}
nESig=AddElement(Signal)

Filter
{
  element_type= "IODigitalFilter"
  Graphics.position= [100, 0]
  IOBlock.input_element_numbers= [nESig]
  IOBlock.input_element_types= [1] % i=IOElement
  IOBlock.input_local_number= [1]
  IOBlock.discrete_time_step= 0.001 %Sample time dT
  IOBlock.fc= 10 %cut off frequency of filter in Hz
  IOBlock.Q= 0.7 %Q-factor
3.4. CONTROL ELEMENTS

}

nEFilt=AddElement(Filter)

SensorOutput
{
    sensor_type= "ElementSensor"
    element_number= nESig
    value= "IOBlock.output[1]"
}
AddSensor(SensorOutput)

SensorOutput.element_number= nEFilt
AddSensor(SensorOutput)

3.4.3 IORandomSource

Short description
Discontinuous random source using alternatively an internal C++ based pseudo random generator or a linear feedback shift register. It has no input and one output.

Description of the different modi

<table>
<thead>
<tr>
<th>method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IOBlock.method must be set to 0. The built-in random generator is used.</td>
</tr>
<tr>
<td>1</td>
<td>IOBlock.method must be set to 1. Generate a pseudo random binary signal by using Linear Feedback Shift Register.</td>
</tr>
</tbody>
</table>

Figure 3.49: IORandomSource

Data objects of IORandomSource:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IORandomSource&quot;</td>
<td>specification of element type. Once the element is added to the nbs, you MUST NOT change this type anymore!</td>
</tr>
</tbody>
</table>
### Graphics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>Reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>Draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>Rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>Background color: -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>Foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td>Number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
</tr>
</tbody>
</table>

### IOBlock

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>Number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>Number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>Number of states</td>
</tr>
<tr>
<td>IOBlock.discrete_time_step</td>
<td>double</td>
<td>Sample time dT</td>
</tr>
<tr>
<td>IOBlock.discrete_time_offset</td>
<td>double</td>
<td>Sample offset off: Tk = k*dT + off</td>
</tr>
<tr>
<td>IOBlock.max_amplitude</td>
<td>double</td>
<td>Max. amplitude of random value.</td>
</tr>
<tr>
<td>IOBlock.mean_value</td>
<td>double</td>
<td>Offset of random signal.</td>
</tr>
<tr>
<td>IOBlock.method</td>
<td>bool</td>
<td>Random generator method.</td>
</tr>
<tr>
<td>IOBlock.bits</td>
<td>integer</td>
<td>Number of bits for random signal.</td>
</tr>
<tr>
<td>IOBlock.constant_amplitude</td>
<td>bool</td>
<td>Output values are +amplitude or -amplitude if flag is activate.</td>
</tr>
<tr>
<td>IOBlock.seed</td>
<td>double</td>
<td>Seed € [0,1]... initialization of random generator</td>
</tr>
<tr>
<td>IOBlock.init_val</td>
<td>double</td>
<td>Initial value of the generator x(t=0) = y(t=0)</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>Value name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>Data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-1</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

### Example

see file addRandomSource.txt
3.4 CONTROL ELEMENTS

random
{
  element_type = "IORandomSource"
  IOblock
  {
    discrete_time_step = 0.05
    discrete_time_offset = 0.0
    max_amplitude = 2
    mean_value = 2.5
    method = 1
    bits = 15
    constant_amplitude = 0
    seed = 0.5
    init_val = 2.5
  }
  Graphics
  {
    position = [0,-50]
  }
}

nRandom = AddElement(random)

3.4.4 IOLinearTransformation

Short description
Continuous linear transformation. The transfer function type is SISO (single input-single output) or MIMO (multi input-multi output).

Equations

\[ y = Au + b; \]  \hspace{1cm} (3.32)

Matrix \( A \) and vector \( b \) are user defined.

Description of the different modi

<table>
<thead>
<tr>
<th>Linear transformation ( y = Au )</th>
<th>Set ( b ) to zero.</th>
</tr>
</thead>
<tbody>
<tr>
<td>gain ( y_1 = A_{1,1}u_1 )</td>
<td>Set ( A ) as scalar value and ( b ) is zero.</td>
</tr>
<tr>
<td>constant ( y_1 = b_1 )</td>
<td>Set ( A ) to zero and ( b ) to the constant value.</td>
</tr>
</tbody>
</table>
### Data objects of IOLinearTransformation:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOLinearTransformation&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOLinearTransformation&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th></th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td>[]</td>
<td></td>
</tr>
</tbody>
</table>

#### IOBlock

<table>
<thead>
<tr>
<th>IOBlock.number_of_inputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>4</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
</tbody>
</table>

| IOBlock.input_element_numbers | vector |   | [] | vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted! |
| IOBlock.input_element_types   | vector |   | [] | vector with types of connected inputs; 1=IOElement, 2=Sensor |
| IOBlock.input_local_number   | vector |   | [] | vector with i-th number of output of previous IOelement connected to this element |

| IOBlock.A_matrix | matrix |   | \([0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 0, 0]\) | transformation matrix A: \(y = Ax + b\) |
| IOBlock.b_vector | vector |   | \([0, 0, 0, 0]\) | offset vector b: \(y = Ax + b\) |

![Figure 3.50: IOLinearTransformation](image-url)
3.4. CONTROL ELEMENTS

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example
see file LinearTransformation.txt

Include("addTime.txt")

transformation
{
  element_type = "IOLinearTransformation"
  Graphics
  {
    position = [50, 0] %reference drawing position
  }
  IOBlock
  {
    input_element_numbers = [nTime]
    input_element_types = [1]
    input_local_number = [1]
    A_matrix = [2]
    b_vector = [0.5]
  }
}
nTrans = AddElement(transformation)
nSens = nTrans
nDisp = nTrans
Include("addSens.txt")
Include("addDisplay.txt")

3.4.5 IOQuantizer

Short description
A quantizer block passes its input signal through a stair-step function so that many neighboring points on the input axis are mapped to one point on the output axis. The effect is to quantize a smooth signal into a stair-step output. It is a SISO (single input-single output) control element.
Equations

\[
y(u) = \begin{cases} 
  r \lfloor \frac{u}{r} + 0.5r \rfloor, & \text{if } r! = 0 \\
  u, & \text{if } r = 0
\end{cases}
\]  

(3.33)

The user defined rounding value is \( r \).

---

**Figure 3.51: IOQuantizer**

---

### Data objects of IOQuantizer:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;IOQuantizer&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;IOQuantizer&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td></td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td></td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4=360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td></td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td></td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>matrix</td>
<td></td>
<td></td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IBlock**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBOck.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IBOck.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IBOck.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IBOck.input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IBOck.input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>IBOck.input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>IBOck.rounding_value</td>
<td>double</td>
<td></td>
<td>0.1</td>
<td>Max. amplitude of random value.</td>
</tr>
</tbody>
</table>
Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file Quantizer.txt

Include("addTime.txt")

quantizer
{
  element_type = "IOQuantizer"
  IOBlock
  {
    rounding_value = 0.2
    input_element_numbers = [nTime]
    input_element_types = [1]
    input_local_number = [1]
  }
  Graphics
  {
    position = [50,0]
  }
}

nQuantizer = AddElement(quantizer)

nSens = nQuantizer
nDisp = nQuantizer

Include("addSens.txt")
Include("addDisplay.txt")

3.4.6 IOContinuousTransferFunction

Short description

The STransferFunction is a linear transfer function for continuous state-space elements. It is a SISO (single input-single output) type.

Equations

\[ y(s) = G(s)u(s) \]
\[ G(s) = \frac{\text{num}(s)}{\text{den}(s)} \]

user input:
\[
\text{num}(s) = num_1 + num_2 s + num_3 s^2 + \ldots + num_{n+1} s^n
\]
\[
\text{den}(s) = den_1 + den_2 s + den_3 s^2 + \ldots + den_{n+1} s^n
\]

Figure 3.52: IOContinuousTransferFunction

Data objects of IOContinuousTransferFunction:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOContinuousTransferFunction&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOContinuousTransferFunction&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

Graphics

| Graphics.show_connector | bool     |   | 1                   | Flag to draw connector                                                     |
| Graphics.position      | vector   |   | [0, 0]              | reference drawing position                                                 |
| Graphics.draw_size     | vector   |   | [20, 20, 0]         | draw size                                                                  |
| Graphics.rotation      | double   |   | 0                   | rotation: 1==90°, 2==180°, 3==270°, 4=360°                                 |
| Graphics.background_color | vector  |   | [-1, -1, -1]        | background color; -1=transparent                                           |
| Graphics.foreground_color | vector  |   | [0, 0, 0]           | foreground color                                                           |
| Graphics.input_nodes_num | vector  |   | []                 | number of input of drawing position "input_nodes"                          |
| Graphics.input_nodes  | matrix   |   | []                 |                                                                               |

IOBlock

| IOBlock.number_of_inputs | integer | R | 0 | number of inputs |
| IOBlock.number_of_outputs | integer | R | 1 | number of outputs |
| IOBlock.number_of_states | integer | R | 3 | number of states |
| IOBlock.input_element_numbers | vector  |   | [] | vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted! |
| IOBlock.input_element_types | vector  |   | [] | vector with types of connected inputs; 1=IOElement, 2=Sensor |
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>IOBlock</th>
<th>input_local_number</th>
<th>vector</th>
<th>[]</th>
<th>vector with i-th number of output of previous IOelement connected to this element</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock</td>
<td>numerator</td>
<td>vector</td>
<td>[1, 0, 0, 0]</td>
<td>ascending numerator coefficients $n$ of transfer-function. $TF = \frac{num}{den}$ with $num = n(1)s^0 + n(2)s + n(3)s^2 + ...$ Will be normalized automatically!</td>
</tr>
<tr>
<td>IOBlock</td>
<td>denominator</td>
<td>vector</td>
<td>[0, 0, 0, 1]</td>
<td>ascending denominator coefficients $d$ of transfer-function. $TF = \frac{num}{den}$ with $den = d(1)s^0 + d(2)s + d(3)s^2 + ...$ Will be normalized automatically!</td>
</tr>
</tbody>
</table>

Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-3</td>
</tr>
<tr>
<td>Internal.first_order_variable</td>
<td>first order variables of the element. range: 1-3</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file STTransferFunction.txt

Include("addTime.txt")

```
str transferfunction
{
  element_type = "IOContinuousTransferFunction"
  IOBlock
  {
     input_element_numbers = [nTime]
     input_element_types = [1]
     input_local_number = [1]
  }
  Graphics
  {
     position = [50, 0]
  }
}
```

nSTransferFunction = AddElement(str transferfunction)

nSens = nSTransferFunction
nDisp = nSTransferFunction

Include("addSens.txt")
3.4.7 IOLinearODE

Short description

The LinearODE Element represents a linear ordinary differential equation of SISO (single input-single output) or MIMO (multi input-multi output) type.

Equations

\[
\begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx + Du
\end{align*}
\]

Matrices \(A, B, C\) and \(D\) are user defined.

![Figure 3.53: IOLinearODE](image)

Data objects of IOLinearODE:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;IOLinearODE&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;IOLinearODE&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>([0, 0])</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>([20, 20, 0])</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>([-1, -1, -1])</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>([0, 0, 0])</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>0</td>
<td>number of inputs</td>
</tr>
</tbody>
</table>
### 3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th><strong>IOBlock</strong>.number_of_outputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IOBlock</strong>.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with i-th number of output of previous IOelement connected to this element</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.A_coeffs</td>
<td>matrix</td>
<td></td>
<td>0</td>
<td>Coefficients of state matrix $A$, $x_{\text{dot}} = Ax + Bu$</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.B_coeffs</td>
<td>matrix</td>
<td></td>
<td>0</td>
<td>Coefficients of input matrix $B$, $x_{\text{dot}} = Ax + Bu$</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.C_coeffs</td>
<td>matrix</td>
<td></td>
<td>0</td>
<td>Coefficients of output matrix $C$, $y = Cx + Du$</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.D_coeffs</td>
<td>matrix</td>
<td></td>
<td>0</td>
<td>Coefficients of output matrix $D$, $y = Cx + Du$</td>
</tr>
<tr>
<td><strong>IOBlock</strong>.initial_vector</td>
<td>vector</td>
<td></td>
<td></td>
<td>Initial values of time-domain variables</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1](#)

<table>
<thead>
<tr>
<th><code>value name</code></th>
<th><code>description</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IOBlock.output</strong></td>
<td><strong>IOBlock.output[i]</strong> ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td><strong>IOBlock.input</strong></td>
<td><strong>IOBlock.input[i]</strong> ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

**Example**

see file LinearODE.txt

Include("addTime.txt")

lin
{
    element_type = "IOLinearODE"
    Graphics
    {
        position = [50, 0] %reference drawing position
    }
    IOBlock
    {
        input_element_numbers = [nTime]
        input_element_types = [1]
        input_local_number = [1]
        A_coeffs = [1]
        B_coeffs = [1]
        C_coeffs = [1]
\[
\text{D_coeffs} = [1] \\
\text{initital_vector} = [1] \\
\}
\]
\]
nLin = AddElement(lin)

nSens = nLin
nDisp = nLin

Include("addSens.txt")
Include("addDisplay.txt")

3.4.8 IOMathFunction

Short description

A IOMathFunction contains a mathematical expression with functions and logical operators or a lookup table with different modes for piecewise interpolation. The output is result of the evaluation of the MathFunction as a function of input.

Description of the different modi

<table>
<thead>
<tr>
<th>parsed function</th>
<th>In order to use the parser for mathematical expressions, the variable IOBlock.MathFunction.piecewise_mode has to be set to -1. In IOBlock.MathFunction.parsed_function one specifies a string representing parsed function, e.g. ( A \times \sin(u) ) with function parameter ( u ) defined in IOBlock.MathFunction.parsed_function_parameter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>piecewise mode - constant</td>
<td>IOBlock.MathFunction.piecewise_mode must be set to 0. The vectors IOBlock.MathFunction.piecewise_points and IOBlock.MathFunction.piecewise_values are used. The output value is piecewise constant with jumps at the supporting points.</td>
</tr>
<tr>
<td>piecewise mode - linear</td>
<td>IOBlock.MathFunction.piecewise_mode must be set to 1. The vectors IOBlock.MathFunction.piecewise_points and IOBlock.MathFunction.piecewise_values are used. The output value is piecewise linear between the supporting points.</td>
</tr>
<tr>
<td>piecewise mode - quadratic</td>
<td>IOBlock.MathFunction.piecewise_mode must be set to 2 and in addition to the other piecewise modes the vector IOBlock.MathFunction.piecewise_diff_values is needed. The output is a quadratic interpolation between the supporting points.</td>
</tr>
</tbody>
</table>
### Data objects of IOMathFunction:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOMathFunction&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOMathFunction&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation; 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td><strong>IOBlock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td><strong>IOBlock.MathFunction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.MathFunction.piecewise_mode</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>modus for piecewise interpolation: -1=not piece-wise, 0=constant, 1=linear, 2=quadratic</td>
</tr>
<tr>
<td>IOBlock.MathFunction.piecewise_points</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
</tbody>
</table>
### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-1</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

### Example

see file IOMathFunction.txt

```plaintext
Time
{
  element_type= "IOTime"
}
nTime = AddElement(Time)

% IOMathfunction with one input piecewise
IOBlock
{
  element_type= "IOMathFunction"
  Graphics.position= [50, 0]
  IOBlock
  {
    input_element_numbers= [nTime]
    input_element_types= [1]
    input_local_number= [1]
    MathFunction
    {
      piecewise_mode= 0
      piecewise_points= [0, 1, 1.5, 2]
      piecewise_values= [0, 1, 0.7, 0]
    }
  }
}
```
nElem = AddElement(IOBlock)

SensorOutput
{
    sensor_type= "ElementSensor"
    element_number= nElem
    value= "IOBlock.output[1]"
}
AddSensor(SensorOutput)

% IOMathfunction with multiple inputs and parsed function
IOBlock
{
    element_type= "IOMathFunction"
    Graphics.position= [100, 0]
    IOBlock
    {
        input_element_numbers= [nTime, nTime]
        input_element_types= [1, 1]
        input_local_number= [1, 1]
        MathFunction
        {
            piecewise_mode= -1
            parsed_function = "u*(v>4)&&(v<6)"
            parsed_function_parameter = "u,v"
        }
    }
}

nElem = AddElement(IOBlock)

SensorOutput.element_number = nElem
AddSensor(SensorOutput)

### 3.4.9 IOSaturate

**Short description**

Continuous saturation element for upper and lower limits. It is a SISO (single input-single output) control element.

**Equations**

\[
    y(u) = \begin{cases} 
        ul, & \text{if } u > ul \\
        u, & \text{if } ll \leq u \leq ul \\
        ll, & \text{if } u < ll 
    \end{cases} \tag{3.34}
\]

In the defined equation \( ul \) is the upper limit and \( ll \) is the lower limit.
Data objects of IOSaturate:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOSaturate&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOSaturate&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
<td></td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4=360°</td>
</tr>
<tr>
<td>Graphics.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input_nodes_num</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td>IOBlock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>input_element_numbers</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>input_element_types</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>input_local_number</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>upper_limit</td>
<td>double</td>
<td>0.1</td>
<td>Upper limit of saturate.</td>
<td></td>
</tr>
<tr>
<td>lower_limit</td>
<td>double</td>
<td>0</td>
<td>Lower limit of saturate.</td>
<td></td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file Saturate.txt

Include("addTime.txt")

saturate
{
   element_type = "IOSaturate"
   IOBlock
   {
      upper_limit = 2.6
      lower_limit = 2.4
      input_element_numbers = [nTime]
      input_element_types = [1]
      input_local_number = [1]
   }
   Graphics
   {
      position = [50,0]
   }
}

nSaturate = AddElement(saturate)

nSens = nSaturate
nDisp = nSaturate

Include("addSens.txt")
Include("addDisplay.txt")

3.4.10 IODeadZone

Short description

Continuous dead-zone element. The outputs between upper and lower limit is zero. This leads to an offset of the input signal by the corresponding lower or upper limit. It is a SISO (single input-single output) control element.
Equations

\[
y(u) = \begin{cases} 
  u - sd, & \text{if } u < sd \\
  0, & \text{if } u \geq sd \text{ and } u \leq ed \\
  u - ed, & \text{if } u > ed 
\end{cases} \quad (3.35)
\]

In the defined equation, \( sd \) is the start dead-zone value, \( ed \) is the end dead-zone value.

![Figure 3.56: IODeadZone](image)

Data objects of IODeadZone:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>( R )</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IODeadZone&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IODeadZone&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>( R )</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics:\show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics:\position</td>
<td>vector</td>
<td></td>
<td>( [0, 0] )</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics:\draw_size</td>
<td>vector</td>
<td></td>
<td>( [20, 20, 0] )</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics:\rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90F, 2==180F, 3==270F, 4==360F</td>
</tr>
<tr>
<td>Graphics:\background_color</td>
<td>vector</td>
<td></td>
<td>([-1, -1, -1])</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics:\foreground_color</td>
<td>vector</td>
<td></td>
<td>([0, 0, 0])</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics:\input_nodes_num</td>
<td>vector</td>
<td></td>
<td>( [] )</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics:\input_nodes</td>
<td>matrix</td>
<td></td>
<td>( [] )</td>
<td></td>
</tr>
<tr>
<td>IOB_Block:\number_of_inputs</td>
<td>integer</td>
<td>( R )</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOB_Block:\number_of_outputs</td>
<td>integer</td>
<td>( R )</td>
<td>1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOB_Block:\number_of_states</td>
<td>integer</td>
<td>( R )</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOB_Block:\input_element_numbers</td>
<td>vector</td>
<td></td>
<td>( [] )</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IOB_Block:\input_element_types</td>
<td>vector</td>
<td></td>
<td>( [] )</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>IOBlock. input_local_number</th>
<th>vector</th>
<th>[]</th>
<th>vector with i-th number of output of previous IOelement connected to this element</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.start_deadzone</td>
<td>double</td>
<td>0</td>
<td>Start of dead zone.</td>
</tr>
<tr>
<td>IOBlock.end_deadzone</td>
<td>double</td>
<td>0</td>
<td>End of dead zone.</td>
</tr>
</tbody>
</table>

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example
see file DeadZone.txt

#include("addTime.txt")

defadzone
{
    element_type = "IODeadZone"
    IODBlock
    {
        start_deadzone = 1
        end_deadzone = 2
        input_element_numbers = [nTime]
        input_element_types = [1]
        input_local_number = [1]
    }
    Graphics
    {
        position = [50,0]
    }
}
nDeadZone = AddElement(deadzone)
	nSens = nDeadZone	nDisp = nDeadZone

#include("addSens.txt")
#include("addDisplay.txt")
3.4.11 IOPProduct

Short description

Continuous product (or division) of one or more inputs. A dedicated exponent for every input and a offset can be applied.

Equations

\[ y(u) = u_1^{exp_1}u_2^{exp_2}...u_n^{exp_n} + offset \]  

All exponents are stored in a vector. For a simple multiplication with a input the dedicated exponent is set to 1, for a division the exponent is set to -1. The offset is a scalar value.

![IOPProduct](image)

Figure 3.57: IOPProduct

Data objects of IOPProduct:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOPproduct&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOPproduct&quot; name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1 number of the element in the mbs</td>
</tr>
<tr>
<td>element_type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>element_number</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td>[]</td>
<td></td>
</tr>
</tbody>
</table>

**IOBlock**

<table>
<thead>
<tr>
<th>IOBlock.number_of_inputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th><strong>Input element numbers</strong></th>
<th>vector</th>
<th>[]</th>
<th>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input element types</strong></td>
<td>vector</td>
<td>[]</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td><strong>Input local number</strong></td>
<td>vector</td>
<td>[]</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td><strong>Exponents</strong></td>
<td>vector</td>
<td>[0]</td>
<td>Exponent of inputs.</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>double</td>
<td>0</td>
<td>Output offset.</td>
</tr>
</tbody>
</table>

**Observable special values:**

For more information see section [3.1](#)

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

**Example**

see file Product.txt

```plaintext
Include("addTime.txt")
Include("addRandomSource.txt")

product
{
  element_type = "IOProduct"
  IOBlock
  {
    exponents = [2,3]
    offset = -1
    input_element_numbers = [nTime,nRandom]
    input_element_types = [1,1]
    input_local_number = [1,1]
  }
  Graphics
  {
    position = [50,0]
  }
}

nProduct = AddElement(product)
nSens = nProduct
nDisp = nProduct
```
3.4.12 IOTime

Short description

Continuous time source. This element simply outputs the time.

![IOTime](image)

Figure 3.58: IOTime

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOTime&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOTime&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

### Graphics

<table>
<thead>
<tr>
<th>Graphics.show_connector</th>
<th>bool</th>
<th></th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td>[]</td>
<td></td>
</tr>
</tbody>
</table>

### IOBlock

<table>
<thead>
<tr>
<th>IOBlock.number_of_inputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
</tbody>
</table>
Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file addTime.txt

time
{
    element_type = "IOTime"
    name = "time"
    Graphics
    {
        position = [0, 0] %reference drawing position
        draw_size = [20, 20, 0] %draw size
    }
}

nTime = AddElement(time)

3.4.13 IOPulseGenerator

Short description

Continuous pulse generator. This element outputs repeating sequence or rectangular pulses after a certain delay. It has no input and one output.

Equations

\[
\Delta t = t - t_{offset} \tag{3.37}
\]

\[
t_{rest} = \Delta t \mod p \tag{3.38}
\]

\[
y(t) = \begin{cases} 
a, & \text{if } \Delta t \geq 0 \text{ and } t_{rest} < pw \\
0, & \text{else} \end{cases} \tag{3.39}
\]

User defined variables are pulse amplitude \(a\), time offset \(t_{offset}\), signal period \(p\) and pulse width \(pw\).
### Data objects of IOPulseGenerator:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>R</td>
<td>&quot;IOPulseGenerator&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;IOPulseGenerator&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

- Graphics.show_connector: bool
  - default: 1
  - description: Flag to draw connector
- Graphics.position: vector
  - default: [0, 0]
  - description: reference drawing position
- Graphics.draw_size: vector
  - default: [20, 20, 0]
  - description: draw size
- Graphics.rotation: double
  - default: 0
  - description: rotation: 1==90°, 2==180°, 3==270°, 4==360°
- Graphics.background_color: vector
  - default: [-1, -1, -1]
  - description: background color; -1=transparent
- Graphics.foreground_color: vector
  - default: [0, 0, 0]
  - description: foreground color
- Graphics.input_nodes_num: vector
  - default: []
  - description: number of input of drawing position "input_nodes"
- Graphics.input_nodes: matrix
  - default: []

#### IOBlock

- IOBlock.number_of_inputs: integer
  - default: 0
  - description: number of inputs
- IOBlock.number_of_outputs: integer
  - default: 1
  - description: number of outputs
- IOBlock.number_of_states: integer
  - default: 0
  - description: number of states
- IOBlock.amplitude: double
  - default: 1
  - description: Amplitude of rectangle pulse generator.
- IOBlock.offset: double
  - default: 0
  - description: Time offset (s).
- IOBlock.period: double
  - default: 1
  - description: Period of signal (s).
- IOBlock.pulse_width: double
  - default: 0.5
  - description: Pulse width (s).
- IOBlock.use_external_time_source: integer
  - default: 0
  - description: 1|(0) ... (Don't) use external input as time source.

### Observable special values:

For more information see section 3.1
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file addPulseGenerator.txt

```
pulse
{
    element_type = "IOPulseGenerator"
    IOBlock
    {
        amplitude = 2
        offset = 1
        period = 0.2
        pulse_width = 0.1
    }
    Graphics
    {
        position = [0,-50]
    }
}
nPulse = AddElement(pulse)
```

3.4.14 IOTimeWindow

Short description

This element helps to capture a special time window. It has two inputs and one output.

Equations

\[
(a) \quad y(u) = \begin{cases} 
    u_2, & \text{if } t_{start} \leq u_1 \leq t_{end} \\
    0, & \text{else}
\end{cases} 
\] 

\[
(b) \quad y(u) = \begin{cases} 
    u_2, & \text{if } t_{start} \leq u_1 \\
    0, & \text{else}
\end{cases} 
\]

Description of the different modi

| \( t_{end} > t_{start} \) | Output is determined with inequation (a). |
| \( t_{end} \leq t_{start} \) | Output is determined with inequation (b). |
Data objects of IOTimeWindow:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOTimeWindow&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOTimeWindow&quot; name of the element</td>
<td></td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
<td></td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
<td></td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td>vector of input of drawing position &quot;input_nodes&quot;</td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td>vector of input of drawing position &quot;input_nodes&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>IOBlock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
<td></td>
</tr>
<tr>
<td>IOBlock.t_start</td>
<td>double</td>
<td></td>
<td>0</td>
<td>Start time (s).</td>
</tr>
<tr>
<td>IOBlock.t_end</td>
<td>double</td>
<td></td>
<td>0</td>
<td>End time (s).</td>
</tr>
</tbody>
</table>
Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file TimeWindow.txt

Include("addTime.txt")
Include("addPulseGenerator.txt")

```plaintext
window
{
    element_type = "IOTimeWindow"
    IOBlock
    {
        t_start = 1
        t_end = 2
        input_element_numbers = [nTime,nPulse]
        input_element_types = [1,1]
        input_local_number = [1,1]
    }
    Graphics
    {
        position = [50,0]
    }
}
nWindow = AddElement(window)
```

nSens = nWindow
nDisp = nWindow

Include("addSens.txt")
Include("addDisplay.txt")

### 3.4.15 IOStopComputation

Short description

This element stops the computation, if input is unequal zero. It has one input and no output. When the element IOStopComputation stops the computation, the values of the sensors are written to the sol-file. On the one hand the last integration step is always included, on the other hand a time step is included which will not fit to the equidistant points of time in the solution file.
Data objects of IOStopComputation:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOStopComputation&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOStopComputation&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4=360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td></td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IOBlock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>closecomputation</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>1. close HOTINT, 0..stop computation</td>
</tr>
<tr>
<td>errorcode</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>error code returned when this element triggers HOTINT to close</td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file StopComputation.txt

Include("addPulseGenerator.txt")

stop
{
  element_type = "IOStopComputation"
  IOBlock
  {
    input_element_numbers = [nPulse]
    input_element_types = [1]
    input_local_number = [1]
  }
  Graphics
  {
    position = [50,-50]
  }
}
nStop = AddElement(stop)

3.4.16 IOElementDataModifier

Short description

This element can be used to modify data of a constraint or element. It has one input and no output.
**Figure 3.62: IOElementDataModifier**

**Data objects of IOElementDataModifier:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOElementDataModifier&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOElementDataModifier&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
<td></td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
<td></td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4=360°</td>
<td></td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
<td></td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
<td></td>
</tr>
<tr>
<td>Graphics.input_node_num</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IOBlock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.start_of_timestep_only</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>modify element data at start time step only.</td>
</tr>
<tr>
<td>IOBlock.mod_variable_name</td>
<td>string</td>
<td>nn</td>
<td>variable name of the modified element data</td>
<td></td>
</tr>
<tr>
<td>IOBlock.mod_element_number</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>element number of the modified element or constraint</td>
</tr>
</tbody>
</table>
Observable special values:
For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output[i]</td>
<td>IOBlock.output[i] measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input[i]</td>
<td>IOBlock.input[i] access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example
see file ElementDataModifier.txt

```
gravLoad
{
  load_type = "Gravity"
  direction = 3 % z - direction
  gravity_constant = 9.81 % m/s^2
}
nLoad = AddLoad(gravLoad)

mass3D
{
  element_type = "Mass3D"
  loads = [nLoad]
  Physics.mass= 1
}
nMass3D = AddElement(mass3D)

IOTime
{
  element_type = "IOTime"
}
nIOTime = AddElement(IOTime)

springDamperActuator
{
  element_type = "SpringDamperActuator"
  Position1.element_number = nMass3D % number of constrained element
}
nSpringDamperActuator = AddConnector(springDamperActuator)

modifier
{
  element_type = "IOElementDataModifier"
  IOBlock
  {
    input_element_numbers = [nIOTime] % element connected to input
  }
}
input_element_types = [1]
input_local_number = [1]
mod_variable_name = "Connector.SpringDamperActuator.spring_length_offset"  %modified element data
mod_element_number = nSpringDamperActuator %modified constraint

} }
AddElement(modifier)

3.4.17 IODisplay

Short description

This element can be used to display any (single) numerical value fed into the (single) input.

![Figure 3.63: IODisplay](image)

Data objects of IODisplay:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IODisplay&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IODisplay&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td></td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td></td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td></td>
<td>[60, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td></td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td></td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
</tbody>
</table>
### 3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_of_states</td>
<td>integer</td>
<td>number of states</td>
</tr>
<tr>
<td>input_element_numbers</td>
<td>vector</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>input_element_types</td>
<td>vector</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>input_local_number</td>
<td>vector</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>number_of_digits</td>
<td>integer</td>
<td>number of digits</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-1</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

### Example

see file Display.txt

```bash
Include("addTime.txt")

display
{
  element_type = "IODisplay"
  IOBlock
  {
    input_element_numbers = \[nTime\]
    input_element_types = \[1\]
    input_local_number = \[1\]
    number_of_digits = 3
  }
  Graphics
  {
    position = \[70,0\]
  }
}
nDisplay = AddElement(display)
```
3.4.18 IOGraph3D

Short description
This element can be used to plot 3 input values as a 3d graph directly in the rendered 3D-scene in the main window.

Data objects of IOGraph3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOGraph3D&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOGraph3D&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics show connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics draw size</td>
<td>vector</td>
<td>[60, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4=360°</td>
</tr>
<tr>
<td>Graphics background color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics foreground color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics input_nodes_num</td>
<td>vector</td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics input_nodes</td>
<td>matrix</td>
<td>[]</td>
<td></td>
</tr>
</tbody>
</table>

**IOBlock**

<table>
<thead>
<tr>
<th>IOBlock number_of_inputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock input_element_numbers</td>
<td>vector</td>
<td>[]</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
<td></td>
</tr>
<tr>
<td>IOBlock input_element_types</td>
<td>vector</td>
<td>[]</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
<td></td>
</tr>
<tr>
<td>IOBlock input_local_number</td>
<td>vector</td>
<td>[]</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
<td></td>
</tr>
<tr>
<td>IOBlock number_of_plotted_steps</td>
<td>integer</td>
<td>100</td>
<td>if positive, only values of the last number_of_plotted_steps are plotted in graph</td>
<td></td>
</tr>
<tr>
<td>IOBlock sample_time</td>
<td>double</td>
<td>0</td>
<td>must be positive, every sampling_time seconds the graph is updated</td>
<td></td>
</tr>
<tr>
<td>IOBlock origin</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Global position of the graph's origin</td>
<td></td>
</tr>
<tr>
<td>IOBlock magnification</td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Magnification of the graph in each global direction</td>
<td></td>
</tr>
</tbody>
</table>

Observable special values:
For more information see section 3.1
3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>Internal.data_variable</th>
<th>data variables of the element which are no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-601</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.Input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

Example

see file Graph3D.txt

3.4.19 IOMinMax

Short description

This block returns the minimum, maximum or average value of the input. Up to a specific point of time, this functionality is switched off and the output \( y \) is equal to the input \( u \). This block can be used to postprocess sensor values.

Description of the different modi

<table>
<thead>
<tr>
<th>Mod</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = minimum</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \min_{t \geq t_0} (u) &amp; \text{for } t &gt; t_0 \end{cases} ] with ( t_0 = \text{IOBlock.start.time} )</td>
</tr>
<tr>
<td>2 = maximum</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \max_{t \geq t_0} (u) &amp; \text{for } t &gt; t_0 \end{cases} ]</td>
</tr>
<tr>
<td>3 = average</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \frac{1}{N} \sum_{t_i \geq t_0} u_i &amp; \text{for } t_i &gt; t_0 \end{cases} ]</td>
</tr>
<tr>
<td>4 = minimum(abs)</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \min_{t \geq t_0} (</td>
</tr>
<tr>
<td>5 = maximum(abs)</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \max_{t \geq t_0} (</td>
</tr>
<tr>
<td>6 = average(abs)</td>
<td>[ y = \begin{cases} u &amp; \text{for } t \leq t_0 \ \frac{1}{N} \sum_{t_i \geq t_0}</td>
</tr>
</tbody>
</table>

![MinMax](image)

Figure 3.64: IOMinMax
## Data objects of IOMinMax:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;IOMinMax&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;IOMinMax&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics</th>
<th>type</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td></td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IOBlock**

<table>
<thead>
<tr>
<th>IOBlock</th>
<th>type</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R 0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R 1</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R 0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td></td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td></td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
<tr>
<td>IOBlock.input_local_number</td>
<td>vector</td>
<td></td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
</tr>
<tr>
<td>IOBlock.mode</td>
<td>integer</td>
<td>1</td>
<td>1=min, 2=max, 3.avg, 4.min(abs), 5.max(abs), 6.avg(abs)</td>
</tr>
<tr>
<td>IOBlock.start_time</td>
<td>double</td>
<td>0</td>
<td>Up to this point of time, the output is equal to the input. Afterwards the output is computed according to the mode.</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section [3.1](#)

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.data_variable</td>
<td>data variables of the element which have no degrees of freedom (e.g. inelastic strain, contact state, friction state, etc.). range: 1-2</td>
</tr>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

Example

see file IOMinMax.txt

Time.element_type= "IOTime"
nElem1 = AddElement(Time)

MinMax
{
    element_type= "IOMinMax"
    Graphics.position= [50, 0]
    IOTBlock
    {
        input_element_numbers= [nElem1]
        input_element_types= [1]
        input_local_number= [1]
        mode = 1 % minimum
        start_time = 0.5
    }
}
nElem2 = AddElement(MinMax)

SensorOutput
{
    sensor_type= "ElementSensor"
    element_number= nElem1
    value= "IOTBlock.output[1]"
}
AddSensor(SensorOutput)

SensorOutput.element_number= nElem2
AddSensor(SensorOutput)

3.4.20 IOTCPIPBlock

Short description

This I/O element is a communication block based on TCP/IP which allows HOTINT to connect to other programs or tools, opening up a wide range of possible applications including external control, user-defined “add-ons”, or even co-simulation. Based on the specified IP (v4) address and port number the IOTCPIPBlock sets up a server socket and waits for a connection request from a client. Hence, HOTINT here plays the server role, and the external program is the client application.

Limitations

For the use of this element some kind of active network adapter is required. If you only want to communicate locally on your computer and do not have an active network adapter, you can alternatively use a so-called “loopback device” which emulates an active real network adapter in a real network. To this end, either use the localhost address 127.0.0.1 (this is the default) – or one address from the 127.0.0.0/8 subnet (127.0.0.1-127.255.255.254) – or create and configure an actual virtual network adapter. The following steps summarize how such a loopback adapter
can be installed on Microsoft Windows:

(1) Open the device manager
(2) Select the network category and choose “Action → Add legacy hardware” via the menu
(3) Choose the option for manual installation and select the category “network adapters” from the list
(4) In the next dialog select “Microsoft” as vendor and “Microsoft loopback adapter” as hardware component
(5) Proceed and finish the installation

Example: Communication with Simulink/Matlab
This example demonstrates how to realize a connection between HOTINT and Matlab/Simulink. The purpose of the TCP/IP block is to use other powerful tools for some computations. For example it is possible to do the control law calculations for the actuation of the multibody system in Simulink (as alternative to the IOBlocks in HOTINT). It’s also very simple to do a parameter variation, see the advanced example in the folder “examples/balancing_cart_TCP/IP”. In “examples/TCP/IP” a very simple communication example is included: From the HOTINT side four different double values (simulation time $t$ multiplied by the gain factors one to four) are transmitted to Simulink, see figure 3.66. Simulink summates the first and second respectively the third and fourth value and sends the two double values back to HOTINT. The values are captured by sensors, stored in the solution file and can be visualized in the plot tool.

Comment: For testing purposes you can also use the executable “TCP/IP_client.exe” which has the same functionality as the Simulink example. To use this client executable create a “IP.txt” file in the same folder. The first four lines represent the IP address of the HOTINT computer, the fifth line is the port number.

To start this example, following things have to be done:
(1) Start Matlab/Simulink and open the file communication.mdl in the folder “examples/TCP/IP”, see figure 3.67
Comment: If the “Instrument Control Toolbox” is not installed the TCP/IP communication blocks appear red and indicate an unresolved reference to a library block (bad link). The figure shows the basic structure that should not be changed. The output of the “TCP/IP Receive” block is a vector $y_{rec} = [t, x_1, \ldots, x_n, f]^T$ with HOTINT time $t$, data variables $x_1$ to $x_n$ and the handling flag $f$. The “Selector” block outputs the last element of the vector (flag $f$) for the flag handling. You have to adapt the these two blocks if you want to change the number of received variables. There is no need to change the “flag handling in” block.
(2) Make sure that the “Current Folder” is the folder which include the communication.mdl file.
(3) Double click the “TCP/IP Receive” block and select the “Remote address” (i.e., the IP address) of the computer HOTINT is running on and select a “Port”. Repeat this point for the “TCP/IP Send” block.
Comment: If HOTINT and Simulink is running on the same computer you can also choose localhost (“127.0.0.1”).
(4) Set the “Sample Time” of every block (TCP/IP Receive, Constants,…) and choose fixed step size in the “Solver Options”.
(5) Open the subsystem “computations”, see figure 3.68. This subsystem contains all computations $y = f(u)$ with input $u$ and output $y$.
Comment: Change this subsystem to your needs.
(6) Open the subsystem “flag handling out”, see figure 3.70. In default no handling flags are transmitted to HOTINT.
3.4. CONTROL ELEMENTS

Comment: Change this subsystem to your needs.

(7) Save the mdl file. (8) Open the TCPIP_hid HOTINT file and type in the same “ip_address” and “port_number” as for the Matlab/Simulink side.

(9) Make sure that “max_step_size” and “min_step_size” in the subtree “SolverOptions.Timeint” are set to the same value as the fixed “Sample Time” in Simulink.

Comment: This is very important especially for the case of time dependent blocks like integrators in Simulink.

(10) Save the file.

(11) Load the communication_hid file in HOTINT.

(12) Click the “Start simulation” button in Simulink.

(13) Click the “Start!” button in HOTINT.

Comment: The points 11-13 have to be executed within the timeout limits. You can change the latter in the TCP/IP blocks for both HOTINT and Simulink. During these steps connection errors might occur due to firewall restrictions; you will probably have to set the corresponding permissions in your firewall(s).

It is also recommended to choose the Simulink “Simulation stop time” higher as the “end_time” in HOTINT. The reason is that HOTINT sends a stop flag after the last simulation step and in Simulink this flag is used to execute a “Stop” block which ends the communication and simulation.

Additional notes

Data exchange is performed at a stage before every time step in HOTINT, following below protocol:

The outgoing data, i.e. the data sent from HOTINT to the client, is an array of 8-byte double precision numbers which contains, in that order, the current simulation time (1 double), the current values of the inputs of the I/O element, and one additional element corresponding to a communication control flag (see the Communication flags - section below for more details). Hence, the total amount of outgoing data is (number of inputs + 2) times 8 bytes (double precision numbers). After the client has received and processed that data, it sends back a data package to HOTINT – the incoming data for the I/O element – which again consists of an array of double precision numbers, this time with the length (number of outputs + 1). The first (number of output) double precision values determine the outputs of the I/O element, and the last element again is used for the transfer of communication flags.

HOTINT now begins the computation of one time step, where the transmitted data from the client is accessible via the outputs of the IOTCPIPBlock.

Important notes

- The waiting procedure for the client connection request, as well as the send and receive operations all are so-called “blocking calls”. This means that HOTINT will wait for those operations to finish, and during that time, not respond to any user input. Therefore, a reasonable timeout (default is 30 seconds) should be specified for the IOTCPIPBlock to allow TCP/IP connection or transmission error handling.

- You will probably have to adjust your firewall settings and set appropriate permissions for HOTINT and the client application.

- Depending on the implementation of the client, it might be necessary to start the server,
i.e., HOTINT, first.

Since HOTINT is running on Microsoft Windows, the memory byte order, also called “endiness”, is “Little Endian”, which means that the least significant bytes/digits are stored ‘first’ in memory, i.e., on the smallest memory address. Therefore, any data sent from or received by the IOTCPIPBlo c k has or must have that byte order, respectively. You probably have to take that into account on the client side, especially if the client is running on a different platform and/or architecture on another computer.

**Communication flags**

Currently, the following 4-byte flags are implemented:

1. Neutral flag: \(0x00000000\) (integer value: \(0\)). This flag signals that the application is running (properly) and no further action is required.
2. Reset flag: \(0x00000001\) (integer value: \(1\)). This flag is sent from HOTINT to the client in the first step of the computation. This can be used, for instance, to reset the client application.
3. Error flag: \(0x00000002\) (integer value: \(2\)). Indicates that an error has occurred. If HOTINT receives the error flag, an error message is issued, the connection is closed and the program execution terminated.
4. Close flag: \(0x00000003\) (integer value: \(3\)). This flag is sent from HOTINT to the client to indicate that the computation has finished and the connection will be closed, which is the case when the computation has actually finished, or the “Stop”-button has been hit.
5. Any other value: Treated as error flag (3).

One of these flags is stored in and read from the last 8 bytes of the exchanged data – corresponding to one additional double precision number – in either direction in every time step. Currently, for simplicity, the flag is just casted explicitly from an integer to a double precision number which then can be transmitted and casted back to an integer exactly. Of course, this procedure must be followed on both the server and the client side.

---

**Figure 3.65: general concept of TCP/IP coupling**
3.4. CONTROL ELEMENTS

Figure 3.66: TCP/IP Block with 4 inputs and 2 outputs

Figure 3.67: TCP/IP communication with Matlab/Simulink (do not change this structure)
Figure 3.68: Subsystem computations

Figure 3.69: TCP/IP subsystem “flag handling in”
Data objects of IOTCPIPBlocK:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td>&quot;IOTCPIPBlocK&quot; specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;IOTCPIPBlocK&quot; name of the element</td>
<td></td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R 1</td>
<td>number of the element in the mbs</td>
</tr>
<tr>
<td>Graphics.show_connector</td>
<td>bool</td>
<td>1</td>
<td>Flag to draw connector</td>
</tr>
<tr>
<td>Graphics.position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics.draw_size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics.rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics.background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics.foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics.input_nodes_num</td>
<td>vector</td>
<td>0</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>Graphics.input_nodes</td>
<td>matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOBlock.number_of_inputs</td>
<td>integer</td>
<td>R 0</td>
<td>number of inputs</td>
</tr>
<tr>
<td>IOBlock.number_of_outputs</td>
<td>integer</td>
<td>R 0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock.number_of_states</td>
<td>integer</td>
<td>R 0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock.input_element_numbers</td>
<td>vector</td>
<td>0</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
</tr>
<tr>
<td>IOBlock.input_element_types</td>
<td>vector</td>
<td>0</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
</tr>
</tbody>
</table>

Figure 3.70: TCP/IP subsystem “flag handling out”
<table>
<thead>
<tr>
<th><strong>Value Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock.output</td>
<td>IOBlock.output[i] ... measures the i-th output of this IOBlock, if available</td>
</tr>
<tr>
<td>IOBlock.input</td>
<td>IOBlock.input[i] ... access to the i-th input of this IOBlock, if available</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
time.element_type= "IOTime" %specification of element type.  
nTime = AddElement(time)

gain
{
  element_type= "IOLinearTransformation" %specification of element type.
  Graphics.position= [50, 0]
  IOBlock
  {
    input_element_numbers= [nTime] %v. of element(s) or sensor number(s)
    input_element_types= [1] %v. with types of connected inputs; 1=IOElement
    input_local_number= [1] %v. with i-th number of output of previous IOElement
    A_matrix= [1] %transformation matrix A: y=A.u+b
    b_vector= [0] %offset vector b: y=A.u+b
  }
}
nGain1 = AddElement(gain)

gain.IOBlock.A_matrix= [2]```
3.4. CONTROL ELEMENTS

```plaintext
gain.Graphics.position = [50, -30]
gain.IOBlock.A_matrix = [3]
gain.Graphics.position = [50, -60]
gain.IOBlock.A_matrix = [4]
gain.Graphics.position = [50, -90]
```

TCPIP

```plaintext
{  
element_type = "IOTCPIPBlock" % specification of element type.  
Graphics.position = [100, 0]  
IOBlock  
{     
    input_element_numbers = [nGain1,nGain2,nGain3,nGain4]  
    input_element_types = [1,1,1,1]  
    input_local_number = [1,1,1,1]  
    port_number = 50000  
    ip_address = "127.0.0.1"  
    received_data_size = 2  
    timeout = 10000  
}  
}
nTCPIP = AddElement(TCPIP)
```

```plaintext
sensor.sensor_type = "ElementSensor"  
sensor.element_number = nTCPIP  
sensor.value = "IOBlock.output[1]"  
nSensor1 = AddSensor(sensor)
```

```plaintext
sensor.value = "IOBlock.output[2]"  
nSensor2 = AddSensor(sensor)
SolverOptions.Timeint.max_step_size = 1  
SolverOptions.Timeint.min_step_size = 1  
SolverOptions.start_time = 0  
SolverOptions.end_time = 10
```

3.4.21 IOX2C

Short description

This I/O element is a communication block based on IOTCPIPBlock which allows HOTINT to connect to X2C. For the mapping of the inputs and outputs in HOTINT and X2C strings are used.

Additional notes

HOTINT is the server and X2C Application the client. HOTINT needs synchronisation time (sample time * 1.5) as double value.

Initialization
• X2C builds connection
• X2C sends synchronisation time
• HOTINT sends mapping of imports and outputs (see Port Identifier Order)

Initialization of imports and outputs
• HOTINT sends (init-) input values to X2C (see Port Value Exchange)
• X2C Update

Continuous communication
• X2C sends output values to HOTINT (see Port Value Exchange)
• HOTINT Update
• HOTINT sends input values to X2C (see Port Value Exchange)
• X2C Update

Closing of Communication
• usually done by server (HOTINT) by setting status flag
• in error case also possible for client by setting status flag

<table>
<thead>
<tr>
<th>Import Label</th>
<th>Length of Import Label</th>
<th>Import Label 1</th>
<th>Length of Import Label 1</th>
<th>...</th>
<th>Length of Import Label n</th>
<th>Import Label n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Label</td>
<td>Length of Output Label</td>
<td>Output Label 1</td>
<td>Length of Output Label 1</td>
<td>...</td>
<td>Length of Output Label n</td>
<td>Output Label n</td>
</tr>
</tbody>
</table>

Figure 3.71: IOX2C Port Identifier Order

<table>
<thead>
<tr>
<th>Status Flag</th>
<th>Value Import 1 (double)</th>
<th>Value Import 2 (double)</th>
<th>...</th>
<th>Value Import n (double)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Flag</td>
<td>Value Output 1 (double)</td>
<td>Value Output 2 (double)</td>
<td>...</td>
<td>Value Output n (double)</td>
</tr>
</tbody>
</table>

Figure 3.72: IOX2C Port Value Exchange

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>neutral, no errors, continue with procedure</td>
</tr>
<tr>
<td>0x00000001</td>
<td>reset, reset procedure and communication</td>
</tr>
<tr>
<td>0x00000002</td>
<td>error, error occurred, error handling by application</td>
</tr>
<tr>
<td>0x00000003</td>
<td>close, stop procedure and close communication</td>
</tr>
</tbody>
</table>

Figure 3.73: IOX2C Status Flag Values

Data objects of IOX2C:
### 3.4. CONTROL ELEMENTS

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOX2C&quot;</td>
<td>specification of element type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOX2C&quot;</td>
<td>name of the element</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the element in the mbs</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics show connector</th>
<th>bool</th>
<th>1</th>
<th>Flag to draw connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics position</td>
<td>vector</td>
<td>[0, 0]</td>
<td>reference drawing position</td>
</tr>
<tr>
<td>Graphics draw size</td>
<td>vector</td>
<td>[20, 20, 0]</td>
<td>draw size</td>
</tr>
<tr>
<td>Graphics rotation</td>
<td>double</td>
<td>0</td>
<td>rotation: 1==90°, 2==180°, 3==270°, 4==360°</td>
</tr>
<tr>
<td>Graphics background color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1=transparent</td>
</tr>
<tr>
<td>Graphics foreground color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>Graphics input_nodes_num</td>
<td>vector</td>
<td>[]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
</tbody>
</table>

#### IOBlock

<table>
<thead>
<tr>
<th>IOBlock number_of_inputs</th>
<th>integer</th>
<th>R</th>
<th>0</th>
<th>number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBlock number_of_outputs</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of outputs</td>
</tr>
<tr>
<td>IOBlock number_of_states</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of states</td>
</tr>
<tr>
<td>IOBlock input_element_numbers</td>
<td>vector</td>
<td>[]</td>
<td>vector of element(s) or sensor number(s) connected to input, only valid element numbers permitted!</td>
<td></td>
</tr>
<tr>
<td>IOBlock input_element_types</td>
<td>vector</td>
<td>[]</td>
<td>vector with types of connected inputs; 1=IOElement, 2=Sensor</td>
<td></td>
</tr>
<tr>
<td>IOBlock input_local_number</td>
<td>vector</td>
<td>[]</td>
<td>vector with i-th number of output of previous IOElement connected to this element</td>
<td></td>
</tr>
<tr>
<td>IOBlock port_number</td>
<td>integer</td>
<td>50000</td>
<td>Port number, e.g. ‘50000’.</td>
<td></td>
</tr>
<tr>
<td>IOBlock ip_address</td>
<td>string</td>
<td>&quot;127.0.0.1&quot;</td>
<td>IP address, e.g. ‘127.0.0.1’ (localhost). Do not neglect the dots between the numbers.</td>
<td></td>
</tr>
<tr>
<td>IOBlock sample_time</td>
<td>double</td>
<td>-1</td>
<td>Time span after which data is communicated. If value is -1 (default) then communication is performed at each time step.</td>
<td></td>
</tr>
<tr>
<td>IOBlock timeout</td>
<td>integer</td>
<td>30000</td>
<td>TCP/IP timeout in milliseconds; default is 30000.</td>
<td></td>
</tr>
<tr>
<td>IOBlock disable_receive</td>
<td>integer</td>
<td>0</td>
<td>incoming data communication is neglected.</td>
<td></td>
</tr>
<tr>
<td>IOBlock disable_send</td>
<td>integer</td>
<td>0</td>
<td>outgoing data communication is neglected.</td>
<td></td>
</tr>
<tr>
<td>IOBlock input_names</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Names of the inputs, separated with commas. Names have to be consistent with settings in X2C. Order of names has to match vectors specifying input element types, e.g. voltage, current.</td>
<td></td>
</tr>
<tr>
<td>IOBlock output_names</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Names of the outputs, separated with commas. Names have to be consistent with settings in X2C. e.g. position, velocity</td>
<td></td>
</tr>
</tbody>
</table>

#### Observable special values:

For more information see section [3.1](#).
3.4.22 IOLinearTransducer

Short description
The LinearTransducer realizes an electro-magnetic linear transducer.

Degrees of freedom
The element has 1 degree of freedom, the magnetic flux

Equations
The LinearTransducer computes the force which has to be applied to the mechanical bodies, e.g. Rigid3D. The magnetic flux $\Psi$ is computed as

$$\frac{d\Psi}{dt} = u - Ri(z, \Psi)$$

(3.42)

with displacement $z$, current $i$, voltage $u$ and resistance $R$. The force $f$ of the transducer is a function of displacement and current, $f(z, i)$, whereas the current is a function of displacement and magnetic flux $i(z, \Psi)$. To compute these values the LinearTransducer uses radial basis functions (RBF) of the form

$$y(x) = \sum_{i=1}^{N} w_i \phi(\|x_{sc_i} - c_i\|) + v \left[ \begin{array}{c} 1 \\ x_{sc_i} \end{array} \right]$$

(3.43)

with $N$ supporting points (centers) $c$ and the weights $w$ and $v$. The argument $x = [z, \Psi]$ for the current and $x = [z, i]$ for the force. The argument $x$ is scaled with the scaling vector $s$,

$$x_{sc_i} = x_i/s_i$$

(3.44)

Different kernels of the RBF are available:

- RBF kernel = 1: $\phi(\|x_{sc} - c_i\|) = \phi(r) = r^3$
- RBF kernel = 2: $\phi(\|x_{sc} - c_i\|) = \phi(r) = r^2 \ln(r)$

In the linear case you can use the following simplification for the force

$$v = [F_0, C_m, k_i]$$

(3.45)

$$w = []$$

(3.46)

$$c = []$$

(3.47)

$$s = [1; 1]$$

(3.48)
3.4. CONTROL ELEMENTS

Note: \( C_m \) is the destabilizing (negative) magnetic stiffness. The linear case is therefore equal to

\[
f(z, i) = F_0 + C_m z + k_i i
\]  

(3.49)

and for the current

\[
v = \left[ -\frac{\Psi_0}{L}, -k_i, 1 \right]
\]  

(3.50)

\[
w = []
\]  

(3.51)

\[
c = []
\]  

(3.52)

\[
s = [1; 1]
\]  

(3.53)

which is equal to

\[
i(z, \Psi) = -\frac{\Psi_0}{L} - \frac{k_i}{L} z + \frac{1}{L} \Psi
\]  

(3.54)

Inputs and Outputs

- input 1: voltage \( u \) in V
- input 2: displacement \( z \) in m
- output 1: force \( f \) in N
- output 2: negative force \( -f \) in N
- output 3: current \( i \) in A

![Figure 3.74: IOLinearTransducer](image)

Data objects of IOLinearTransducer:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element_type</td>
<td>string</td>
<td></td>
<td>&quot;IOLinearTransducer&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;IOLinearTransducer&quot;</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
</tr>
</tbody>
</table>

**Graphics**

- Graphics.show_connector bool 1 Flag to draw connector
- Graphics.position vector [0, 0] reference drawing position
- Graphics.draw_size vector [20, 20, 0] draw size
- Graphics.rotation double 0 rotation: 1==90F, 2==180F, 3==270F, 4==360F
### Graphics

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background_color</td>
<td>vector</td>
<td>[-1, -1, -1]</td>
<td>background color; -1 = transparent</td>
</tr>
<tr>
<td>foreground_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>foreground color</td>
</tr>
<tr>
<td>input_nodes_num</td>
<td>vector</td>
<td>[ ]</td>
<td>number of input of drawing position &quot;input_nodes&quot;</td>
</tr>
<tr>
<td>input_nodes</td>
<td>matrix</td>
<td>[ ]</td>
<td></td>
</tr>
</tbody>
</table>

### IOBlock

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_of_inputs</td>
<td>integer</td>
<td>R 2</td>
<td>number of inputs</td>
</tr>
<tr>
<td>number_of_outputs</td>
<td>integer</td>
<td>R 3</td>
<td>number of outputs</td>
</tr>
<tr>
<td>number_of_states</td>
<td>integer</td>
<td>R 1</td>
<td>number of states</td>
</tr>
<tr>
<td>input_element_numbers</td>
<td>vector</td>
<td>[2, 3]</td>
<td>numbers of IOElement or sensor providing the inputs [voltage, displacement]</td>
</tr>
<tr>
<td>input_element_types</td>
<td>vector</td>
<td>[1, 1]</td>
<td>types of connected inputs; 1 = IOElement, 2 = Sensor</td>
</tr>
<tr>
<td>local_number</td>
<td>vector</td>
<td>[1, 1]</td>
<td>i-th number of output of previous IOElement connected to this element</td>
</tr>
</tbody>
</table>

### Physics

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resistance</td>
<td>double</td>
<td>0.6</td>
<td>electrical resistance in Ohm</td>
</tr>
</tbody>
</table>

#### Physics.RBF_Force

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaling_vector</td>
<td>vector</td>
<td>[1, 1]</td>
<td>scaling of displacement, x(1), and current, x(2): $x_{scaled}(i) = x(i)/\text{scaling_vector}(i)$</td>
</tr>
<tr>
<td>centers</td>
<td>matrix</td>
<td>[ ]</td>
<td>$c(i)$ is the scaled supporting point</td>
</tr>
<tr>
<td>weights_RBF</td>
<td>vector</td>
<td>[ ]</td>
<td>$w(i)$ is the weight of the supporting point $c(i)$</td>
</tr>
<tr>
<td>weights_poly</td>
<td>vector</td>
<td>[0, 1.4e+004, 14]</td>
<td>$v(i)$, weights of the polynomial $v^* [1; x(1); x(2)]$, in linear case: $v = [F_0, C_m, k_i]$</td>
</tr>
<tr>
<td>RBF_kernel</td>
<td>integer</td>
<td>1</td>
<td>kernel of the RBF: $1 \cdot r \cdot r \cdot r$, $2 \cdot r \cdot r \cdot \ln(r)$</td>
</tr>
</tbody>
</table>

#### Physics.RBF_Current

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaling_vector</td>
<td>vector</td>
<td>[1, 1]</td>
<td>scaling of displacement, x(1), and current, x(2): $x_{scaled}(i) = x(i)/\text{scaling_vector}(i)$</td>
</tr>
<tr>
<td>centers</td>
<td>matrix</td>
<td>[ ]</td>
<td>$c(i)$ is the scaled supporting point</td>
</tr>
<tr>
<td>weights_RBF</td>
<td>vector</td>
<td>[ ]</td>
<td>$w(i)$ is the weight of the supporting point $c(i)$</td>
</tr>
<tr>
<td>weights_poly</td>
<td>vector</td>
<td>[0, -1e+003, 71.4]</td>
<td>$v(i)$, weights of the polynomial $v^* [1; x(1); x(2)]$, in linear case: $v = [-\Psi_0/L, -k_i/L, 1/L]$</td>
</tr>
<tr>
<td>RBF_kernel</td>
<td>integer</td>
<td>1</td>
<td>kernel of the RBF: $1 \cdot r \cdot r \cdot r$, $2 \cdot r \cdot r \cdot \ln(r)$</td>
</tr>
</tbody>
</table>

### Observable special values:

For more information see section 3.1

<table>
<thead>
<tr>
<th>value name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal.DOF</td>
<td>degrees of freedom (or generalized unknowns) of the element. range: 1-1</td>
</tr>
<tr>
<td>Internal.first_order_variable</td>
<td>first order variables of the element. range: 1-1</td>
</tr>
</tbody>
</table>
3.4. CONTROL ELEMENTS

| IOBlock.output | IOBlock.output[i] ... measures the i-th output of this IOBlock |
| IOBlock.input | IOBlock.input[i] ... access to the i-th input of this IOBlock |

Example

see file IOLinearTransducer.txt

```plaintext
volt.element_type= "IOLinearTransformation"
volt.IOBlock.A_matrix= []
volt.IOBlock.b_vector= [10] % constant voltage of 10 V
nEVolt = AddElement(volt)

rigidbody.element_type= "Mass1D"
rigidbody.Physics.mass= 0.315 % add 1 body
nEBody = AddElement(rigidbody)

elemSet.set_type = "ElementSet"
elemSet.element_numbers= [nEBody] % set with 1 body
nESetTilgerMoving = AddSet(elemSet)

spring.element_type= "CoordinateConstraint" % spring w.r.t. ground
spring.Physics.use_penalty_formulation= 1 % spring has to be sufficient stiff!
spring.Physics.Penalty.damping= 15 % damping coefficient Dp for viscous damping
spring.Physics.Penalty.spring_stiffness= 20000 % general or penalty stiffness parameter Sp
spring.Coordinate1.element_number= nEBody
spring.Coordinate1.local_coordinate= 1
AddConnector(spring)

sens.sensor_type= "FVElementSensor"
sens.element_number= nEBody
sens.field_variable= "displacement" % displacement is needed as input
nSensDisp = AddSensor(sens)

elektroMagnet
{
  element_type= "IOLinearTransducer"
  Graphics.position= [100, 0]
  IOBlock.input_element_numbers= [nEVolt, nSensDisp] % inputs [voltage, displacement]
  IOBlock.input_element_types= [1, 2] % types; 1=IOElement, 2=Sensor
}
nELinTransducer = AddElement(elektroMagnet)

force
{
  load_type= "GCLoad"
  load_function_type= 2 % time dependency of the load: 2..IOElement
  IOElement.input_element_number= nELinTransducer % number of IOElement in the mbs
  IOElement.input_local_number= 1 % number of output of IOElement connected to this element
```
nLTilger = AddLoad(force)  % add load to mbs
AssignLoad(nESetTilgerMoving,nLTilger)  % assign load to element

sensEl
{
    name = "IO voltage"
    sensor_type = "ElementSensor"
    element_number = nELinTransducer
    value = "IOBlock.input[1]"
}
AddSensor(sensEl)

sensEl.name = "IO displacement"
AddSensor(sensEl)

sensEl.name = "IO force"
AddSensor(sensEl)

sensEl.name = "IO current"
AddSensor(sensEl)
3.5 Material

These materials are available:

- Material, 3.5.1
- MaterialThermalExpansion, 3.5.2
- MaterialElastoplastic, 3.5.3
- MaterialElastoplasticThermalExpansion, 3.5.4

Note:
In HOTINT several classes are treated as 'material'. BeamProperties are also 'materials', and can therefore be edited and deleted in the GUI with the menu items of the materials. In the script language the command AddMaterial is just available for the materials in the list above.

3.5.1 Material

Short description

Material is the basic Object for defining material properties for standard finite elements (in contrast to structural finite elements such as beams and plates).

Additional notes

For static problems define the elastic properties Solid.youngs_modulus and Solid.poisson_ratio, whereas for dynamic problems also Solid.density is required. If the problem is planar (Solid.plane is set to 1), then the plane strain case is assumed unless Solid.plane_stress is set to 1. If the material is inelastic, then also the properties in the subtree Inelasticity have to be set.

Data objects of Material:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material_number</td>
<td>integer</td>
<td>R</td>
<td>4</td>
<td>type of the material</td>
</tr>
<tr>
<td>material_type</td>
<td>string</td>
<td></td>
<td>&quot;Material&quot;</td>
<td>name of the material</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Material&quot;</td>
<td>material color (as yet used with FEMesh, only)</td>
</tr>
</tbody>
</table>

Graphics

| Graphics.color            | vector   | [0, 0, 1] | material color  |

Solid

| Solid.plane              | bool     | 0          | true: 2D, false: 3D |
| Solid.plane_stress       | bool     | 0          | for 2D-Elements only; 1: plane stress, 0: plane strain |
| Solid.density            | double   | 0          | material density, must not be set for static problems |
| Solid.youngs_modulus     | double   | 0          | Youngs modulus   |
| Solid.poisson_ratio      | double   | 0          | Poisson ratio    |
| Solid.is_orthotropic_material | bool     | 0          | check to enable orthotropic parameters |

Solid.Orthotropic

| Solid.Orthotropic.E1     | double   | 0          | elasticity along axis 1 |
| Solid.Orthotropic.E2     | double   | 0          | elasticity along axis 2 |
| Solid.Orthotropic.E3     | double   | 0          | elasticity along axis 3 |
### Solid.Orthotropic

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NU12</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 12</td>
</tr>
<tr>
<td>NU13</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 13</td>
</tr>
<tr>
<td>NU23</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 23</td>
</tr>
<tr>
<td>G12</td>
<td>double</td>
<td>0</td>
<td>shear modulus 12</td>
</tr>
<tr>
<td>G13</td>
<td>double</td>
<td>0</td>
<td>shear modulus 13</td>
</tr>
<tr>
<td>G23</td>
<td>double</td>
<td>0</td>
<td>shear modulus 23</td>
</tr>
</tbody>
</table>

### Damping

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damping.c_M</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (( C = c__M \times M + c__K \times K ))</td>
</tr>
<tr>
<td>Damping.c_K</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (( C = c__M \times M + c__K \times K ))</td>
</tr>
</tbody>
</table>

### Inelasticity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inelasticity.inelasticity_solution_method</td>
<td>string</td>
<td>&quot;fixed_point&quot;</td>
<td>fixed_point, return_mapping (see Simo and Hughes, Computational Inelasticity 1998)</td>
</tr>
<tr>
<td>nr_inelastic_variables</td>
<td>integer</td>
<td>R</td>
<td>number of inelastic variables used by material class</td>
</tr>
</tbody>
</table>

### Example

see file Material.txt

```plaintext
class my_material
{
    material_type = "Material"
    Solid.density = 7800
    Solid.youngs_modulus = 2e10
}

nMaterial = AddMaterial(my_material)
```

### 3.5.2 MaterialThermalExpansion

#### Short description

Material which considers thermal expansion.

#### Data objects of MaterialThermalExpansion:

<table>
<thead>
<tr>
<th>Data name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material_number</td>
<td>integer</td>
<td>R</td>
<td>4</td>
</tr>
<tr>
<td>material_type</td>
<td>string</td>
<td>&quot;MaterialThermalExpansion&quot;</td>
<td>type of the material</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;MaterialThermalExpansion&quot;</td>
<td>name of the material</td>
</tr>
</tbody>
</table>

#### Graphics

| Graphics.color          | vector  | [0, 0, 1] | material color (as yet used with FEMesh, only)                              |

#### Solid

<table>
<thead>
<tr>
<th>Solid</th>
<th>bool</th>
<th>0</th>
<th>true: 2D, false: 3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid.plane</td>
<td>bool</td>
<td>0</td>
<td>for 2D-Elements only; 1: plane stress, 0: plane strain</td>
</tr>
<tr>
<td>Solid.plane_stress</td>
<td>bool</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Solid.density</td>
<td>double</td>
<td>0</td>
<td>material density, must not be set for static problems</td>
</tr>
<tr>
<td>Solid.youngs_modulus</td>
<td>double</td>
<td>0</td>
<td>Youngs modulus</td>
</tr>
<tr>
<td>Solid.poisson_ratio</td>
<td>double</td>
<td>0</td>
<td>Poisson ratio</td>
</tr>
<tr>
<td>Solid.is_orthotropic_material</td>
<td>bool</td>
<td>0</td>
<td>check to enable orthotropic parameters</td>
</tr>
</tbody>
</table>
3.5. MATERIAL

<table>
<thead>
<tr>
<th>Solid.Orthotropic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid.Orthotropic.E1 double 0 elasticity along axis 1</td>
</tr>
<tr>
<td>Solid.Orthotropic.E2 double 0 elasticity along axis 2</td>
</tr>
<tr>
<td>Solid.Orthotropic.E3 double 0 elasticity along axis 3</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU12 double 0 poisson ration 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU13 double 0 poisson ration 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU23 double 0 poisson ration 23</td>
</tr>
<tr>
<td>Solid.Orthotropic.G12 double 0 shear modulus 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.G13 double 0 shear modulus 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.G23 double 0 shear modulus 23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Damping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damping.c_M double 0 used for Rayleigh damping ( C = c_M * M + c_K * K )</td>
</tr>
<tr>
<td>Damping.c_K double 0 used for Rayleigh damping ( C = c_M * M + c_K * K )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inelasticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inelasticity.inelasticity_solution_method string &quot;fixed_point&quot; fixed_point, return_mapping (see Simo and Hughes, Computational Inelasticity 1998)</td>
</tr>
<tr>
<td>Inelasticity.nr_inelastic_variables integer R 0 number of inelastic variables used by material class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ThermalExpansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThermalExpansion.delta_T double 0 temperature difference to reference material</td>
</tr>
<tr>
<td>ThermalExpansion.coefficient double 0 thermal expansion coefficient</td>
</tr>
</tbody>
</table>

Example
see file MaterialThermalExpansion.txt

3.5.3 MaterialElastoplastic

Short description
MaterialElastoplastic is a material which obeys linear elasticity and Prandtl Reuss flow rule with Mises yield condition. Also, material hardening is considered.

Additional notes
Please make sure, that all items in subtree Inelasticity are defined.

Data objects of MaterialElastoplastic:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material_number</td>
<td>integer</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>material_type</td>
<td>string</td>
<td></td>
<td>&quot;MaterialElastoplastic&quot;</td>
<td>type of the material</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;MaterialElastoplastic&quot;</td>
<td>name of the material</td>
</tr>
</tbody>
</table>

Graphics

| Graphics.color     | vector  | [0, 0, 1] | material color (as yet used with FEMesh, only) |

Solid
### Solid Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid.plane</td>
<td>bool</td>
<td>0</td>
<td>true: 2D, false: 3D</td>
</tr>
<tr>
<td>Solid.plane_stress</td>
<td>bool</td>
<td>0</td>
<td>for 2D-Elements only; 1: plane stress, 0: plane strain</td>
</tr>
<tr>
<td>Solid.density</td>
<td>double</td>
<td>0</td>
<td>material density, must not be set for static problems</td>
</tr>
<tr>
<td>Solid.youngs_modulus</td>
<td>double</td>
<td>0</td>
<td>Youngs modulus</td>
</tr>
<tr>
<td>Solid.poisson_ratio</td>
<td>double</td>
<td>0</td>
<td>Poisson ratio</td>
</tr>
<tr>
<td>Solid.is_orthotropic_material</td>
<td>bool</td>
<td>0</td>
<td>check to enable orthotropic parameters</td>
</tr>
<tr>
<td>Solid.Orthotropic.E1</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 1</td>
</tr>
<tr>
<td>Solid.Orthotropic.E2</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 2</td>
</tr>
<tr>
<td>Solid.Orthotropic.E3</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 3</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU12</td>
<td>double</td>
<td>0</td>
<td>poisson ration 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU13</td>
<td>double</td>
<td>0</td>
<td>poisson ration 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU23</td>
<td>double</td>
<td>0</td>
<td>poisson ration 23</td>
</tr>
<tr>
<td>Solid.Orthotropic.G12</td>
<td>double</td>
<td>0</td>
<td>shear modulus 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.G13</td>
<td>double</td>
<td>0</td>
<td>shear modulus 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.G23</td>
<td>double</td>
<td>0</td>
<td>shear modulus 23</td>
</tr>
<tr>
<td>Damping.c_M</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (C = c_M * M + c_K * K)</td>
</tr>
<tr>
<td>Damping.c_K</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (C = c_M * M + c_K * K)</td>
</tr>
</tbody>
</table>

### Inelasticity

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inelasticity.inelasticity_solution_method</td>
<td>string</td>
<td>&quot;fixed_point&quot;</td>
<td>fixed_point, return_mapping (see Simo and Hughes, Computational Inelasticity 1998)</td>
</tr>
<tr>
<td>Inelasticity.yield_stress</td>
<td>double</td>
<td>0</td>
<td>Yield Stress S_y, used for yield criterion: Mises(S) := sqrt(2/3)*</td>
</tr>
<tr>
<td>Inelasticity.tangent_module</td>
<td>double</td>
<td>0</td>
<td>Tangent module K used in plastic zones, i.e. K = d(Mises(S))/d</td>
</tr>
<tr>
<td>Inelasticity.nr_inelastic_variables</td>
<td>integer</td>
<td>R 7</td>
<td>number of inelastic variables used by material class</td>
</tr>
</tbody>
</table>

### 3.5.4 MaterialElastoplasticThermalExpansion

**Short description**

MaterialElastoplasticThermalExpansion is an elastoplastic material which considers thermal expansion.

**Data objects of MaterialElastoplasticThermalExpansion:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>Type</th>
<th>R</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material_number</td>
<td>integer</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>material_type</td>
<td>string</td>
<td></td>
<td>&quot;MaterialElastoplasticThermalExpansion&quot;</td>
<td>type of the material</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;MaterialElastoplasticThermalExpansion&quot;</td>
<td>name of the material</td>
</tr>
</tbody>
</table>

### Graphics

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.color</td>
<td>vector</td>
<td>[0, 0, 1]</td>
<td>material color (as yet used with FEMesh, only)</td>
</tr>
</tbody>
</table>

### Solid

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid.plane</td>
<td>bool</td>
<td>0</td>
<td>true: 2D, false: 3D</td>
</tr>
</tbody>
</table>
### 3.5. MATERIAL

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid.plane_stress</td>
<td>bool</td>
<td>0</td>
<td>for 2D-Elements only; 1: plane stress, 0: plane strain</td>
</tr>
<tr>
<td>Solid.density</td>
<td>double</td>
<td>0</td>
<td>material density, must not be set for static problems</td>
</tr>
<tr>
<td>Solid.youngs_modulus</td>
<td>double</td>
<td>0</td>
<td>Youngs modulus</td>
</tr>
<tr>
<td>Solid.poisson_ratio</td>
<td>double</td>
<td>0</td>
<td>Poisson ratio</td>
</tr>
<tr>
<td>Solid.is_orthotropic_material</td>
<td>bool</td>
<td>0</td>
<td>check to enable orthotropic parameters</td>
</tr>
<tr>
<td>Solid.Orthotropic.E1</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 1</td>
</tr>
<tr>
<td>Solid.Orthotropic.E2</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 2</td>
</tr>
<tr>
<td>Solid.Orthotropic.E3</td>
<td>double</td>
<td>0</td>
<td>elasticity along axis 3</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU12</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU13</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.NU23</td>
<td>double</td>
<td>0</td>
<td>poisson ratio 23</td>
</tr>
<tr>
<td>Solid.Orthotropic.G12</td>
<td>double</td>
<td>0</td>
<td>shear modulus 12</td>
</tr>
<tr>
<td>Solid.Orthotropic.G13</td>
<td>double</td>
<td>0</td>
<td>shear modulus 13</td>
</tr>
<tr>
<td>Solid.Orthotropic.G23</td>
<td>double</td>
<td>0</td>
<td>shear modulus 23</td>
</tr>
<tr>
<td>Damping.c_M</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (C = c_M * M + c_K * K)</td>
</tr>
<tr>
<td>Damping.c_K</td>
<td>double</td>
<td>0</td>
<td>used for Rayleigh damping (C = c_M * M + c_K * K)</td>
</tr>
<tr>
<td>Inelasticity.inelasticity_solution_method</td>
<td>string</td>
<td>&quot;fixed_point&quot;</td>
<td>fixed_point, return_mapping (see Simo and Hughes, Computational Inelasticity 1998)</td>
</tr>
<tr>
<td>Inelasticity.yield_stress</td>
<td>double</td>
<td>0</td>
<td>Yield Stress S_y, used for yield criterion: Mises(S) := sqrt(2./3.)*dev S &lt;= S_y, with S denoting 2PK stress tensor</td>
</tr>
<tr>
<td>Inelasticity.tangent_module</td>
<td>double</td>
<td>0</td>
<td>Tangent module K used in plastic zones, i.e. K = d(Mises(S))/d[Ep], corresponds to formerly used module of hardening H = sqrt(K)/S_y</td>
</tr>
<tr>
<td>Inelasticity.nr_inelastic_variables</td>
<td>integer</td>
<td>R 7</td>
<td>number of inelastic variables used by material class</td>
</tr>
<tr>
<td>ThermalExpansion.delta_T</td>
<td>double</td>
<td>0</td>
<td>temperature difference to reference material</td>
</tr>
<tr>
<td>ThermalExpansion.coefficient</td>
<td>double</td>
<td>0</td>
<td>thermal expansion coefficient</td>
</tr>
</tbody>
</table>
3.6 BeamProperties

These beam properties are available:

- Beam3DProperties, 3.6.1

Note:
In HOTINT several classes are treated as 'material'. BeamProperties are also 'materials', and can therefore be edited and deleted in the GUI with the menu items of the materials.
In the script language the command `AddBeamProperties` has to be used for the beam properties in the list above.

3.6.1 Beam3DProperties

Short description

Beam3DProperties defines material and geometric properties for beam structural finite elements.

Additional notes

First, specify the cross_section_type of the beam, which may be either rectangular (if set to 1), circular (if set to 2), or tubular (if set to 3) or polygonal (if set to 4). In either case the cross_section_size is a vector of 2, 1, 2, or 2n entries, where n confers to the number of vertices of a closed polygon. Then specify the stiffnesses and moments of inertias, as they are need by your beam and problem.

Data objects of Beam3DProperties:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material_number</td>
<td>integer</td>
<td>R</td>
<td>4</td>
<td>Material number</td>
</tr>
<tr>
<td>material_type</td>
<td>string</td>
<td></td>
<td>&quot;Beam3DProperties&quot;</td>
<td>type of the material</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Beam3DProperties&quot;</td>
<td>name of the material</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 1]</td>
<td>material color (as yet used with FEMesh, only)</td>
</tr>
<tr>
<td>Solid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solid.density</td>
<td>double</td>
<td></td>
<td>0</td>
<td>material density, must not be set for static problems</td>
</tr>
<tr>
<td>Solid.youngs_modulus</td>
<td>double</td>
<td></td>
<td>0</td>
<td>Youngs modulus</td>
</tr>
<tr>
<td>Solid.poisson_ratio</td>
<td>double</td>
<td></td>
<td>0</td>
<td>Poisson ratio</td>
</tr>
<tr>
<td>Damping</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Damping.c_M</td>
<td>double</td>
<td></td>
<td>0</td>
<td>used for Rayleigh damping ( C = c_M * M + c_K * K )</td>
</tr>
<tr>
<td>Damping.c_K</td>
<td>double</td>
<td></td>
<td>0</td>
<td>used for Rayleigh damping ( C = c_M * M + c_K * K )</td>
</tr>
<tr>
<td>Inelasticity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inelasticity.inelasticity_solution_method</td>
<td>string</td>
<td></td>
<td>&quot;fixed_point&quot;</td>
<td>fixed_point, return_mapping (see Simo and Hughes, Computational Inelasticity 1998)</td>
</tr>
<tr>
<td>Inelasticity.yield_stress</td>
<td>double</td>
<td></td>
<td>0</td>
<td>Yield Stress s_y, e.g.,</td>
</tr>
<tr>
<td>Inelasticity.tangent_module</td>
<td>double</td>
<td></td>
<td>0</td>
<td>Modulus of hardening H</td>
</tr>
<tr>
<td>Inelasticity.nr_inelastic_variables</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of inelastic variables used by material class</td>
</tr>
</tbody>
</table>
### 3.6. BEAMPROPERTIES

<table>
<thead>
<tr>
<th>cross_section_type</th>
<th>integer</th>
<th>1</th>
<th>1: rectangular, 2: circular, 3: tubular, 4: polygonal</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross_section_size</td>
<td>vector</td>
<td>[0, 0]</td>
<td>vector length of cross_section_size depends on cross_section_type: length 1 for circular cross section, length 2 for rectangular cross section (y and z extension) or tubular cross section (outer and inner diameter), and length 2*n for polygonal cross section (p1y,p1z,p2y,p2z,...,pnx,pnz)</td>
</tr>
<tr>
<td>cross_section_area</td>
<td>double</td>
<td>R 0</td>
<td>area of the beam’s cross section</td>
</tr>
<tr>
<td>Ix</td>
<td>double</td>
<td>R 0</td>
<td>polar moment of inertia of the beam’s cross section</td>
</tr>
<tr>
<td>Iy</td>
<td>double</td>
<td>R 0</td>
<td>area moment of inertia of the beam’s cross section w.r.t. y-axis (2D-beam)</td>
</tr>
<tr>
<td>Iz</td>
<td>double</td>
<td>R 0</td>
<td>area moment of inertia of the beam’s cross section w.r.t. z-axis</td>
</tr>
<tr>
<td>EA</td>
<td>double</td>
<td>-1</td>
<td>Young’s modulus * area</td>
</tr>
<tr>
<td>EIy</td>
<td>double</td>
<td>-1</td>
<td>bending stiffness w.r.t. y-axis (2D-beam)</td>
</tr>
<tr>
<td>EIz</td>
<td>double</td>
<td>-1</td>
<td>bending stiffness w.r.t. z-axis</td>
</tr>
<tr>
<td>GAKy</td>
<td>double</td>
<td>-1</td>
<td>shear stiffness including shear correction factor ky (2D-beam)</td>
</tr>
<tr>
<td>GAKz</td>
<td>double</td>
<td>-1</td>
<td>shear stiffness including shear correction factor kz</td>
</tr>
<tr>
<td>GJkx</td>
<td>double</td>
<td>-1</td>
<td>torsional stiffness including shear correction factor kx</td>
</tr>
<tr>
<td>RhoA</td>
<td>double</td>
<td>-1</td>
<td>density * area</td>
</tr>
<tr>
<td>RhoIx</td>
<td>double</td>
<td>-1</td>
<td>density * second area of moment w.r.t. x-axis</td>
</tr>
<tr>
<td>RhoIy</td>
<td>double</td>
<td>-1</td>
<td>density * second area of moment w.r.t. y-axis (2D-beam)</td>
</tr>
<tr>
<td>RhoIz</td>
<td>double</td>
<td>-1</td>
<td>density * second area of moment w.r.t. z-axis</td>
</tr>
</tbody>
</table>

#### Example

see file Beam3DProperties.txt

bp
{
    material_type= "Beam3DProperties"
    cross_section_type= 1
    cross_section_size= [0.1, 0.1]
    EA= 2100000000
    EIy= 1750000
    EIz= 1750000
    GJkx= 2692307.692307693
}
nBeamProperties = AddBeamProperties(bp)
3.7 Node

These nodes are available:

- Node3D, \textcolor{red}{3.7.1}
- Node3DS1rot1, \textcolor{red}{3.7.2}
- Node3DS2S3, \textcolor{red}{3.7.3}
- Node3DRxyz, \textcolor{red}{3.7.4}
- Node3DR123, \textcolor{red}{3.7.5}
- Node3DS1S2, \textcolor{red}{3.7.6}

Note:
In HOTINT different types of nodes exist. The main difference between these types are the number of degrees of freedom. Depending on the chosen type of an element, the correct node has to be used. Each element provides some information about the needed nodes. In the script language the command \texttt{AddNode} is used for adding a node to the system.

3.7.1 Node3D

Short description

Node3D is the basic finite element node in 3D. It owns a reference position in 3D, and 3 degrees of freedom resembling the displacement in 3D.

Degrees of freedom

This node provides three degrees of freedom, all of which are components of the displacement vector \( u = (q_1, q_2, q_3)^T \) measured in the global frame of the multibody system.

Geometry

The geometry of the node is defined by its current position \( r \) measured in the global frame of the multibody system, which is the sum of the user defined reference position \( r_0 \) and the displacement vector \( u \), which is composed of the nodal degrees of freedom.

Data objects of Node3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td>&quot;Node3D&quot;</td>
<td>specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;Node&quot;</td>
<td>Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>1</td>
<td>Node Number.</td>
</tr>
<tr>
<td>Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reference_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Position (2D/3D) in reference configuration.</td>
</tr>
<tr>
<td>Initialization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node_initial_values</td>
<td>vector</td>
<td>[0, 0, 0, 0, 0]</td>
<td>initial values for all degrees of freedom of node</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGB_color</td>
<td>vector</td>
<td>[0.4, 0.4, 0.1]</td>
<td>red, green, blue color of element, range = 0..1,</td>
</tr>
<tr>
<td>visible</td>
<td>integer</td>
<td>1</td>
<td>Visibility of node.</td>
</tr>
</tbody>
</table>
Example
see file Node3D.txt

node
{
    node_type = "Node3D"
    Geometry.reference_position = [0,0,0]
}
nNode1 = AddNode(node)

3.7.2 Node3DS1rot1
Short description
Node3DS1rot1 is a finite element node for elements in 3D, and provides 7 degrees of freedom.

Degrees of freedom
This node provides 7 degrees of freedom: the first 3 degrees of freedom are the displacement
\( (q_1, q_2, q_3)^T = u = r - r_0 \), the next 3 DOFs denote the change of the first slope, which is the
partial derivative of the position \( (q_4, q_5, q_6)^T = r_\xi - r_0\xi \) with \( \xi \) denoting the first of the three
coordinates \((\xi, \eta, \zeta)\) of the reference element, and the 7th degree of freedom is the local rotation
\( q_7 = \theta \) of the node around its current direction \( S_1 \).

Geometry
The reference geometry of the node is defined by the user via (a) \( \text{Geometry}.\text{reference}\_position \)
and (b) the rotation \( \text{Geometry}.\text{reference}\_rot\_angles \). The rotation is prescribed by the user
in form of kardan angles (initially, local \((S_1, S_2, S_3)\) and global frame \((x, y, z)\) are identical, then
rotate local frame around \( S_1 \), then \( S_2 \) and finally \( S_3 \)). The current position is evaluated by
adding displacement (the first three degrees of freedom) to the reference position of the node
(degrees of freedom: \( (q_1, q_2, q_3)^T \)), and the current rotation of the node is obtained by adding
the change of the first axis of the local frame (DOFs: \( (q_4, q_5, q_6)^T \)) to the first axis of the local
frame in reference configuration of the node, and finally rotating the two other axes around the
first axis of the local frame by the amount of the 7th degree of freedom \( q_7 = \theta \).

Data objects of Node3DS1rot1:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;Node3DS1rot1&quot; specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;Node3DS1rot1&quot; Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Node Number.</td>
</tr>
<tr>
<td>Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry.</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Position (2D/3D) in reference configuration.</td>
</tr>
<tr>
<td>reference_position</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry.</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Kardan rotation angles (X,Y,Z) in rad in global frame of node in reference configuration.</td>
</tr>
<tr>
<td>reference_rot_angles</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Initialization:

**node_initial_values**

vector

\[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\]

*initial values for all degrees of freedom of node*

**Graphics**

<table>
<thead>
<tr>
<th>Graphics.RGB_color</th>
<th>vector</th>
<th>[0.4, 0.4, 0.1]</th>
<th>red, green, blue color of element, range = 0..1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.visible</td>
<td>integer</td>
<td>1</td>
<td>Visibility of node.</td>
</tr>
</tbody>
</table>

**Example**

see file Node3DS1rot1.txt

```java
node
{
  node_type= "Node3DS1rot1"
  Geometry
  {
    reference_position= [0, 0, 0]
    reference_rot_angles= [0, 0, 0]
  }
}
```

```
noNode1 = AddNode(node)
```

### 3.7.3 Node3DS2S3

**Short description**

Node3DS2S3 is a finite element node for elements in 3D, and provides 9 degrees of freedom.

**Degrees of freedom**

This node provides 9 degrees of freedom: the first 3 degrees of freedom are the displacement \((q_1, q_2, q_3)^T = u = r - r_0\), the next 3 DOFs denote the change of the *second slope*, which are the partial derivatives of the position \((q_4, q_5, q_6)^T = r_{\eta} - r_{0,\eta}\) and \((q_7, q_8, q_9)^T = r_{\zeta} - r_{0,\zeta}\), where \(\eta\) and \(\zeta\) denote the second and third of the three coordinates \((\xi, \eta, \zeta)\) of the reference element.

**Geometry**

The reference geometry of the node is defined by the user via (a) Geometry.reference_position and (b) the slopes Geometry.ref_slope2 and Geometry.ref_slope3. The current position is evaluated by adding the displacement (the first three degrees of freedom \((q_1, q_2, q_3)^T\)) to the reference position of the node, and further the current slopes of the node are obtained by adding the change of the second and third slopes (DOFs: \((q_4, q_5, q_6)^T\) and \((q_7, q_8, q_9)^T\)) to the second and third slopes in reference configuration of the node.

**Data objects of Node3DS2S3:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;Node3DS2S3&quot;</td>
<td>specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Node3DS2S3&quot;</td>
<td>Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>Node Number.</td>
</tr>
</tbody>
</table>
3.7. NODE

3.7.4 Node3DRxyz

Short description

Node3DRxyz is a finite element node in 3D. It has a 3D reference position, a reference orientation described by bryant angles and 6 degrees of freedom.

Degrees of freedom

The first 3 degrees of freedom are used to describe the displacement \( (q_1, q_2, q_3)^T = \mathbf{u} = \mathbf{r} - \mathbf{r}_0 \), the last 3 are used for the description of linearized (small) angles \( (\phi_x, \phi_y, \phi_z)^T \). All degrees of freedom are w.r.t. the global coordinate system.

Geometry

The reference position of the node is defined by the user via `Geometry.reference_position` and the reference orientation via `Geometry.reference_rot_angles`. The current position is evaluated by adding the displacement (the first three degrees of freedom \( (q_1, q_2, q_3)^T \)) to the reference position of the node.

Data objects of Node3DRxyz:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td></td>
<td>&quot;Node3DRxyz&quot;</td>
<td>specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Node3DRxyz&quot;</td>
<td>Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>Node Number.</td>
</tr>
<tr>
<td>Geometry.reference_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Position (2D/3D) in reference configuration.</td>
<td></td>
</tr>
<tr>
<td>Geometry.reference_rot_angles</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Kardan rotation angles (X,Y,Z) in rad in global frame of node in reference configuration.</td>
<td></td>
</tr>
</tbody>
</table>

Initialization

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_initial_values</td>
<td>vector</td>
<td>[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>initial values for all degrees of freedom of node</td>
<td></td>
</tr>
</tbody>
</table>

Graphics

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.4, 0.4, 0.1]</td>
<td>[red, green, blue] color of element, range = 0..1,</td>
<td></td>
</tr>
<tr>
<td>Graphics.visible</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>Visibility of node.</td>
</tr>
</tbody>
</table>
### 3.7.5 Node3DR123

**Short description**

Node3DR123 is a finite element node in 3D. It has a 3D reference position, a reference orientation described by bryant angles and 6 degrees of freedom.

**Degrees of freedom**

The first 3 degrees of freedom are used to describe the displacement \((q_1, q_2, q_3)^T = u = r - r_0\), the last 3 are used for the description of linearized (small) angles \((\phi_x, \phi_y, \phi_z)^T\). All degrees of freedom are w.r.t. the reference coordinate system of the node.

**Geometry**

The reference position of the node is defined by the user via `Geometry.reference_position` and the orientation via `Geometry.reference_rot_angles`. The current position is evaluated by adding the displacement (the first three degrees of freedom \((q_1, q_2, q_3)^T\) transformed into the global coordinate system) to the reference position of the node.

#### Data objects of Node3DR123:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;Node3DR123&quot;</td>
<td>specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>R</td>
<td>&quot;Node3DR123&quot;</td>
<td>Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>Node Number.</td>
</tr>
<tr>
<td>Geometry.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reference_position</td>
<td>vector</td>
<td>R</td>
<td>[0, 0, 0]</td>
<td>Position (2D/3D) in reference configuration.</td>
</tr>
<tr>
<td>Geometry.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reference_rot_angles</td>
<td>vector</td>
<td>R</td>
<td>[0, 0, 0]</td>
<td>Kardan rotation angles (X,Y,Z) in rad in global frame of node in reference configuration.</td>
</tr>
<tr>
<td>Initialization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node_initial_values</td>
<td>vector</td>
<td>R</td>
<td>[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>initial values for all degrees of freedom of node</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>R</td>
<td>[0.4, 0.4, 0.1]</td>
<td>red, green, blue color of element, range = 0.1,</td>
</tr>
<tr>
<td>Graphics.visible</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>Visibility of node.</td>
</tr>
</tbody>
</table>

### 3.7.6 Node3DS1S2

**Short description**

Node3DS1S2 is a finite element node for elements in 3D, and provides 9 degrees of freedom.

**Degrees of freedom**

This node provides 9 degrees of freedom: the first 3 degrees of freedom are the displacement \((q_1, q_2, q_3)^T = u = r - r_0\), the next 3 DOFs denote the change of the first slope, which are the partial derivatives of the position \((q_4, q_5, q_6)^T = r_{\xi} - r_{0,\xi}\) and \((q_7, q_8, q_9)^T = r_{\eta} - r_{0,\eta}\), where \(\xi\) and \(\eta\) denote the first and second of the three coordinates \((\xi, \eta, \zeta)\) of the reference element.
3.7. NODE

Geometry
The reference geometry of the node is defined by the user via (a) Geometry.reference_position and (b) the slopes Geometry.ref_slope1 and Geometry.ref_slope2. The current position is evaluated by adding the displacement (the first three degrees of freedom \((q_1, q_2, q_3)^T\)) to the reference position of the node, and further the current slopes of the node are obtained by adding the change of the first and second slopes (DOFs: \((q_4, q_5, q_6)^T\) and \((q_7, q_8, q_9)^T\)) to the first and second slopes in reference configuration of the node.

Data objects of Node3DS1S2:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_type</td>
<td>string</td>
<td></td>
<td>&quot;Node3DS1S2&quot;</td>
<td>specification of node type. Once the node is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Node3DS1S2&quot;</td>
<td>Node identifier.</td>
</tr>
<tr>
<td>node_number</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>Node Number.</td>
</tr>
<tr>
<td>Geometry.reference_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>Position (2D/3D) in reference configuration.</td>
<td></td>
</tr>
<tr>
<td>Geometry.reference_slope1</td>
<td>vector</td>
<td>[1, 0, 0]</td>
<td>slope 1 of node in reference configuration.</td>
<td></td>
</tr>
<tr>
<td>Geometry.reference_slope2</td>
<td>vector</td>
<td>[0, 1, 0]</td>
<td>slope 2 of node in reference configuration.</td>
<td></td>
</tr>
</tbody>
</table>

Initialization

| Initialization              | vector | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] | initial values for all degrees of freedom of node                                                                                         |

Graphics

| Graphics.RGB_color         | vector | [0.4, 0.4, 0.1] | red, green, blue color of element, range = 0..1,                                                                                       |
| Graphics.visible          | integer | 1               | Visibility of node.                                                                                                                      |
3.8 Load

These loads are available:
- GCLoad, 3.8.1
- BodyLoad, 3.8.2
- ForceVector2D, 3.8.3
- ForceVector3D, 3.8.4
- MomentVector3D, 3.8.5
- Gravity, 3.8.6
- SurfacePressure, 3.8.7
- BodyLoadSpatial, 3.8.8

For all loads it is possible to vary the value of the load with respect to time. The following options are available:
1. MathFunction
2. IOElement

3.8.0.1 MathFunction

The value $F(t)$ of a load at time $t$ is computed as:

$$ F(t) = f(t) \vec{F} $$

(3.55)

$f(t)$ represents the value of the MathFunction at time $t$, e.g. $f(t) = \sin(t)$.

$\vec{F}$ represents the (constant) force vector, if a force vector is used in the specific type of load, e.g. ForceVector3D.

If no force vector is available for the load, then the load is defined by $f(t)$ only. Any additional scalar value (e.g. load_value in GCLoad) is set to 1!

3.8.0.2 IOElement

The value $F(t)$ of a load at time $t$ is computed as:

$$ F(t) = f(t) \vec{F} $$

(3.56)

$f(t)$ represents the value of the output of the IOElement at time $t$. By the use of IOElements it is possible to define loads, that are not only dependent on time, but on any possible input of an IOElement.

$\vec{F}$ represents the (constant) force vector, if a force vector is used in the specific type of load, e.g. ForceVector3D.

If no force vector is available for the load, then the load is defined by $f(t)$ only. Any additional scalar value (e.g. load_value in GCLoad) is set to 1!

3.8.1 GCLoad

A load acting on a generalized coordinate (gc) of the element.

Data objects of GCLoad:
3.8. LOAD

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;GCLoad&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>generalized_coordinate</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>(local) number of the generalized coordinate</td>
</tr>
<tr>
<td>load_value</td>
<td>double</td>
<td></td>
<td>0</td>
<td>value of the load acting in the direction of generalized_coordinate</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0..constant, 1..MathFunction, 2..IOElement</td>
</tr>
<tr>
<td>MathFunction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MathFunction.piecewise_mode</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
</tr>
<tr>
<td>MathFunction.piecewise_points</td>
<td>vector</td>
<td></td>
<td></td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
<tr>
<td>MathFunction.piecewise_values</td>
<td>vector</td>
<td></td>
<td></td>
<td>values at supporting points</td>
</tr>
<tr>
<td>MathFunction.piecewise_diff_values</td>
<td>vector</td>
<td></td>
<td></td>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td>MathFunction.parsed_function</td>
<td>string</td>
<td></td>
<td></td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>MathFunction.parsed_function_parameter</td>
<td>string</td>
<td></td>
<td></td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
<tr>
<td>IOElement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOElement.input_element_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of IOElement in the mbs</td>
</tr>
<tr>
<td>IOElement.input_local_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of output of IOElement connected to this element</td>
</tr>
</tbody>
</table>

Example

see file GCLoad.txt

myLoad % define the load
{
    load_type = "GCLoad"
    generalized_coordinate = 1 % (local) number of the generalized coordinate
    load_value = 10
}

nLoad=AddLoad(myLoad)

emptyMass3D % define some element
{
    element_type = "Mass3D"
    Physics.mass= 1
    loads = [nLoad] % add the load to the element
}

nElement = AddElement(emptyMass3D)
3.8.2 BodyLoad

The load value is integrated over the volume of the body and applied to the body in the specified direction. For the case of a rigid body, a force of size \( \text{load}_\text{value} = \text{density} \times \text{gravity}_\text{constant} \) applies a force according to the gravitational force.

**Data objects of BodyLoad:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;BodyLoad&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>direction</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>direction of the load</td>
</tr>
<tr>
<td>load_value</td>
<td>double</td>
<td></td>
<td>0</td>
<td>value of the load acting</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0..constant, 1..MathFunction, 2..IOElement</td>
</tr>
</tbody>
</table>

**MathFunction**

- **piecewise_mode** integer -1 modus for piecewise interpolation: -1 = not piecewise, 0 = constant, 1 = linear, 2 = quadratic
- **piecewise_points** vector \([\]\) supporting points (e.g. time or place) for piecewise interpolation
- **piecewise_values** vector \([\]\) values at supporting points
- **piecewise_diff_values** vector \([\]\) differential values at supporting points - for quadratic interpolation
- **parsed_function** string "\[\]" string representing parsed function, e.g. \("A*sin(omega*t)\)"
- **parsed_function_parameter** string "\[\]" string representing parameter of parsed function, e.g. \("t\)"

**IOElement**

- **input_element_number** integer 0 number of IOElement in the mbs
- **input_local_number** integer 0 number of output of IOElement connected to this element

3.8.3 ForceVector2D

**Data objects of ForceVector2D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;ForceVector2D&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>force_vector</td>
<td>vector</td>
<td></td>
<td>([0, 0])</td>
<td>defines the magnitude and direction of the force</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td></td>
<td>([0, 0])</td>
<td>(local) position where the force is applied to the element</td>
</tr>
<tr>
<td>local_force</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>flag which describes, if local or global coordinate system is used: 1 = force in local body coordinate system, 0 = global force</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0..constant, 1..MathFunction, 2..IOElement</td>
</tr>
</tbody>
</table>
### MathFunction

<table>
<thead>
<tr>
<th>MathFunction.</th>
<th>piecewise_mode</th>
<th>integer</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathFunction.</td>
<td>piecewise_points</td>
<td>vector</td>
<td>[]</td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
<tr>
<td>MathFunction.</td>
<td>piecewise_values</td>
<td>vector</td>
<td>[]</td>
<td>values at supporting points</td>
</tr>
<tr>
<td>MathFunction.</td>
<td>piecewise_diff_values</td>
<td>vector</td>
<td>[]</td>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td>MathFunction.</td>
<td>parsed_function</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>MathFunction.</td>
<td>parsed_function_parameter</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
</tbody>
</table>

### IOElement

<table>
<thead>
<tr>
<th>IOElement.</th>
<th>input_element_number</th>
<th>integer</th>
<th>0</th>
<th>number of IOElement in the mbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOElement.</td>
<td>input_local_number</td>
<td>integer</td>
<td>0</td>
<td>number of output of IOElement connected to this element</td>
</tr>
</tbody>
</table>

---

**Example**

see file ForceVector2D.txt

```
myLoad  % define the load
{
  load_type = "ForceVector2D"
  force_vector = [10,0] % magnitude and direction
}

nLoad=AddLoad(myLoad)

L_x = 0.10 % length
L_y = 0.20 % width
L_z = 0.01 % height (for drawing and computation of mass)
density= 7850

myRigid2D % add rigid body
{
  element_type= "Rigid2D" %specification of element type.
  name= "R2D" %name of the element
  Graphics.body_dimensions = [L_x, L_y, 0]
  loads = [nLoad] % add the load to the element
  Physics
  {
    mass= density*L_x*L_y*L_z
    moment_of_inertia= 1.0/12.0*mass*(L_x^2+L_y^2)
  }
  Initialization
  {
    initial_position= [0, 0] %[X, Y]
    initial_rotation= [0.0] %rot1_Z in rad
    initial_velocity= [0, 0] %[X, Y]
    initial_angular_velocity= [0] %rad/s
  }
```
3.8.4 ForceVector3D

A load acting on an element at a specified (local) position.

**Data objects of ForceVector3D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;ForceVector3D&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>force_vector</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>defines the magnitude and direction of the force</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>(local) position where the force is applied to the element</td>
</tr>
<tr>
<td>local_force</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>flag which describes, if local or global coordinate system is used: 1 = force in local body coordinate system, 0 = global force</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0..constant, 1..MathFunction, 2..IOElement</td>
</tr>
</tbody>
</table>

**MathFunction**

- **piecewise_mode**
  - type: integer
  - default: -1
  - description: modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic
- **piecewise_points**
  - type: vector
  - default: []
  - description: supporting points (e.g. time or place) for piecewise interpolation
- **piecewise_values**
  - type: vector
  - default: []
  - description: values at supporting points
- **piecewise_diff_values**
  - type: vector
  - default: []
  - description: differential values at supporting points - for quadratic interpolation
- **parsed_function**
  - type: string
  - default: ""
  - description: string representing parsed function, e.g. 'A*sin(omega*t)'
- **parsed_function_parameter**
  - type: string
  - default: ""
  - description: string representing parameter of parsed function, e.g. 't'

**IOElement**

- **input_element_number**
  - type: integer
  - default: 0
  - description: number of IOElement in the mbs
- **input_local_number**
  - type: integer
  - default: 0
  - description: number of output of IOElement connected to this element

**Example**

see file ForceVector3D.txt

```plaintext
myLoad  % define the load
{
    load_type = "ForceVector3D"
    force_vector = [10,0,0]  % magnitude and direction
}
myLoad=AddLoad(myLoad)
```
emptyMass3D % define some element
{
    element_type = "Mass3D"
    Physics.mass = 1
    loads = [nLoad] % add the load to the element
}

nElement = AddElement(emptyMass3D)

ViewingOptions.Loads.show_loads = 1
ViewingOptions.Loads.arrow_size = 0.2

### 3.8.5 MomentVector3D

A torque acting on an element at a specified (local) position.

#### Data objects of MomentVector3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;MomentVector3D&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>moment_vector</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>defines the magnitude and direction of the moment</td>
</tr>
<tr>
<td>position</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>(local) position where the moment is applied to the element</td>
</tr>
<tr>
<td>local_moment</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>flag which describes, if local or global coordinate system is used: 1 = moment in local body coordinate system, 0 = global moment</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0..constant, 1..MathFunction, 2..IOElement</td>
</tr>
</tbody>
</table>

**MathFunction**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathFunction.piecewise_mode</td>
<td>integer</td>
<td>-1</td>
<td>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
<td></td>
</tr>
<tr>
<td>MathFunction.piecewise_points</td>
<td>vector</td>
<td></td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
<td></td>
</tr>
<tr>
<td>MathFunction.piecewise_values</td>
<td>vector</td>
<td></td>
<td>values at supporting points</td>
<td></td>
</tr>
<tr>
<td>MathFunction.piecewise_diff_values</td>
<td>vector</td>
<td></td>
<td>differential values at supporting points - for quadratic interpolation</td>
<td></td>
</tr>
<tr>
<td>MathFunction.parsed_function</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
<td></td>
</tr>
<tr>
<td>MathFunction.parsed_function_parameter</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>string representing parameter of parsed function, e.g. 't'</td>
<td></td>
</tr>
</tbody>
</table>

**IOElement**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOElement.input_element_number</td>
<td>integer</td>
<td>0</td>
<td>number of IOElement in the mbs</td>
<td></td>
</tr>
<tr>
<td>IOElement.input_local_number</td>
<td>integer</td>
<td>0</td>
<td>number of output of IOElement connected to this element</td>
<td></td>
</tr>
</tbody>
</table>
3.8.6 Gravity

The load is integrated over the volume of the body and applied to the body in the specified direction. The density of the body is used to compute the force.

Data objects of Gravity:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;Gravity&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>direction</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>global direction of the gravity</td>
</tr>
<tr>
<td>gravity_constant</td>
<td>double</td>
<td></td>
<td>9.81</td>
<td>use negative sign if necessary</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0=constant, 1=..MathFunction, 2=IOElement</td>
</tr>
</tbody>
</table>

MathFunction

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>piecewise_mode</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</td>
</tr>
<tr>
<td>piecewise_points</td>
<td>vector</td>
<td></td>
<td></td>
<td>supporting points (e.g. time or place) for piecewise interpolation</td>
</tr>
<tr>
<td>piecewise_values</td>
<td>vector</td>
<td></td>
<td></td>
<td>values at supporting points</td>
</tr>
<tr>
<td>piecewise_diff_values</td>
<td>vector</td>
<td></td>
<td></td>
<td>differential values at supporting points - for quadratic interpolation</td>
</tr>
<tr>
<td>parsed_function</td>
<td>string</td>
<td></td>
<td></td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>parsed_function_parameter</td>
<td>string</td>
<td></td>
<td></td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
</tbody>
</table>

IOElement

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_element_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of IOElement in the mbs</td>
</tr>
<tr>
<td>input_local_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of output of IOElement connected to this element</td>
</tr>
</tbody>
</table>

Example

see file Gravity.txt

myLoad % define the load
{
    load_type = "Gravity"
    name = "gravity for all elements"
    direction = 2
    gravity_constant = 9.81
}

nLoad=AddLoad(myLoad)

emptyMass3D % define some element
{
    element_type = "Mass3D"
    Physics.mass= 1
    loads = [nLoad] % add the load to the element
3.8. LOAD

}nElement = AddElement(emptyMass3D)

ViewingOptions.Loads.show_loads = 1
ViewingOptions.Loads.arrow_size = 0.2

3.8.7 SurfacePressure

Data objects of SurfacePressure:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;Load&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td></td>
<td>&quot;SurfacePressure&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>direction</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>local surface (inner/outer)</td>
</tr>
<tr>
<td>surface_pressure</td>
<td>double</td>
<td></td>
<td>0</td>
<td>use negative sign if necessary</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>time dependency of the load: 0.constant, 1.MathFunction, 2.IOElement</td>
</tr>
</tbody>
</table>

MathFunction

MathFunction. 
piecewise_mode  integer  -1  modulus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic
MathFunction. 
piecewise_points vector  []  supporting points (e.g. time or place) for piecewise interpolation
MathFunction. 
piecewise_values vector  []  values at supporting points
MathFunction. 
piecewise_diff_values vector  []  differential values at supporting points - for quadratic interpolation
MathFunction. 
parsed_function string  ""  string representing parsed function, e.g. 'A*sin(omega*t)'  
MathFunction. 
parsed_function_parameter string  ""  string representing parameter of parsed function, e.g. 't'

IOElement

IOElement. 
input_element_number integer  0  number of IOElement in the mbs
IOElement. 
input_local_number  integer  0  number of output of IOElement connected to this element

3.8.8 BodyLoadSpatial

Data objects of BodyLoadSpatial:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;BodyLoadSpatial&quot;</td>
<td>name of the load</td>
</tr>
<tr>
<td>load_type</td>
<td>string</td>
<td>R</td>
<td>&quot;BodyLoadSpatial&quot;</td>
<td>specification of load type. Once the load is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the load in the mbs</td>
</tr>
<tr>
<td>direction</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>direction of the load</td>
</tr>
<tr>
<td>load_value</td>
<td>double</td>
<td></td>
<td>0</td>
<td>value of the load acting</td>
</tr>
<tr>
<td>load_function_type</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>time dependency of the load: 0=constant, 1=MathFunction, 2=IOElement</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---</td>
<td>---</td>
<td>------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**MathFunction**

<table>
<thead>
<tr>
<th>MathFunction.piecewise_mode</th>
<th>integer</th>
<th>R</th>
<th>-1</th>
<th>modus for piecewise interpolation: -1=not piecewise, 0=constant, 1=linear, 2=quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathFunction.parsed_function</td>
<td>string</td>
<td>R</td>
<td>&quot;&quot;</td>
<td>string representing parsed function, e.g. 'A<em>sin(omega</em>t)'</td>
</tr>
<tr>
<td>MathFunction.parsed_function_parameter</td>
<td>string</td>
<td>R</td>
<td>&quot;t,x,y,z&quot;</td>
<td>string representing parameter of parsed function, e.g. 't'</td>
</tr>
</tbody>
</table>
3.9 Sensor

These sensors are available:

- FVElementSensor, 3.9.1
- ElementSensor, 3.9.2
- LoadSensor, 3.9.3
- MultipleSensor, 3.9.4
- SystemSensor, 3.9.5
- FVGlobalPositionSensor, 3.9.6

In HOTINT it is possible to access all degrees of freedom and many more interesting values with sensors. In general these values are stored to a file at specified time steps. Many options concerning these settings are available in SolverOptions.Solution.

You can use the PlotTool to visualize the sensor values but it is also possible to import the solution file easily in other software for postprocessing.

If you want to modify sensor values online (e.g. convert the units from rad to degrees or subtract an offset) it is recommended to use ControlElements.

In the script language the command AddSensor is used to add a sensor to the system.

3.9.1 FVElementSensor

The FieldVariableElementSensor evaluates the value of a field variable at a specified position. There are two possibilities to define this position:

- element number + local position
- element number + local node number

The descriptions of the elements above include a list of available field variables for each element. Possible field variables are e.g.

- position, velocity and displacement
- bryant_angle and angular_velocity
- beam_axial_extension, beam_torsion, beam_curvature
- many more

Data objects of FVElementSensor:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>


**signal.Storage_Mode** integer 5 storage mode of the sensor signal (0 -> no data storage, 1 -> general solution file of the simulation, 2 -> separate file with the results of this particular sensor, 4 - internal array in memory - all time signal points)

**sensor_number** integer R 1 number of the sensor in the mbs

**name** string "sensor" name of the sensor for the output files and for the plot tool

**precision** integer -1 precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision)

**sensor.type** string "FVElemensSensor" specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!

**element_number** integer 1 number of the element, to which the sensor is applied

**node_number** integer 0 local node number. If > 0, then the position of this node is used.

**local_position** vector [0, 0, 0] local position at which the field variable is evaluated.

**field_variable** string "position" name of the field variable, e.g. 'position', see the documentation of the elements for the available field variables

**component** string "x" component of the field variable, e.g. 'x'

---

**Example**

see file FVElemensSensor.txt

```plaintext
emptyMass3D
{
  element_type = "Mass3D"
  Physics.mass= 1
}
```

```plaintext
nElement = AddElement(emptyMass3D)
```

```plaintext
sensor
{
  sensor_type= "FVElementSensor"
  element_number= nElement %number of the element
  field_variable= "position" %name of the field variable
  component= "x" %component of the field variable
}
```

```plaintext
nSensor = AddSensor(sensor)
```

---

**3.9.2 ElementSensor**

The ElementSensor returns special values evaluated in the element. It can be used e.g. for measuring a specific degree of freedom of an element. The descriptions of the elements above include a list of available special values for each element.

**Data objects of ElementSensor:**
3.9. SENSOR

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signal_Storage_Mode</td>
<td>integer</td>
<td>5</td>
<td></td>
<td>storage mode of the sensor signal (0 -&gt; no data storage, 1 -&gt; general solution file of the simulation, 2 -&gt; separate file with the results of this particular sensor, 4 - internal array in memory - all time signal points)</td>
</tr>
<tr>
<td>sensor_number</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>number of the sensor in the mbs</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;sensor&quot;</td>
<td>name of the sensor for the output files and for the plot tool</td>
</tr>
<tr>
<td>precision</td>
<td>integer</td>
<td>-1</td>
<td></td>
<td>precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision)</td>
</tr>
<tr>
<td>sensor_type</td>
<td>string</td>
<td></td>
<td>&quot;ElementSensor&quot;</td>
<td>specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>element_number</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>number of the element, to which the sensor is applied</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td></td>
<td>&quot;&quot;</td>
<td>special value of the element, use [] to access vector or matrix values, e.g. force[1] or stress[2,3]</td>
</tr>
</tbody>
</table>

Example

see file ElementSensor.txt

emptyMass3D
{
   element_type = "Mass3D"
   Physics.mass= 1
}
nElement = AddElement(emptyMass3D)

ElemSensor
{
   sensor_type= "ElementSensor"
   element_number= nElement
   value= "Internal.second_order_variable[1]"
}
nElemSensor = AddSensor(ElemSensor)

3.9.3 LoadSensor

The LoadSensor can be applied to loads in order to measure their time dependency. The value \( F(t) \) of a load at time \( t \) is computed (see the description of the loads for more details) as:

\[
F(t) = f(t)\vec{F} \tag{3.57}
\]

The LoadSensor returns the value of the factor \( f(t) \) and not the value \( F(t) \). If the LoadSensor is used for a scalar load (e.g. GCLoad), then \( f(t) \) and \( F(t) \) are equal. If the LoadSensor is used for a load vector (e.g. ForceVector3D) then \( f(t) \) and \( F(t) \) may not be equal.

The LoadSensor can not be shown in the graphical output, because the load does not have a position by itself and may be applied to several elements or nodes.

Data objects of LoadSensor:
### Data name

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signal_Storage_Mode</td>
<td>integer</td>
<td>5</td>
<td>storage mode of the sensor signal (0 -&gt; no data storage, 1 -&gt; general solution file of the simulation, 2 -&gt; separate file with the results of this particular sensor, 4 - internal array in memory - all time signal points)</td>
<td></td>
</tr>
<tr>
<td>sensor_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the sensor in the mbs</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;sensor&quot;</td>
<td>name of the sensor for the output files and for the plot tool</td>
</tr>
<tr>
<td>precision</td>
<td>integer</td>
<td>-1</td>
<td>precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision)</td>
<td></td>
</tr>
<tr>
<td>sensor_type</td>
<td>string</td>
<td></td>
<td>&quot;LoadSensor&quot;</td>
<td>specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>load_number</td>
<td>integer</td>
<td>1</td>
<td>number of the load, to which the sensor is applied</td>
<td></td>
</tr>
</tbody>
</table>

### Example

see file LoadSensor.txt

```plaintext
myLoad  % define the load
{
  load_type = "ForceVector3D"
  force_vector = [10,0,0]  % magnitude and direction
  load_function_type = 1  % time dependent load
  MathFunction
  {
    piecewise_mode = -1  %modus -1=not piecewise
    parsed_function = "sin(100*t)"  %string representing parsed function
    parsed_function_parameter = "t"  % parameter of parsed function
  }
}

load = AddLoad(myLoad)

emptyMass3D  % define some element
{
  element_type = "Mass3D"
  Physics.mass = 1
  loads = [load]  % add the load to the element
}

nElement = AddElement(emptyMass3D)

sensor
{
  sensor_type = "LoadSensor"
  load_number = load  %number of the load
}

nSensor = AddSensor(sensor)
```
3.9. SENSOR

3.9.4 MultipleSensor

The MultipleSensor applies mathematical operations to a list of sensors. The sensor can be used, e.g., to get the maximum or average value of a list of sensors. The following mathematical operations are possible (use these words for 'operation'):

- average
- minimum
- maximum
- sum
- norm
- norm2
- maxabs

If weights are used, then the value of each sensor is multiplied with the weight before the mathematical operation is performed. To compute a weighted sum of the first 4 sensors, the entries would be e.g. sensor_numbers = [1,2,3,4] and weights = [0.125,0.125,0.25,0.5].

Data objects of MultipleSensor:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>
| signal_Storage_Mode | integer |  5 |         | storage mode of the sensor signal (0 -> no data storage, 1 -> general solution file of the simulation, 2 -> separate file with the results of this particular sensor, 4 - internal array in memory - all time signal points)
| sensor_number       | integer |   |  1      | number of the sensor in the mbs                                             |
| name                | string  |   | "sensor"| name of the sensor for the output files and for the plot tool              |
| precision           | integer |   | -1      | precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision) |
| sensor_type         | string  |   | "MultipleSensor" | specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore! |
| sensor_numbers      | vector  |   |         | number of the sensors, that are used for computation                       |
| weights             | vector  |   |         | weights for e.g. a weighted sum. This vector must have the same length as sensor_numbers or must be empty! |
| operation           | string  |   | "maximum"| mathematical operation that is applied to the sensor values, e.g. 'maximum', 'average', ... |

3.9.5 SystemSensor

The SystemSensor can be applied to global degrees of freedom, eigenvalues, several iteration numbers or performance indicators. It returns the value of the specified quantity at time t, and
can not be shown in the graphical output, because a system quantity does in general not have a position by itself.

**Data objects of SystemSensor:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signal.Storage_Mode</td>
<td>integer</td>
<td>R</td>
<td>5</td>
<td>storage mode of the sensor signal (0 -&gt; no data storage, 1 -&gt; general solution file of the simulation, 2 -&gt; separate file with the results of this particular sensor, 4 - internal array in memory - all time signal points)</td>
</tr>
<tr>
<td>sensor_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the sensor in the mbs</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;SystemSensor&quot;</td>
<td>name of the sensor for the output files and for the plot tool</td>
</tr>
<tr>
<td>precision</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision)</td>
</tr>
<tr>
<td>sensor_type</td>
<td>string</td>
<td></td>
<td>&quot;SystemSensor&quot;</td>
<td>specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>object</td>
<td>string</td>
<td></td>
<td>&quot;none&quot;</td>
<td>Object tracked by systemsensor. Is either 'DOF' (global degree of freedom), 'EV' (global eigenvalue), 'jacobians', 'newton_iterations', 'newton_iterations_total', 'discontinuous_iterations', 'rhs_evaluations', 'rhs_evaluations_jacobian', 'volume', 'potential_energy', 'kinetic_energy', 'FE_color_minimum', 'FE_color_maximum', 'NNodes', 'NElements' or 'ElapsedTime'</td>
</tr>
<tr>
<td>global_index</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>Number of the global index. Has to be set if (and only if) object is 'DOF' or 'EV'.</td>
</tr>
<tr>
<td>set_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

see file SystemSensor.txt

myLoad  % define the load
{
  load_type = "ForceVector3D"
  force_vector = [10,0,0]  % magnitude and direction
  load_function_type = 1  % time dependent load
  MathFunction
  {
    piecewise_mode= -1  %modus -1=not piecewise
    parsed_function= "sin(100*t)"  %string representing parsed function
    parsed_function_parameter= "t"  % parameter of parsed function
  }
}

nLoad=AddLoad(myLoad)

eemptyMass3D  % define some element
{
  element_type = "Mass3D"
3.9. SENSOR

Physics.mass = 1
loads = [nLoad] % add the load to the element
}

nElement = AddElement(emptyMass3D)

Systemsensor
{
    name = "Systemsensor Jacobians" % name of the sensor for the output files and for the plot tool
    sensor_type = "SystemSensor" % specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!
    object = "jacobians" % object tracked by systemsensor. Is either 'DOF' (global degree of freedom), 'EV' (global eigenvalue), 'jacobians', 'newton_iterations', 'discontinuous_iterations', 'rhs_evaluations', or 'rhs_evaluations_jacobian'
    global_index = 0 % Number of the global index. Has to be set if (and only if) object is 'DOF'
}

AddSensor(Systemsensor)

Systemsensor.object = "rhs_evaluations"
Systemsensor.name = "Systemsensor RHS Evaluations"
AddSensor(Systemsensor)

Systemsensor.object = "DOF"
Systemsensor.global_index = 4
Systemsensor.name = "Systemsensor DOF 4"
AddSensor(Systemsensor)

3.9.6 FVGlobalPositionSensor

The FieldVariableGlobalPositionSensor measures a particular field variable of a set of elements at the first intersection found

- with a given plane (in case of beam elements), or
- with a given straight line (in case of plate/shell elements), or
- with a given global position (in case of solid elements).

The user specifies a list of related elements (related_elements), with which this sensor may operate. In sequential order it is analysed if an element intersects with the given global hyper-plane, global straight line, or global position. As soon as an intersection is found (in terms of element number and a local position) the fieldvariable value at this position is returned. If no intersection is found, then the sensor returns 0. Two member variables are required in total to define all geometric properties of the three modes (a)-(c): one position vector (global_position), and one direction vector (global_direction).

Data objects of FVGlobalPositionSensor:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signal_Storage_Mode</td>
<td>integer</td>
<td></td>
<td>5</td>
<td>storage mode of the sensor signal (0 -&gt; no data storage, 1 -&gt; general solution file of the simulation, 2 -&gt; separate file with the results of this particular sensor, 4 -&gt; internal array in memory - all time signal points)</td>
</tr>
<tr>
<td>sensor_number</td>
<td>integer</td>
<td>R</td>
<td>1</td>
<td>number of the sensor in the mbs</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;sensor&quot;</td>
<td>name of the sensor for the output files and for the plot tool</td>
</tr>
<tr>
<td>precision</td>
<td>integer</td>
<td>-1</td>
<td>precision of output in solution files. Use -1 for default (SolverOptions.Solution.Sensor.output_precision)</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>----</td>
<td>------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>sensor_type</td>
<td>string</td>
<td>&quot;FVGlobalPositionSensor&quot;</td>
<td>specification of sensor type. Once the sensor is added to the mbs, you MUST NOT change this type anymore!</td>
<td></td>
</tr>
<tr>
<td>related_elements</td>
<td>vector</td>
<td>[1]</td>
<td>list of elements, on which the sensor acts</td>
<td></td>
</tr>
<tr>
<td>global_position</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>global position.</td>
<td></td>
</tr>
<tr>
<td>global_direction</td>
<td>vector</td>
<td>[1, 0, 0]</td>
<td>either pane normal (case a) or direction of a straight line (case b) or meaningless (case c).</td>
<td></td>
</tr>
<tr>
<td>field_variable</td>
<td>string</td>
<td>&quot;position&quot;</td>
<td>name of the field variable, e.g. 'position', see the documentation of the elements for the available field variables</td>
<td></td>
</tr>
<tr>
<td>component</td>
<td>string</td>
<td>&quot;x&quot;</td>
<td>component of the field variable, e.g. 'x'</td>
<td></td>
</tr>
</tbody>
</table>
3.10 SensorProcessors

These sensors are available:

- no SensorProcessor available

SensorProcessors are objects that can perform operations on sensog signals. A Sensor can have any number of processors associated to it, they are applied in the sequence they are added. (Processor 1 is applied to the initial signal, Processor 2 is applied to the result of Processor 1...).

SensorProcessors must be associated with a Sensor. In the script Language use the Command AddSensorProcessor to add to a specific Sensor.
3.11 GeomElement

These GeomElements are available:

- GeomMesh3D, 3.11.1
- GeomCylinder3D, 3.11.2
- GeomSphere3D, 3.11.3
- GeomCube3D, 3.11.4
- GeomOrthoCube3D, 3.11.5

GeomElements are used in HOTINT to improve the appearance of your simulation model. GeomElements do not have any physical meaning in HOTINT and have to be attached to the ground or some (real) reference element. The GeomElement will move with this reference element.

GeomElements are also a good tool to define surfaces e.g. used for coupled simulations with fluid-structure interaction.

In the script language the command AddGeomElement is used to add GeomElements to the system.

3.11.1 GeomMesh3D

Data objects of GeomMesh3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_element_type</td>
<td>string</td>
<td></td>
<td>&quot;GeomMesh3D&quot;</td>
<td>specification of GeomElement type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;GeomElement&quot;</td>
<td>name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>0 ... ground, otherwise insert number of existing element</td>
</tr>
</tbody>
</table>

**Graphics**

- Graphics.RGB_color: vector: [0.2, 0.2, 0.8], range = 0..1
- Graphics.transparency: double: -1, transparency [0..1], 0=transparent, 1=solid, set -1 if global transparency is used
- Graphics.drawstyle: integer: 3, \(+1:\) draw outline, \(+2:\) fill area, \(+4:\) highlight points, \(+8:\) colored: outline
- Graphics.pointsize: double: 0.1, size for highlighted points [m]
- Graphics.linethickness: double: 1, thickness of lines [pts]
- Graphics.smooth_drawing: bool: 1, Draw smooth interpolation of surface
- Graphics.draw_edge_angle: double: 36, Minimum angle between two triangles that defines an edge (°)
- tolerance: double: 0, if two points are closer than tolerance, they are treated as same point
- tolerance: double: 0, tolerance computes as tolerance_relative times the radius of the bounding box of the points

**Geometry**

- Geometry.transform_scale: vector: [1, 1, 1], Resize GeomElement in X, Y and Z direction [sX, sY, sZ]
- Geometry.transform_rotation: vector: [0, 0, 0], Resize GeomElement in X, Y and Z direction [sX, sY, sZ]
### 3.11. GEOMELEMENT

**MeshData**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform_position</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>Translate GeomElement in X, Y and Z direction [tX, tY, tZ]</td>
</tr>
<tr>
<td>MeshData.triangles</td>
<td>matrix</td>
<td></td>
<td></td>
<td>Fill in point numbers of each triangle: p1, p2, p3; p4, p5, p6 ...</td>
</tr>
<tr>
<td>MeshData.points</td>
<td>matrix</td>
<td></td>
<td></td>
<td>Fill in point coordinates: X1, Y1, Z1; X2, Y2, Z2 ...</td>
</tr>
</tbody>
</table>

### 3.11.2 GeomCylinder3D

**Data objects of GeomCylinder3D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_element_type</td>
<td>string</td>
<td></td>
<td>&quot;GeomCylinder3D&quot;</td>
<td>specification of GeomElement type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;GeomElement&quot;</td>
<td>name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td>0</td>
<td>0 ... ground, otherwise insert number of existing element</td>
<td></td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics</th>
<th>vector</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.2, 0.2, 0.8]</td>
<td>[red, green, blue], range = 0..1</td>
</tr>
<tr>
<td>Graphics.transparency</td>
<td>double</td>
<td></td>
<td>-1</td>
<td>transparency [0..1], 0 = transparent, 1 = solid, set -1 if global transparency is used</td>
</tr>
<tr>
<td>Graphics.drawstyle</td>
<td>integer</td>
<td></td>
<td>3</td>
<td>+1: draw outline, +2 fill area, +4 highlight points, +8 colored: outline</td>
</tr>
<tr>
<td>Graphics.pointright</td>
<td>double</td>
<td></td>
<td>0.1</td>
<td>size for highlighted points [m]</td>
</tr>
<tr>
<td>Graphics.linethickness</td>
<td>double</td>
<td></td>
<td>1</td>
<td>thickness of lines [pts]</td>
</tr>
<tr>
<td>Graphics.draw_resolution</td>
<td>integer</td>
<td></td>
<td>16</td>
<td>Number of quadrangles to draw the cylinder surface</td>
</tr>
<tr>
<td>Graphics.split_coloring</td>
<td>bool</td>
<td>0</td>
<td>true if one side should be slightly lighter than the other</td>
<td></td>
</tr>
</tbody>
</table>

**Geometry**

<table>
<thead>
<tr>
<th>Geometry</th>
<th>double</th>
<th></th>
<th>0</th>
<th>radius of the cylinder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.radius_hole</td>
<td>double</td>
<td></td>
<td>0</td>
<td>inner radius of the cylinder (0 if full cylinder)</td>
</tr>
<tr>
<td>Geometry.axis_point1</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>point on axis of rotation</td>
</tr>
<tr>
<td>Geometry.axis_point2</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>point on axis of rotation</td>
</tr>
</tbody>
</table>

### 3.11.3 GeomSphere3D

**Data objects of GeomSphere3D:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_element_type</td>
<td>string</td>
<td></td>
<td>&quot;GeomSphere3D&quot;</td>
<td>specification of GeomElement type. Once the element is added to the mbs, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>&quot;GeomElement&quot;</td>
<td>name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td>0</td>
<td>0 ... ground, otherwise insert number of existing element</td>
<td></td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Graphics</th>
<th>vector</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.2, 0.2, 0.8]</td>
<td>[red, green, blue], range = 0..1</td>
</tr>
</tbody>
</table>
### 3.11.4 GeomCube3D

Data objects of GeomCube3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_element_type</td>
<td>string</td>
<td>&quot;GeomCube3D&quot;</td>
<td></td>
<td>specification of GeomElement type. Once the element is added to the mb, you MUST NOT change this type anymore!</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;GeomElement&quot;</td>
<td></td>
<td>name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td>0</td>
<td>0</td>
<td>0...ground, otherwise insert number of existing element</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.2, 0.2, 0.8]</td>
<td></td>
<td>red, green, blue, range = 0.1</td>
</tr>
<tr>
<td>Graphics.transparency</td>
<td>double</td>
<td>-1</td>
<td>0</td>
<td>transparency [0, 1], 0=transparent, 1=solid, set -1 if global transparency is used</td>
</tr>
<tr>
<td>Graphics.drawstyle</td>
<td>integer</td>
<td>3</td>
<td>3+1</td>
<td>+1: draw outline, +2 fill area, +4 highlight points, +8 colored: outline</td>
</tr>
<tr>
<td>Graphics.poinsize</td>
<td>double</td>
<td>0.1</td>
<td>0.1</td>
<td>size for highlighted points [m]</td>
</tr>
<tr>
<td>Graphics.linetickness</td>
<td>double</td>
<td>1</td>
<td>1</td>
<td>thickness of lines [pts]</td>
</tr>
</tbody>
</table>

**Geometry**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.point1</td>
<td>vector</td>
<td>[-0.5, -0.5, -0.5]</td>
<td></td>
<td>Bottom point 1 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point2</td>
<td>vector</td>
<td>[0.5, -0.5, -0.5]</td>
<td></td>
<td>Bottom point 2 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point3</td>
<td>vector</td>
<td>[-0.5, 0.5, -0.5]</td>
<td></td>
<td>Bottom point 3 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point4</td>
<td>vector</td>
<td>[0.5, 0.5, -0.5]</td>
<td></td>
<td>Bottom point 4 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point5</td>
<td>vector</td>
<td>[-0.5, -0.5, 0.5]</td>
<td></td>
<td>Bottom point 5 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point6</td>
<td>vector</td>
<td>[0.5, -0.5, 0.5]</td>
<td></td>
<td>Bottom point 6 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point7</td>
<td>vector</td>
<td>[-0.5, 0.5, 0.5]</td>
<td></td>
<td>Bottom point 7 of bottom points: 1-2-4-3</td>
</tr>
<tr>
<td>Geometry.point8</td>
<td>vector</td>
<td>[0.5, 0.5, 0.5]</td>
<td></td>
<td>Bottom point 8 of bottom points: 1-2-4-3</td>
</tr>
</tbody>
</table>

### 3.11.5 GeomOrthoCube3D

Data objects of GeomOrthoCube3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_element_type</td>
<td>string</td>
<td>&quot;GeomOrthoCube3D&quot;</td>
<td></td>
<td>specification of GeomElement type.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;GeomElement&quot;</td>
<td></td>
<td>name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td>0</td>
<td>0</td>
<td>0...ground, otherwise insert number of existing element</td>
</tr>
</tbody>
</table>
### 3.11. GEOMELEMENT

<table>
<thead>
<tr>
<th>geom_element_type</th>
<th>string</th>
<th>&quot;GeomOrthoCube3D&quot; specification of GeomElement type. Once the element is added to the mbs, you MUST NOT change this type anymore!</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>&quot;GeomElement&quot; name of the GeomElement</td>
</tr>
<tr>
<td>reference_element_number</td>
<td>integer</td>
<td>0 ... ground, otherwise insert number of existing element</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Graphics.RGB_color</th>
<th>vector</th>
<th>[0.2, 0.2, 0.8]</th>
<th>red, green, blue, range = 0..1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.transparency</td>
<td>double</td>
<td>-1</td>
<td>transparency [0..1], 0=transparent, 1=solid, set -1 if global transparency is used</td>
</tr>
<tr>
<td>Graphics.drawstyle</td>
<td>integer</td>
<td>3</td>
<td>+1: draw outline, +2 fill area, +4 highlight points, +8 colored; outline</td>
</tr>
<tr>
<td>Graphics.pointsize</td>
<td>double</td>
<td>0.1</td>
<td>size for highlighted points [m]</td>
</tr>
<tr>
<td>Graphics.linethickness</td>
<td>double</td>
<td>1</td>
<td>thickness of lines [pts]</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th>Geometry.center_point</th>
<th>vector</th>
<th>[0, 0, 0]</th>
<th>Center point in global coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry.size</td>
<td>vector</td>
<td>[1, 1, 1]</td>
<td>Dimension of cube in X, Y and Z-direction</td>
</tr>
<tr>
<td>Geometry.rotation_matrix</td>
<td>matrix</td>
<td>[1, 0, 0; 0, 1, 0; 0, 0, 1]</td>
<td>The rotation matrix defines the orientation of the cube (global_point = matrix . local_point).</td>
</tr>
</tbody>
</table>
3.12  Set

These Sets are available:

- ElementSet, 3.12.1
- GlobalNodeSet, 3.12.2
- LocalNodeSetA, 3.12.3
- LocalNodeSetB, 3.12.4
- GlobalCoordSet, 3.12.5
- LocalCoordSetA, 3.12.6
- LocalCoordSetB, 3.12.7
- FaceSetA, 3.12.8
- SensorSet, 3.12.9

In HOTINT sets of different types can be defined. They fall in different categories (ElementSets, PointSet). For some categories there is more than one possibility to define a set of a given type. The PointSet can be defined as, for example list of global positions, list of elements and local nodes, ...

Sets can be manipulated via the GUI (add, edit, delete). The script language also provides the command `AddSet`.

The different possibilities to define a set all lead to the same data for further processing. All PointSets are converted to a list of (element number, local position) when used by a command.

Sets can be used as input parameters for functions to generate constraints, manipulate properties.

3.12.1  ElementSet

defines set of elements

Data objects of ElementSet:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;ElementSet&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;ElementSet&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td></td>
<td>[]</td>
<td>Elements in this set</td>
</tr>
</tbody>
</table>

3.12.2  GlobalNodeSet

defines set of global nodes

Data objects of GlobalNodeSet:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;GlobalNodeSet&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;GlobalNodeSet&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>Elements in this set</td>
</tr>
</tbody>
</table>
3.12. SET

| Local positions | matrix | R | Local positions on the elements in this set |
| Global node numbers | vector |   | Global Nodes in this set |
| Local node numbers | vector | R | Local node numbers on the elements in this set |

### 3.12.3 LocalNodeSetA

defines pairs of (element, local nodes)

**Data objects of LocalNodeSetA:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;LocalNodeSetA&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;LocalNodeSetA&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>local_positions</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Local positions on the elements in this set</td>
</tr>
<tr>
<td>local_node_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>Local Nodes in this set</td>
</tr>
</tbody>
</table>

### 3.12.4 LocalNodeSetB

defines set of elements and set of local nodes valid for each of these elements - all combinations

**Data objects of LocalNodeSetB:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;LocalNodeSetB&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;LocalNodeSetB&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>local_positions</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Local positions on the elements in this set</td>
</tr>
<tr>
<td>element_numbers_shortlist</td>
<td>vector</td>
<td></td>
<td></td>
<td>Shortlist of Elements in this set</td>
</tr>
<tr>
<td>local_node_numbers_shortlist</td>
<td>vector</td>
<td></td>
<td></td>
<td>Shortlist of Local Nodes in this set</td>
</tr>
</tbody>
</table>

### 3.12.5 GlobalCoordSet

defines set of global positions

**Data objects of GlobalCoordSet:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;GlobalCoordSet&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;GlobalCoordSet&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>local_positions</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Local positions on the elements in this set</td>
</tr>
<tr>
<td>global_positions</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Global positions registered in this set</td>
</tr>
</tbody>
</table>

### 3.12.6 LocalCoordSetA

defines pairs of (element, local positions)
Data objects of LocalCoordSetA:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;LocalCoordSetA&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;LocalCoordSetA&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>local_positions</td>
<td>matrix</td>
<td></td>
<td></td>
<td>Local positions on the elements in this set</td>
</tr>
</tbody>
</table>

3.12.7 LocalCoordSetB

defines set of elements and set of local positions valid for each of these elements - all combinations

Data objects of LocalCoordSetB:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;LocalCoordSetB&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;LocalCoordSetB&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>local_positions</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Local positions on the elements in this set</td>
</tr>
<tr>
<td>element_numbers_shortlist</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>Shortlist of Elements in this set</td>
</tr>
<tr>
<td>local_positions_shortlist</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>Shortlist of Local Positions in this set</td>
</tr>
</tbody>
</table>

3.12.8 FaceSetA

defines set of element faces

Data objects of FaceSetA:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td></td>
<td>&quot;FaceSetA&quot;</td>
<td>the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td></td>
<td>&quot;FaceSetA&quot;</td>
<td>type of the set</td>
</tr>
<tr>
<td>element_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>Elements in this set</td>
</tr>
<tr>
<td>face_numbers</td>
<td>vector</td>
<td></td>
<td></td>
<td>ElementFaces in this set</td>
</tr>
<tr>
<td>number_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>number of nodes for each face in this set</td>
</tr>
<tr>
<td>node_lists</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>node list for each face in this set</td>
</tr>
<tr>
<td>node_lists_local</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>local node number in element</td>
</tr>
<tr>
<td>face_areas</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>ElementFaceAreas in this set</td>
</tr>
<tr>
<td>used_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>all global nodes used in this set</td>
</tr>
</tbody>
</table>

3.12.9 SensorSet

defines set of sensors

Data objects of SensorSet:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>
3.12. **SET**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_name</td>
<td>string</td>
<td>&quot;SensorSet&quot; the name of the set</td>
</tr>
<tr>
<td>set_type</td>
<td>string</td>
<td>&quot;SensorSet&quot; type of the set</td>
</tr>
<tr>
<td>sensor_numbers</td>
<td>vector</td>
<td>[ ] Sensors in this set</td>
</tr>
</tbody>
</table>
3.13 Mesh

These Meshes are available:

- StructuralMesh, 3.13.1
- SolidMesh, 3.13.2

To handle Meshes in HOTINT a tree-like structure is introduced. This allows to have simple operations to

- generate meshes from geometric primitives
- load meshes from external sources in different formats
- perform operations on parts of the mesh (or the entire mesh)
- automatically connect two or more parts of the mesh
- pick regions from the mesh after processing

The root structure 'Mesh' can be a solid or a structural Mesh. When used in script language, this applies some restrictions on the available operations. The type of a mesh must be known when it is initially created.

Each Mesh consists of MeshComponents. The Components form a tree-like structure. Nodes and Elements of the mesh reside in the components. From the script language the top level components can be accessed.

The general idea is to store nodes and elements in the components as they are generated such that no node is redundant. Access functions thus are often relayed to subordinate components.
3.13. MESH

(child components of the tree).

All Mesh objects must be added to the MultiBodySystem with the AddMestToMBS command in both .cpp models and also .hid models. The Nodes and Elements are added to the MBS individually.

When a MBS is saved the file will contain Nodes and Elements rather than the Mesh. Manipulation via the HOTINT GUI is not implemented (yet) the 'Edit mesh' menu option merely provides information. Changes must be implemented in the .cpp model or the .hid file.

The procedure to generate a complex mesh structure is - for .hid and .cpp models alike - to

1. generate a new mesh object
2. add a source component to the mesh - leaf node always stays leaf node
3. insert other components between mesh and leaf

![Diagram](image)

Figure 3.76: how the assembly 3.75 is generated

More on the available components in 3.14

3.13.1 StructuralMesh

structural mesh

Data objects of StructuralMesh:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh_name</td>
<td>string</td>
<td></td>
<td>&quot;StructuralMesh&quot;</td>
<td>the name of the mesh</td>
</tr>
<tr>
<td>mesh_type</td>
<td>string</td>
<td></td>
<td>&quot;StructuralMesh&quot;</td>
<td>type of the mesh</td>
</tr>
<tr>
<td>number_of_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes in the mesh</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the entire mesh</td>
</tr>
<tr>
<td>number_of_elements</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of elements in the mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the entire mesh</td>
</tr>
<tr>
<td>compute_surface</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>flag to automatically identify domain surfaces</td>
</tr>
</tbody>
</table>
Available Script Functions:
LoadMesh, WriteMesh, Modify, Transform, Distort, Linear2Quadratic, SplitHexes, Refine, Rotate, Mirror, Extrude, AddMeshToMBS, GlueMesh, GetNodesInBox, GetElementsInBox, GetNodesInCylinder, GetNodesInSphere, GetNodesInFunction, GetNodePos, GetElementAtPosition, GetFacesFromNodes, GenerateBeam, GeneratePlate.

### 3.13.2 SolidMesh

solid mesh

**Data objects of SolidMesh:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mesh_name</td>
<td>string</td>
<td></td>
<td>&quot;SolidMesh&quot;</td>
<td>the name of the mesh</td>
</tr>
<tr>
<td>mesh_type</td>
<td>string</td>
<td></td>
<td>&quot;SolidMesh&quot;</td>
<td>type of the mesh</td>
</tr>
<tr>
<td>number_of_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes in the mesh</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the entire mesh</td>
</tr>
<tr>
<td>number_of_elements</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of elements in the mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the entire mesh</td>
</tr>
<tr>
<td>compute_surface</td>
<td>bool</td>
<td></td>
<td>0</td>
<td>flag to automatically identify domain surfaces</td>
</tr>
</tbody>
</table>

Available Script Functions:
LoadMesh, WriteMesh, Modify, Transform, Distort, Linear2Quadratic, SplitHexes, Refine, Rotate, Mirror, Extrude, AddMeshToMBS, GlueMesh, GetNodesInBox, GetElementsInBox, GetNodesInCylinder, GetNodesInSphere, GetNodesInFunction, GetNodePos, GetElementAtPosition, GetFacesFromNodes, GenerateBlock, GenerateCylinder.
3.14 MeshComponents

These MeshComponents are available:

- Primitive: Block, 3.14.1
- Primitive: Cylinder, 3.14.2
- Primitive: Quadrilateral, 3.14.3
- Primitive: Curve, 3.14.4
- Extended: Mirror, 3.14.5
- Extended: Extrude, 3.14.6
- Extended: Rotational, 3.14.7
- Extended: Lin2Quad, 3.14.8
- Extended: SplitHexes, 3.14.9
- Extended: Refine, 3.14.10
- Process: Transform, 3.14.11
- Loader: NetGen2D, 3.14.15
- Loader: NetGen3D, 3.14.16
- Loader: Neutral3D, 3.14.17
- Loader: STL, 3.14.18
- Loader: DataArrays, 3.14.19

NOTE: currently the MeshComponents can not be added or truly manipulated via the GUI. They can be used with special Script-Commands e.g. Mesh.Rotate(component,parameters) that insert the components at the root position automatically and of course also in cpp models.

The MeshComponents that are currently implemented fall in the following categories:

1 Sources - components that can have their own nodes and elements.
   1.1 Primitive - defined geometry meshed regularly with simplest element. Generates own nodes and elements.
   1.2 DomainContainer - Container for the nodes and elements that are loaded from a file.
   1.3 SourceExtended - require a subordinate mesh component. Generates additional or completely replaces nodes and elements.
2 Processors - components that manipulate other components, strictly speaking no own nodes and elements.

2.1 Single Subordinate Processors - require a subordinate mesh component. Manipulate Node positions.

2.2 Multiple Subordinate Processors - require at least one subordinate component. Automatic joining, import routines.

All Mesh component classes are derived from a common base class following the 'decorator' design pattern of object oriented programming.

Figure 3.77: Class Diagram with most important class members and overridden methods

### 3.14.1 Primitive: Block

generates a hex-meshed block

**Data objects of Primitive: Block:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Block&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Block&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS node numbers of the component - entries '1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS element numbers of the component - entries '1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>
### 3.14. MESHCOMPONENTS

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricalNonlinearityStatus</td>
<td>integer</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td>Generation.used_node_type</td>
<td>string</td>
<td>&quot;Node3D&quot; node type used to instantiate the component</td>
</tr>
<tr>
<td>Generation.used_elem_type</td>
<td>string</td>
<td>&quot;Hexahedral&quot; element type used to instantiate the component</td>
</tr>
<tr>
<td>Generation.P1</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P2</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P3</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P4</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P5</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P6</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P7</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P8</td>
<td>vector</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.discretization</td>
<td>vector</td>
<td>discretization for block (x,y,z)</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.7, 0.7, 0.7] (\text{red, green, blue}) color of element, range = 0..1, use default color: [-1,-1,-1]</td>
</tr>
</tbody>
</table>

### 3.14.2 Primitive: Cylinder

generates a hex-meshed 90° segment of a cylinder or pipe

**Data objects of Primitive: Cylinder:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>Type</th>
<th>R</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>R</td>
<td>&quot;Cylinder&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>R</td>
<td>&quot;Cylinder&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the component - entries <code>-1</code> or <code>0</code> hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the component - entries <code>-1</code> or <code>0</code> hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>R</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricalNonlinearityStatus</td>
<td>integer</td>
<td>-1</td>
<td></td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
</tbody>
</table>
### 3.14.3 Primitive: Quadrilateral

generates a quad-meshed quadrilateral

**Data objects of Primitive: Quadrilateral:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Quadrilateral&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Quadrilateral&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

| Generation.body_index         | integer |   | 1                      | the body index / domain number for the entire mesh component     |
| Generation.material_number    | integer |   | 1                      | the material number for the entire mesh component                |
| Generation.GeometricNonlinearity_Status | integer |   | -1                     | -1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |
| Generation.used_node_type     | string  |   | "Node3D"               | node type used to instantiate the component                      |
| Generation.used_elem_type     | string  |   | "ANCFThinPlate3D"      | element type used to instantiate the component                   |
| Generation.P1                 | vector  |   | [0, 0, 0]              | defining points of the component in basic configuration          |
| Generation.P2                 | vector  |   | [1, 0, 0]              | defining points of the component in basic configuration          |
| Generation.P3                 | vector  |   | [0, 1, 0]              | defining points of the component in basic configuration          |
| Generation.P4                 | vector  |   | [1, 0, 1]              | defining points of the component in basic configuration          |
| Generation.discretization     | vector  |   | [1, 1]                 | discretization for quadrilateral (x,y,z)                         |
| Generation.thickness          | double  |   | 0.001                  | thickness for all elements                                       |

**Graphics**

| Graphics.RGB_color            | vector  |   | [0.7, 0.7, 0.7]        | [red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1] |
3.14.4 **Primitive: Curve**

generates discretized beam

**Data objects of Primitive: Curve:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Linear&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Linear&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td></td>
<td></td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td></td>
<td></td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td>Generation.used_node_type</td>
<td>string</td>
<td></td>
<td>&quot;Node3DRxyz&quot;</td>
<td>node type used to instantiate the component</td>
</tr>
<tr>
<td>Generation.used_elem_type</td>
<td>string</td>
<td></td>
<td>&quot;LinearBeam3D&quot;</td>
<td>element type used to instantiate the component</td>
</tr>
<tr>
<td>Generation.P1</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.P2</td>
<td>vector</td>
<td></td>
<td>[1, 0, 0]</td>
<td>defining points of the component in basic configuration</td>
</tr>
<tr>
<td>Generation.discretization</td>
<td>vector</td>
<td></td>
<td>[1]</td>
<td>discretization for linear (x,<em>,</em>)</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.7, 0.7, 0.7]</td>
<td>[red, green, blue] color of element, range = 0.1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

3.14.5 **Extended: Mirror**

Mirror subordinate mesh component

**Data objects of Extended: Mirror:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Mirror&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Mirror&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.same_as_seed_node</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>Generation.same_as_seed_element</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
</tbody>
</table>
### Generation, Geometric Nonlinearity Status

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation_geometric_nonlinearity_status</td>
<td>integer</td>
<td></td>
<td>-1 -1.use_element_defaults, 0.GNS_Linear, 1.GNS_NonlinearSmallStrain, 2.GNS_NonlinearLargeStrain</td>
<td></td>
</tr>
<tr>
<td>Generation_used_node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>Generation_used_elem_type</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
</tbody>
</table>

### Graphics

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.7, 0.7, 0.7] [red, green, blue] color of element, range = 0.1, use default color:[-1,-1,-1]</td>
<td></td>
</tr>
<tr>
<td>plane</td>
<td>integer</td>
<td>1</td>
<td>1 for YZ plane, 2 for XZ plane, 3 for XY plane</td>
</tr>
<tr>
<td>distance</td>
<td>double</td>
<td>0</td>
<td>mirror planes’ distance from origin</td>
</tr>
</tbody>
</table>

### 3.14.6 Extended: Extrude

extrudes subordinate mesh component along axis 1D to 2D, 2D to 3D

**Data objects of Extended: Extrude:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Extrude&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Extrude&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[] MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
<td></td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[] MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
<td></td>
</tr>
</tbody>
</table>

### Generation

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>same_as_seed_node</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>same_as_seed_element</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>Generation_geometric_nonlinearity_status</td>
<td>integer</td>
<td></td>
<td>-1 -1.use_element_defaults, 0.GNS_Linear, 1.GNS_NonlinearSmallStrain, 2.GNS_NonlinearLargeStrain</td>
<td></td>
</tr>
<tr>
<td>used_node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>used_elem_type</td>
<td>string</td>
<td>R</td>
<td>&quot;dependent&quot;</td>
<td></td>
</tr>
<tr>
<td>axis_number</td>
<td>integer</td>
<td></td>
<td>3</td>
<td>direction for extrusion along main axis</td>
</tr>
<tr>
<td>discretization</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>discretization along extrusion axis</td>
</tr>
<tr>
<td>total_extrusion</td>
<td>double</td>
<td></td>
<td>1</td>
<td>distance along extrusion axis</td>
</tr>
<tr>
<td>thickness</td>
<td>double</td>
<td></td>
<td>0.1</td>
<td>plate thickness after extrusion</td>
</tr>
</tbody>
</table>

### Graphics

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0.7, 0.7, 0.7] [red, green, blue] color of element, range = 0.1, use default color:[-1,-1,-1]</td>
<td></td>
</tr>
</tbody>
</table>

### 3.14.7 Extended: Rotational

rotates subordinate 2D mesh component around axis

**Data objects of Extended: Rotational:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
</table>
### 3.14. MESHCOMPONENTS

| component_name | string | "Rotate2D" | the name of the mesh component |
| component_type | string | "Rotate2D" | type of the mesh component |
| list_of_nodes   | vector R | [] | MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh |
| list_of_elements| vector R | [] | MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh |

#### Generation

| Generation.same_as_seed_node | string R | "dependent" |
| Generation.same_as_seed_element | string R | "dependent" |
| Generation.GeometricNonlinearityStatus | integer | -1 | -1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |
| Generation.used_node_type | string R | "dependent" |
| Generation.used_elem_type | string R | "dependent" |
| Generation.modes | integer R | 0 | number of nodes read from external source file |
| Generation.nelems | integer R | 0 | number of elements read from external source file |

#### Graphics

| Graphics.RGB_color | vector | [0.7, 0.7, 0.7] | [red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1] |

### 3.14.8 Extended: Lin2Quad

converts all elements of subordinate mesh component to quadratic, adds required nodes

#### Data objects of Extended: Lin2Quad:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;LinearToQuadratic&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;LinearToQuadratic&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector R</td>
<td>[]</td>
<td></td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector R</td>
<td>[]</td>
<td></td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

#### Generation

| Generation.same_as_seed_node | string R | "dependent" |
| Generation.same_as_seed_element | string R | "dependent" |
| Generation.GeometricNonlinearityStatus | integer | -1 | -1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain |
| Generation.used_node_type | string R | "dependent" |
| Generation.used_elem_type | string R | "dependent" |

#### Graphics

| Graphics.RGB_color | vector | [0.7, 0.7, 0.7] | [red, green, blue] color of element, range = 0..1, use default color: [-1,-1,-1] |
### 3.14.9 Extended: SplitHexes

converts all hexahedrals of subordinate mesh component to tets, prisms or pyramids

#### Data objects of Extended: SplitHexes:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>R</td>
<td>&quot;SplitHexes&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>R</td>
<td>&quot;SplitHexes&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[ ]</td>
<td>MBS node numbers of the component - entries ':' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[ ]</td>
<td>MBS element numbers of the component - entries ':' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

#### Generation

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.same_as_seed_node</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;dependent&quot;</td>
</tr>
<tr>
<td>Generation.same_as_seed_element</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;dependent&quot;</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td>Generation.used_node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;&quot;</td>
<td>node type used to instantiate the component</td>
</tr>
<tr>
<td>Generation.used_elem_type</td>
<td>string</td>
<td>R</td>
<td>&quot;&quot;</td>
<td>element type used to instantiate the component</td>
</tr>
</tbody>
</table>

#### Graphics

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>R</td>
<td>[0.7, 0.7, 0.7]</td>
<td>red,green,blue color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>alternate</td>
<td>bool</td>
<td>R</td>
<td>1</td>
<td>use alternate orientation for split mesh</td>
</tr>
</tbody>
</table>

### 3.14.10 Extended: Refine

refine subordinate hexahedral or quad mesh component

#### Data objects of Extended: Refine:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>R</td>
<td>&quot;Refine&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>R</td>
<td>&quot;Refine&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[ ]</td>
<td>MBS node numbers of the component - entries ':' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[ ]</td>
<td>MBS element numbers of the component - entries ':' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

#### Generation

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.same_as_seed_node</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;dependent&quot;</td>
</tr>
<tr>
<td>Generation.same_as_seed_element</td>
<td>string</td>
<td>R</td>
<td></td>
<td>&quot;dependent&quot;</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td>Generation.used_node_type</td>
<td>string</td>
<td>R</td>
<td>&quot;&quot;</td>
<td>node type used to instantiate the component</td>
</tr>
</tbody>
</table>
3.14. MESHCOMPONENTS

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>used_elem_type</td>
<td>string</td>
<td>element type used to instantiate the component</td>
</tr>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td>[0, 0, 0] [red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

### 3.14.11 Process: Transform

translate, rotate, scale subordinate mesh component with constant transformation matrix

**Data objects of Process: Transform:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>&quot;Transformation&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>&quot;Transformation&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>MBS node numbers of the component - entries '1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>MBS element numbers of the component - entries '1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body_index</td>
<td>integer</td>
<td>0</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>material_number</td>
<td>integer</td>
<td>0</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>GeometricNonlinearityStatus</td>
<td>integer</td>
<td>-1</td>
<td>-1. use element defaults, 0.GNS_Linear, 1.GNS_NonlinearSmallStrain, 2.GNS_NonlinearLargeStrain</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB_color</td>
<td>vector</td>
<td>[0, 0, 0]</td>
<td>[red, green, blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
<tr>
<td>transformation_matrix</td>
<td>matrix</td>
<td>[1, 0, 0, 0; 0, 1, 0, 0; 0, 0, 1; 0, 0, 1]</td>
<td>4 dimensional transformation matrix - translation + scale + rotation</td>
</tr>
</tbody>
</table>

### 3.14.12 Process: Distort

assign new node positions as any function of noderepositions as in subordinate mesh component

**Data objects of Process: Distort:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>&quot;Distortion&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>&quot;Distortion&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>MBS node numbers of the component - entries '1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>MBS element numbers of the component - entries '1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body_index</td>
<td>integer</td>
<td>0</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Data name</td>
<td>type</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------</td>
<td>---</td>
<td>--------------------</td>
</tr>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Modifier&quot;</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Modifier&quot;</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0, 0, 0]</td>
<td>[red,green,blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

### 3.14.13 Process: Modify

modify a single property for all subordinate components

**Data objects of Process: Modify:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;Modifier&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;Modifier&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the component - entries -1 or 0 hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the component - entries -1 or 0 hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Generation**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
</tbody>
</table>

**Graphics**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.7, 0.7, 0.7]</td>
<td>[red,green,blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

### 3.14.14 Process: WriterNeutral3D

writes Mesh in neutral 3d format to file
3.14. MESHCOMPONENTS

Data objects of Process: WriterNeutral3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;MeshWriterNeutral3D&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;MeshWriterNeutral3D&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

Generation

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation.body_index</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.material_number</td>
<td>integer</td>
<td></td>
<td>1</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td>Generation.GeometricNonlinearityStatus</td>
<td>integer</td>
<td></td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td>Generation.target_file</td>
<td>string</td>
<td></td>
<td>&quot;&quot;</td>
<td>target file for the mesh</td>
</tr>
</tbody>
</table>

Graphics

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics.RGB_color</td>
<td>vector</td>
<td></td>
<td>[0.7, 0.7, 0.7]</td>
<td>[red,green,blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

3.14.15 Loader: NetGen2D

reads 2D triangle Mesh from NetGen file *.nf

Data objects of Loader: NetGen2D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;LoaderNetgen2D&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;LoaderNetgen2D&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

Source

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source.source_file</td>
<td>string</td>
<td></td>
<td>&quot;&quot;</td>
<td>external source file for the mesh</td>
</tr>
<tr>
<td>Source.source_nodes</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of nodes read from external source file</td>
</tr>
<tr>
<td>Source.source_elems</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of elements read from external source file</td>
</tr>
<tr>
<td>Source.generated_nodes</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of nodes after domain splitting</td>
</tr>
</tbody>
</table>

BoundaryConditions

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundaryConditions.number_of_boundary_conditions</td>
<td>integer</td>
<td></td>
<td>0</td>
<td>number of boundary conditions</td>
</tr>
<tr>
<td>BoundaryConditions.bcmapping</td>
<td>matrix</td>
<td>R</td>
<td></td>
<td>mapping of loaded bc to internal set numbers (bcNr, dom1, dom2, setNr1, setNr2)</td>
</tr>
</tbody>
</table>

PeriodicBoundaryConditions

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeriodicBoundaryConditions.nodes_1</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>identical nodes side 1 (periodic boundary conditions)</td>
</tr>
<tr>
<td>PeriodicBoundaryConditions.nodes_2</td>
<td>vector</td>
<td>R</td>
<td></td>
<td>identical nodes side 2 (periodic boundary conditions)</td>
</tr>
</tbody>
</table>
3.14.16 Loader: NetGen3D

reads 3D (purely tetrahedral!) Mesh from NetGen file *.nf

Data objects of Loader: NetGen3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;LoaderNetgen3D&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;LoaderNetgen3D&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

Source

<table>
<thead>
<tr>
<th>Source source_file</th>
<th>string</th>
<th></th>
<th>&quot;&quot;</th>
<th>external source file for the mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source source_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes read from external source file</td>
</tr>
<tr>
<td>Source source elems</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of elements read from external source file</td>
</tr>
<tr>
<td>Source generated_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes after domain splitting</td>
</tr>
</tbody>
</table>

BoundaryConditions

| BoundaryConditions number_of_boundary_conditions | integer | R | 0 | number of boundary conditions                                              |
| BoundaryConditions bcmapping         | matrix   | R | [] | mapping of loaded bc to internal set numbers (bcNr, dom1, dom2, setNr1, setNr2) |

PeriodicBoundaryConditions

| PeriodicBoundaryConditions nodes_1 vector nodes_2 vector | [] | [] | identical nodes side 1 (periodic boundary conditions)                      |

3.14.17 Loader: Neutral3D

reads Mesh from file in neutral 3d format

Data objects of Loader: Neutral3D:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>R</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td></td>
<td>&quot;MeshLoaderNeutral3D&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td></td>
<td>&quot;MeshLoaderNeutral3D&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>[]</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

Source

<table>
<thead>
<tr>
<th>Source source_file</th>
<th>string</th>
<th></th>
<th>&quot;&quot;</th>
<th>external source file for the mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source source_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes read from external source file</td>
</tr>
<tr>
<td>Source source elems</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of elements read from external source file</td>
</tr>
<tr>
<td>Source generated_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
<td>number of nodes after domain splitting</td>
</tr>
</tbody>
</table>
### 3.14. MESHCOMPONENTS

<table>
<thead>
<tr>
<th>BoundaryConditions</th>
<th>number of boundary conditions</th>
<th>0</th>
<th>number of boundary conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundaryConditions</td>
<td>bcmapping</td>
<td>matrix</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PeriodicBoundaryConditions</th>
<th>nodes_1</th>
<th>vector</th>
<th>R</th>
<th>identical nodes side 1 (periodic boundary conditions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeriodicBoundaryConditions</td>
<td>nodes_2</td>
<td>vector</td>
<td>R</td>
<td>identical nodes side 2 (periodic boundary conditions)</td>
</tr>
</tbody>
</table>

### 3.14.18 Loader: STL

reads triangle Mesh from STL file

**Data objects of Loader: STL:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>&quot;LoaderSTL&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>&quot;LoaderSTL&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>list_of_nodes</td>
<td>vector</td>
<td>R</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td>list_of_elements</td>
<td>vector</td>
<td>R</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

**Source**

<table>
<thead>
<tr>
<th>Source</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_file</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>external source file for the mesh</td>
</tr>
<tr>
<td>source_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>source_elements</td>
<td>integer</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>generated_nodes</td>
<td>integer</td>
<td>R</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BoundaryConditions</th>
<th>number of boundary conditions</th>
<th>0</th>
<th>number of boundary conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundaryConditions</td>
<td>bcmapping</td>
<td>matrix</td>
<td>R</td>
</tr>
</tbody>
</table>

**PeriodicBoundaryConditions**

<table>
<thead>
<tr>
<th>PeriodicBoundaryConditions</th>
<th>nodes_1</th>
<th>vector</th>
<th>R</th>
<th>identical nodes side 1 (periodic boundary conditions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeriodicBoundaryConditions</td>
<td>nodes_2</td>
<td>vector</td>
<td>R</td>
<td>identical nodes side 2 (periodic boundary conditions)</td>
</tr>
</tbody>
</table>

### 3.14.19 Loader: DataArrays

reads Mesh from double* arrays Hexes assumed

**Data objects of Loader: DataArrays:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>string</td>
<td>&quot;LoaderDataArrays&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td>component_type</td>
<td>string</td>
<td>&quot;LoaderDataArrays&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>list_of_nodes</code></td>
<td>vector</td>
<td>R</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td><code>list_of_elements</code></td>
<td>vector</td>
<td>R</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

### Domain1

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>component_name</code></td>
<td>string</td>
<td>&quot;Domain1&quot;</td>
<td>the name of the mesh component</td>
</tr>
<tr>
<td><code>component_type</code></td>
<td>string</td>
<td>&quot;DomainContainer&quot;</td>
<td>type of the mesh component</td>
</tr>
<tr>
<td><code>list_of_nodes</code></td>
<td>vector</td>
<td>R</td>
<td>MBS node numbers of the component - entries '-1' or '0' hint to nodes not actually used in mesh</td>
</tr>
<tr>
<td><code>list_of_elements</code></td>
<td>vector</td>
<td>R</td>
<td>MBS element numbers of the component - entries '-1' or '0' hint to elements not actually used in mesh</td>
</tr>
</tbody>
</table>

### Domain1.Generation

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>body_index</code></td>
<td>integer</td>
<td>1</td>
<td>the body index / domain number for the entire mesh component</td>
</tr>
<tr>
<td><code>material_number</code></td>
<td>integer</td>
<td>1</td>
<td>the material number for the entire mesh component</td>
</tr>
<tr>
<td><code>GeometricNonlinearityStatus</code></td>
<td>integer</td>
<td>-1</td>
<td>-1..use element defaults, 0..GNS_Linear, 1..GNS_NonlinearSmallStrain, 2..GNS_NonlinearLargeStrain</td>
</tr>
<tr>
<td><code>used_node_type</code></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>node type used to instantiate the component</td>
</tr>
<tr>
<td><code>used_elem_type</code></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>element type used to instantiate the component</td>
</tr>
</tbody>
</table>

### Domain1.Graphics

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RGB_color</code></td>
<td>vector</td>
<td>[0.7, 0.7, 0.7]</td>
<td>[red,green,blue] color of element, range = 0..1, use default color:[-1,-1,-1]</td>
</tr>
</tbody>
</table>

### Source

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>source_file</code></td>
<td>string</td>
<td>&quot;&quot;</td>
<td>external source file for the mesh</td>
</tr>
<tr>
<td><code>source_nodes</code></td>
<td>integer</td>
<td>R</td>
<td>number of nodes read from external source file</td>
</tr>
<tr>
<td><code>source_elems</code></td>
<td>integer</td>
<td>R</td>
<td>number of elements read from external source file</td>
</tr>
<tr>
<td><code>generated_nodes</code></td>
<td>integer</td>
<td>R</td>
<td>number of nodes after domain splitting</td>
</tr>
</tbody>
</table>

### BoundaryConditions

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number_of_boundary_conditions</code></td>
<td>integer</td>
<td>0</td>
<td>number of boundary conditions</td>
</tr>
<tr>
<td><code>bcmapping</code></td>
<td>matrix</td>
<td>R</td>
<td>mapping of loaded bc to internal set numbers, (bcNr, dom1, dom2, setNr1, setNr2)</td>
</tr>
</tbody>
</table>

### PeriodicBoundaryConditions

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nodes_1</code></td>
<td>vector</td>
<td>R</td>
<td>identical nodes side 1 (periodic boundary conditions)</td>
</tr>
<tr>
<td><code>nodes_2</code></td>
<td>vector</td>
<td>R</td>
<td>identical nodes side 2 (periodic boundary conditions)</td>
</tr>
</tbody>
</table>
3.14. MeshComponents

3.14.20 Refinement

The MeshComponent MeshRefine implements a 1 to 3 refinement for Quadrilateral and Hexahedral elements. The implementation is based on [19].

The main advantage of the 1 to 3 refinement is that it can be applied locally without caring about neighboring elements. Also the algorithm for multiple refinement steps is simpler.

**Pseudocode**

- find maximum of all nodes’ subdivision/refinement level. Done if its zero.
- loop over all elements
- identify template and orientation for element from node subdivision/refinement levels (unique bitvalue 0..255)
- generate all additional nodes
  - test for existence with all nodes already generated in this iteration of the algorithm
  - assign subdivision/refinement level to new node - general rule: min(neighbors)
- generate all additional elements, update first (parent) element
- decrease all subdivision/refinement level by 1

**Quadrilateral** The refinement of the Quadrilateral requires 6 templates for 16 possible patterns of marked nodes, the Quadrilateral can be oriented in 4 ways.

![Figure 3.78: all possible templates for the Quadrilateral 3-Refinement][19](Fig 53)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>selected</th>
<th>subtype</th>
<th>Permutations</th>
<th>Fig</th>
<th>imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>y</td>
</tr>
<tr>
<td>corner</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>y</td>
</tr>
<tr>
<td>edge</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>2a</td>
<td>y</td>
</tr>
<tr>
<td>diagonal</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2b</td>
<td>y</td>
</tr>
<tr>
<td>3 corners</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>y</td>
</tr>
<tr>
<td>all</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>y</td>
</tr>
</tbody>
</table>
Hexahedral The refinement of the Hexahedral requires 23 templates for 256 possible patterns of marked nodes, the Hexahedral can be oriented in 24 ways.

<table>
<thead>
<tr>
<th>Template Name</th>
<th>selected</th>
<th>subtype</th>
<th>Permutations</th>
<th>Fig</th>
<th>imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>y</td>
</tr>
<tr>
<td>corner</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>y</td>
</tr>
<tr>
<td>edge</td>
<td>2</td>
<td>0</td>
<td>12</td>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>diagonal 2d</td>
<td>2</td>
<td>1</td>
<td>12</td>
<td>3</td>
<td>n</td>
</tr>
<tr>
<td>diagonal 3d</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>n</td>
</tr>
<tr>
<td>edge edge</td>
<td>3</td>
<td>0</td>
<td>24</td>
<td>5</td>
<td>n</td>
</tr>
<tr>
<td>edge diag2</td>
<td>3</td>
<td>1</td>
<td>24</td>
<td>6</td>
<td>n</td>
</tr>
<tr>
<td>diag2 diag2</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>n</td>
</tr>
<tr>
<td>face</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>y</td>
</tr>
<tr>
<td>tripod</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>n</td>
</tr>
<tr>
<td>left Z</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>right Z</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>10</td>
<td>n</td>
</tr>
<tr>
<td>anchor</td>
<td>4</td>
<td>4</td>
<td>24</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>par. edges</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>twisted</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>inv(edge edge)</td>
<td>5</td>
<td>0</td>
<td>24</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>inv(diag 2d)</td>
<td>5</td>
<td>1</td>
<td>24</td>
<td>-</td>
<td>n</td>
</tr>
</tbody>
</table>
### 3.14. MESHCOMPONENTS

<table>
<thead>
<tr>
<th>inv(diag 3d)</th>
<th>5</th>
<th>2</th>
<th>8</th>
<th>-</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>inv(edge)</td>
<td>6</td>
<td>0</td>
<td>12</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>inv(diag 2d)</td>
<td>6</td>
<td>1</td>
<td>12</td>
<td>11</td>
<td>n</td>
</tr>
<tr>
<td>inv(diag 3d)</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>inv(corner)</td>
<td>7</td>
<td>0</td>
<td>8</td>
<td>12</td>
<td>n</td>
</tr>
<tr>
<td>all</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>y</td>
</tr>
</tbody>
</table>

The table below lists all 24 possible orientations of the hexahedral (6 faces times 4 orientations for each edge). The column 'Node Permutation' lists the node number sequence in this orientation, the inverse permutation was used to compile a mapping from bitvalues back to template and orientation. The column 'coordinates' holds the mapping that is applied to rotate any additional node from the orientation 0 position to the correct one for node generation.

<table>
<thead>
<tr>
<th>Orientation</th>
<th>Node Permutation</th>
<th>inverse Permutation</th>
<th>coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1,2,3,4, 5,6,7,8</td>
<td>1,2,3,4, 5,6,7,8</td>
<td>(x₀,y₀,z₀)</td>
</tr>
<tr>
<td>1</td>
<td>3,1,4,2, 7,5,8,6</td>
<td>2,4,1,3, 6,8,5,7</td>
<td>(y₀,−x₀,z₀)</td>
</tr>
<tr>
<td>2</td>
<td>4,3,2,1, 8,7,6,5</td>
<td>4,3,2,1, 8,7,6,5</td>
<td>(−x₀,−y₀,z₀)</td>
</tr>
<tr>
<td>3</td>
<td>2,4,1,3, 6,8,5,7</td>
<td>3,1,4,2, 7,5,8,6</td>
<td>(−y₀,−x₀,z₀)</td>
</tr>
<tr>
<td>4</td>
<td>5,6,1,2, 7,8,3,4</td>
<td>3,4,7,8, 1,2,5,6</td>
<td>(x₀,z₀,−y₀)</td>
</tr>
<tr>
<td>5</td>
<td>1,5,2,6, 3,7,4,8</td>
<td>1,3,5,7, 2,4,6,8</td>
<td>(y₀,z₀,x₀)</td>
</tr>
<tr>
<td>6</td>
<td>2,1,6,5, 4,3,8,7</td>
<td>2,1,6,5, 4,3,8,7</td>
<td>(−x₀,−y₀,y₀)</td>
</tr>
<tr>
<td>7</td>
<td>6,2,5,1, 8,4,7,3</td>
<td>4,2,8,6, 3,1,7,5</td>
<td>(−y₀,−x₀,y₀)</td>
</tr>
<tr>
<td>8</td>
<td>7,8,5,6, 3,4,1,2</td>
<td>7,8,5,6, 3,4,1,2</td>
<td>(x₀,−y₀,−z₀)</td>
</tr>
<tr>
<td>9</td>
<td>5,7,6,8, 1,3,2,4</td>
<td>5,7,6,8, 1,3,2,4</td>
<td>(y₀,−x₀,−z₀)</td>
</tr>
<tr>
<td>10</td>
<td>6,5,8,7, 2,1,4,3</td>
<td>6,5,8,7, 2,1,4,3</td>
<td>(−x₀,−y₀,−z₀)</td>
</tr>
<tr>
<td>11</td>
<td>8,6,7,5, 4,2,3,1</td>
<td>8,6,7,5, 4,2,3,1</td>
<td>(−y₀,−x₀,−z₀)</td>
</tr>
<tr>
<td>12</td>
<td>3,4,7,8, 1,2,5,6</td>
<td>1,5,2,6, 3,7,4,8</td>
<td>(x₀,−z₀,y₀)</td>
</tr>
<tr>
<td>13</td>
<td>7,3,8,4, 5,1,6,2</td>
<td>6,8,2,4, 5,7,1,3</td>
<td>(y₀,−z₀,−x₀)</td>
</tr>
<tr>
<td>14</td>
<td>8,7,4,3, 6,5,2,1</td>
<td>8,7,4,3, 6,5,2,1</td>
<td>(−x₀,−z₀,−y₀)</td>
</tr>
<tr>
<td>15</td>
<td>4,8,3,7, 2,6,1,5</td>
<td>7,5,3,1, 8,6,4,2</td>
<td>(−y₀,−z₀,−x₀)</td>
</tr>
<tr>
<td>16</td>
<td>2,6,4,8, 1,5,3,7</td>
<td>5,1,7,3, 6,2,8,4</td>
<td>(−x₀,y₀,z₀)</td>
</tr>
<tr>
<td>17</td>
<td>4,2,8,6, 3,1,7,5</td>
<td>6,2,5,1, 8,4,7,3</td>
<td>(−z₀,−y₀,−x₀)</td>
</tr>
<tr>
<td>18</td>
<td>8,4,6,2, 7,3,5,1</td>
<td>8,4,6,2, 7,3,5,1</td>
<td>(−y₀,−z₀,−x₀)</td>
</tr>
<tr>
<td>19</td>
<td>6,8,2,4, 5,7,1,3</td>
<td>7,3,8,4, 5,1,6,2</td>
<td>(−z₀,−y₀,−x₀)</td>
</tr>
<tr>
<td>20</td>
<td>5,1,7,3, 6,2,8,4</td>
<td>2,6,4,8, 1,5,3,7</td>
<td>(−x₀,−y₀,z₀)</td>
</tr>
<tr>
<td>21</td>
<td>7,5,3,1, 8,6,4,2</td>
<td>4,8,3,7, 2,6,1,5</td>
<td>(z₀,−y₀,−x₀)</td>
</tr>
<tr>
<td>22</td>
<td>3,7,1,5, 4,8,2,6</td>
<td>3,7,1,5, 4,8,2,6</td>
<td>(z₀,−y₀,−x₀)</td>
</tr>
<tr>
<td>23</td>
<td>1,3,5,7, 2,4,6,8</td>
<td>1,5,2,6, 3,7,4,8</td>
<td>(z₀,−y₀,−x₀)</td>
</tr>
</tbody>
</table>

### 3.14.21 MeshElements

These mesh elements are available:

- MeshHex
CHAPTER 3. HOTINT REFERENCE MANUAL

Figure 3.80: HOTINT: 2 stage refinement of Quadrilateral

For the Mesh objects only the elements do only require rudimentary data to be useful for Mesh generation. The goal is to use as little memory space as possible.

The only strictly necessary data for the element is the list of nodes, the other properties can be obtained by functions if derived classes are used. Note that the list of nodes is a mere list of node numbers, not the actual For faster processing the element type and material are also data members.

Using derived classes it is also possible to implement certain functions for specific types of elements.

The main methods available for a mesh element are:

- ComputeGlobPos(locpos) - computes the global coordinate system position for a given local position in element coordinates. List of Node coordinates as additional input.
- ComputeLocPos() - computes the local coordinate system position for a given global position.
- Invert() - Element can reorientate itself consistently after a Mirror operation
- IntermediateNodes - Element has a list of additional nodes for the linear to quadratic conversion
- Refine - Element has list of refinement templates (many several functions).
Figure 3.81: some possible templates for the Hexahedral 3-Refinement.[19] (Fig 57)
Figure 3.82: HOTINT: Hexahedral Refinement "edge" in orientation 0
3.15 Command

These commands are available:

- AddElement, 3.15.1
- AddGeomElement, 3.15.2
- AssignGeomElementToElement, 3.15.3
- AddConnector, 3.15.4
- AddLoad, 3.15.5
- AddSensor, 3.15.6
- AddSensorProcessor, 3.15.7
- AddMaterial, 3.15.8
- AddBeamProperties, 3.15.9
- AddNode, 3.15.10
- Include, 3.15.11
- Print, 3.15.12
- PrintIf, 3.15.13
- ReadSTLFile, 3.15.14
- RotMat2Angles, 3.15.15
- LoadVectorFromFile, 3.15.16
- TransformPoints, 3.15.17
- ComputeInertia, 3.15.18
- Sum, 3.15.19
- Product, 3.15.20
- Transpose, 3.15.21
- CrossProduct, 3.15.22
- for, 3.15.23
- if, 3.15.24
- GenerateNewMesh, 3.15.25
- GenerateBeam, 3.15.26
- GeneratePlate, 3.15.27
• GenerateBlock, 3.15.28
• GenerateCylinder, 3.15.29
• LoadMesh, 3.15.30
• WriteMesh, 3.15.31
• Transform, 3.15.32
• Distort, 3.15.33
• Modify, 3.15.34
• Linear2Quadratic, 3.15.35
• SplitHexes, 3.15.36
• Refine, 3.15.37
• Rotate, 3.15.38
• Mirror, 3.15.39
• Extrude, 3.15.40
• AddMeshToMBS, 3.15.41
• GetNodesInBox, 3.15.42
• GetNodesInCylinder, 3.15.43
• GetNodesInSphere, 3.15.44
• GetNodesInFunction, 3.15.45
• GetNodePos, 3.15.46
• GetFacesFromNodes, 3.15.47
• GlueMesh, 3.15.48
• GetLocalPosOfGlobalPos, 3.15.49
• GetElementsInBox, 3.15.50
• GetElementAtPosition, 3.15.51
• GenerateNewPlot, 3.15.52
• ExportToFile, 3.15.53
• Close, 3.15.54
• DoesEntryExist, 3.15.55
• GetByName, 3.15.56
3.15. COMMAND

• SetByName, 3.15.57
• Compare, 3.15.58
• StrCat, 3.15.59
• Zeros, 3.15.60
• IntArrayOp, 3.15.61
• Timer, 3.15.62
• AddSet, 3.15.63
• AccessSet, 3.15.64
• GenerateConstraints, 3.15.65
• GenerateSensors, 3.15.66
• AssignMaterial, 3.15.67
• AssignLoad, 3.15.68
• ChangeProperties, 3.15.69
• SetInitialCondition, 3.15.70
• OpenCompiledModel, 3.15.71

3.15.1 AddElement

Adds an element to the system. See the description of the elements above in order to get the available options.

Parameters:
The parameter of this command is an ElementDataContainer with the data of the element.
ATTENTION: the entry element_type must exist!

return values:
The return value of this command is the number of the element in the MBS.

Example

see file AddElement.txt

emptyMass3D
{
    element_type = "Mass3D"
    Physics.mass= 1
}
nElement = AddElement(emptyMass3D)
3.15.2 AddGeomElement

This command adds a geometric element.

**Parameters:**
The parameter of this command is an ElementDataContainer with the data of the geometric element. ATTENTION: the entry geom_element_type must exist!

**Return values:**
The return value of this command is the number of the geometric element in the MBS.

**Example**

see file AddGeomElement.txt

```plaintext
myCube
{
   name = "myGeomElement"
   geom_element_type = "GeomOrthoCube3D"
   Geometry.center_point = [0.0, 0.0, 0.0]
   Geometry.size = [1.0, 1.0, 1.0]
}
AddGeomElement(myCube)
```

3.15.3 AssignGeomElementToElement

This command assigns a geom element to an element. The reference point and rotation of the element are evaluated and the settings of the geom element are modified automatically, such that the current relative orientation of element and geom element keeps the same. You can therefore add and align the geom element independently from the element first and afterwards decide to connect the geom element to the element without the need of changing the settings of the geom element again.

**Parameters:**
The parameters of this command are

1. 1st parameter: an element number
2. 2nd parameter: a geom element number

**Return values:**
returns 0 or an error code

**Example**

see file AssignGeomElementToElement.txt

```plaintext
red = [1,0,0] % colour for "relative" (geom) elements
blue = [0,0,1] % colour for "absolute" (geom) elements

geomElement % define and add some geom element
{
```
name = "absolute geom Element"
geom_element_type = "GeomCylinder3D"
Geometry.radius = 0.1  % radius of the cylinder
Geometry.axis_point1 = [1, 1, 0]  % point on axis of rotation
Geometry.axis_point2 = [1.5, 1, 0]  % point on axis of rotation
Graphics.RGB_color = blue
}

nGeomEl_absolute = AddGeomElement(geomElement)
% the geomElement is added to the mbs with global positions

geomElement.Graphics.RGB_color = red
geomElement.name = "relative geom Element"
geomElement.Geometry.axis_point1 = [0, 0, 0]  % point on axis of rotation
geomElement.Geometry.axis_point2 = [0.5, 0, 0]  % point on axis of rotation
nGeomEl_relative = AddGeomElement(geomElement)
% the geomElement is added a second time to the mbs with different
% global positions and color

elementRelative
{
    name = "relative element"
element_type = "Rigid3DMinCoord"
Graphics.use_alternative_shape = 1
Graphics.geom_elements = [nGeomEl_relative]
Graphics.position_offset = [1, 0, 0]
Graphics.RGB_color = red
}

nERel = AddElement(elementRelative)
% the geomElement "relative" is linked to the element "relative" at the time when the
% element is added to the mbs
% the coordinates of the geomElement are now relative to the reference point of the element
% the element "relative" is not visible, only the geomElement is visible

elementAbsolute
{
    name = "absolute element"
element_type = "Rigid3DMinCoord"
Graphics.position_offset = [1, 1, 0]
Graphics.RGB_color = blue
}

nEAbs = AddElement(elementAbsolute)
% the geomElement "absolute" and the element "absolute" are both visible in mbs
% you can check the alignment

AssignGeomElementToElement(nEAbs, nGeomEl_absolute)
% the element "absolute" vanishes but the geomElement "absolute" stays at the same place
% the settings of the geomElement were adjusted automatically
% the element "absolute" is linked with the geomElement "absolute"
3.15.4 AddConnector

Add a connector to the system. See the description of the connectors above in order to get the available options.

**Parameters:**
The parameter of this command is an ElementDataContainer with the data of the connector. ATTENTION: the entry element_type must exist!

**Return values:**
The return value of this command is the number of the connector in the MBS.

**Example**

see file AddConnector.txt

RigidBody \% define some element
{
  element_type = "Rigid3D"
  Physics.mass = 1
  Graphics.Body_dimensions = [0.1,1,0.1]
}
nElement = AddElement(RigidBody)

myConnector
{
  element_type = "PointJoint"
  Physics
  {
    use_penalty_formulation = 0
    Lagrange
    {
      constrained_directions = [1,1,1]
    }
  }
  Position1
  {
    element_number = nElement
    position = [0,-0.5,0]
  }
  Position2
  {
    element_number = 0 \% = 0 -> global node/coordinate
    position = [0,0,0] \% position of ground
  }
  Graphics.draw_size = 0.05
}
nConnector = AddConnector(myConnector)
3.15. COMMAND

3.15.5 AddLoad

Adds a load to the system. See the description of the loads above in order to get the available options. You have to adjust the value 'loads' in the element to assign the load to the element.

Parameters:
The parameter of this command is an ElementDataContainer with the data of the load. AT- TENTION: the entry load_type must exist!

return values:
The return value of this command is the number of the load in the MBS.

Example

see file AddLoad.txt

myLoad % define the load
{
    load_type = "Gravity"
    name = "gravity for all elements"
    direction = 2
    gravity_constant = 9.81
}
nLoad=AddLoad(myLoad)

emptyMass3D % define some element
{
    element_type = "Mass3D"
    Physics.mass= 1
    loads = [nLoad] % add the load to the element
}
nElement = AddElement(emptyMass3D)

ViewingOptions.Loads.show_loads = 1
ViewingOptions.Loads.arrow_size = 0.2

3.15.6 AddSensor

Adds a sensor to the system. See the description of the sensors above in order to get the available options.

Parameters:
The parameter of this command is an ElementDataContainer with the data of the sensor. AT- TENTION: the entry sensor_type must exist!

return values:
The return value of this command is the number of the sensor in the MBS.

Example

see file AddSensor.txt

emptyMass3D % define some element
\{ 
  element_type = "Mass3D"
  Physics.mass= 1
\} 

nElement = AddElement(emptyMass3D)

mySensor 
{ 
  sensor_type = "FVElementSensor"
  name = "Position of the Mass3D in z-direction"
  element_number = nElement
  field_variable = "position"
  component = "z"
} 

nSensor = AddSensor(mySensor)

ViewingOptions.Sensors.show_sensors = 1

3.15.7 AddSensorProcessor

Adds a sensorProcessor to an existing. See the description of the sensorProcessors above in order to get the available options.

Parameters:
The 1st parameter of this command is an ElementDataContainer with the data of the sensor followed by the index number of the sensor to add to as 2nd parameter. ATTENTION: the entry processor_type must exist!

return values:
The return value of this command is the number of the sensor in the MBS.

Example

see file AddSensorProcessor.txt

emptyMass3D  % define some element
{ 
  element_type = "Mass3D"
  Physics.mass= 1
} 

nElement = AddElement(emptyMass3D)

mySensor 
{ 
  sensor_type = "FVElementSensor"
  name = "Position of the Mass3D in z-direction"
  element_number = nElement
  field_variable = "position"
  component = "z"
} 

nSensor = AddSensor(mySensor)
3.15. COMMAND

myProcessor
{
    processor_type = "OffsetScaleSensorProcessor"
    scaling_factor = -1.0
    offset = 0.0
}
AddSensorProcessor(myProcessor,nSensor)

ViewingOptions.Sensors.show_sensors = 1

3.15.8 AddMaterial

Adds a material to the system. See the description of the materials above in order to get the available options.

**Parameters:**
The parameter of this command is an ElementDataContainer with the data of the material.

**ATTENTION:** the entry material_type must exist!

**return values:**
The return value of this command is the number of the material in the MBS.

**Example**

see file AddMaterial.txt

Material1
{
    material_type= "Material"
    Solid
    {
        density= 7850 % density (rho) for gravitational force
        youngs_modulus= 2.1e11 %Youngs modulus
        poisson_ratio= 0.3 %Poisson ratio
    }
}
AddMaterial(Material1)

3.15.9 AddBeamProperties

Adds a BeamProperty to the system. See the description of the BeamProperties above in order to get the available options.

**Parameters:**
The parameter of this command is an ElementDataContainer with the data of the BeamProperties. **ATTENTION:** the entry material_type must exist!

**return values:**
The return value of this command is the number of the node in the MBS.
Example

see file AddBeamProperties.txt

```plaintext
beam1
{
    material_type = "Beam3DProperties"
    cross_section_size = [0.1,0.1]
    EA = 2e9
    EIy = 2e6
    EIz = 2e6
    GJkx = 2e6
}
AddBeamProperties(beam1)
```

### 3.15.10 AddNode

Adds a node to the system. See the description of the nodes above in order to get the available options.

**Parameters:**
The parameter of this command is an ElementDataContainer with the data of the node. ATTENTION: the entry node_type must exist!

**Return values:**
The return value of this command is the number of the node in the MBS.

Example

see file AddNode.txt

```plaintext
node1
{
    node_type = "Node3DS1rot1"
}
AddNode(node1)
```

ViewingOptions.FiniteElements.Nodes.show = 1
ViewingOptions.FiniteElements.Nodes.node_size = 0.05

### 3.15.11 Include

This command includes a file.

**Parameters:**
The parameter of this command is the absolut or relative filename. If a relative filename is used, then the path is relativ e to the last file! Be careful, if you use this command more than one time in a file.

**Return values:**
There is no return value defined yet.
3.15. COMMAND

Example

see file Include.txt

\%Include("D:\HelloWorld.txt") % absolute file path
Include("..\..\examples\double_pendulum.txt") % relative path 1
\%Include("AddElement.txt") % relative path 2 (same folder)

3.15.12 Print

Prints a text to the output window

**Parameters:**
There are three possibilities to use the command. The parameter can either be:

- a text, e.g. Print("Hello world")
- an ElementDataContainer, e.g. Print(my_mass)
- an ElementData, e.g. Print(my_mass.density)

In the case of a text or an ElementData, only the text itself is printed. In the case of an ElementDataContainer, also the name of the ElementData is printed.

**return values:**
There is no return value for this command

Example

see file Print.txt

Print("Hello world! \n")

TestEDC
{
    number = 1
    text = "this is a text in an edc"
}
Print("\nPrinting the edc:\n")
Print(TestEDC)
Print("\n")
Print("Printing elements of the edc: \n")
Print(TestEDC.text)
Print("\n")
Print(TestEDC.number)
Print("\n")

PrintIf(true,"It’s true\n")
PrintIf(TestEDC.number,"TestEDC.number=1\n")
3.15.13  PrintIf

additional boolean, Prints a text to the output window

Parameters:
First parameter is a flag whether to print at all. There are three possibilities to use the command. The 2nd parameter can either be:

- a text, e.g. Print("Hello world!")
- an ElementDataContainer, e.g. Print(my_mass)
- an ElementData, e.g. Print(my_mass.density)

In the case of a text or an ElementData, only the text itself is printed. In the case of an ElementDataContainer, also the name of the ElementData is printed.

return values:
There is no return value for this command

Example
see file Print.txt

Print("Hello world! \n")

TestEDC
{
   number = 1
   text = "this is a text in an edc"
}
Print("\nPrinting the edc:\n")
Print(TestEDC)
Print("\n")
Print("Printing elements of the edc: \n")
Print(TestEDC.text)
Print("\n")
Print(TestEDC.number)
Print("\n")
PrintIf(true,"It’s true\n")
PrintIf(TestEDC.number,"TestEDC.number=1\n")

3.15.14  ReadSTLFile

This command reads a stl-mesh from a file and stores the data in an ElementDataContainer.

Parameters:
The parameter of this command is the absolute or relative filename.

return values:
The return value is an ElementDataContainer with 2 entries: triangles and points.
3.15. COMMAND

Example

see file ReadSTLFile.txt

STL = ReadSTLFile("mesh.stl")

myGeomElementMesh3D
{
    geom_element_type = "GeomMesh3D"
    % MeshData = STL % not possible yet
    MeshData.triangles = STL.triangles
    MeshData.points = STL.points
}

nGeom1 = AddGeomElement(myGeomElementMesh3D)

3.15.15 RotMat2Angles

For a given rotation matrix the Euler Angles Z-X-Z and Kardan Angles are computed

Parameters:
The parameters of this command are as follows

1. rotationmatrix

- return values:
The return value is a set Vector3D containing the angles [rad].

Example

see file RotMat2Angles.txt

a = 45 * PI/180 % convert to rad
rotZ = [ cos(a), -sin(a), 0; sin(a), cos(a), 0; 0, 0, 1 ]
angles1 = RotMat2Angles(rotZ)
%% expect: angles1.EulerZXZ = [a 0 0] = [0.7853981633974483 0 0]
%% expect: angles1.Kardan = [0 0 a] = [0 0 0.7853981633974483]

b = 1/sqrt(2)
rotY = [ b,0,b; 0,1,0; -b,0,b ]
angles2 = RotMat2Angles(rotY)
%% expect: angles2.EulerZXZ = [ pi/2, pi/4, -pi/2 ]
%% expect: angles2.Kardan = [0 0 a] = [0 pi/4 0]

3.15.16 LoadVectorFromFile

This command reads a vector from a file and returns this vector.

Parameters:
The parameters of this command are

1. The name of the file as string

2. An integer defining in which column (default) or row of the file the vector is stored
3. (optional) 0.. take the column (default), 1.. take the row

- **return values:**
The return value is the vector.

**Example**

see file LoadVectorFromFile.txt

```plaintext
%========== basic example ==============
% t = LoadVectorFromFile("solution.txt",1) % relative path
% x = LoadVectorFromFile("D:sol.txt",2) % absolute also possible
x = LoadVectorFromFile("solution.txt",2)
x7 = x[7] % direct access to element of vector

%========== extended example ===========
% use loaded vectors to define a MathFunction
Time.element_type= "IIDTime"
nTime = AddElement(Time) % time as input for the MathFunction

Mathf
{
element_type= "IOMathFunction"
Graphics.position= [100, 0]
IODBlock
{
  input_element_numbers= [nTime]
  input_element_types= [1] % vector with types of connected inputs; 1=IOElement
  input_local_number= [1] % i-th number of output of previous IOelement
  MathFunction
  {
    piecewise_mode= 1 % modus for piecewise interpolation: 1=linear
    piecewise_points= t % supporting points for piecewise interpolation
    piecewise_values= x % values at supporting points
  }
}
} nMF = AddElement(Mathf)

% sensor to measure the output of the mathfunction
sensor.sensor_type= "ElementSensor"
sensor.element_number= nMF
sensor.value= "IODBlock.output[1]"
AddSensor(sensor)
```

3.15.17 **TransformPoints**

With this command, the geometry described by the points can be transformed. It is possible to apply rotation and/or translation and/or scaling. The new point \( p_N \) is computed according to the formula \( p_N = \text{trans} + \text{rot} \cdot p \).
3.15. COMMAND

Parameters:
The parameters of this command are as follows

1. points: Matrix of the points: Each line represents a point \( p \). The 3 columns are the \( x \)-, \( y \)- and \( z \)-coordinate

2. trans: Vector of translation, 3 dimensions!

3. rot: rotation matrix (3x3), can be used for scaling as well as rotation

return values:
The return value is a Matrix containing the transformed points \( pN \).

Example

see file TransformPoints.txt

\[
\text{STL} = \text{ReadSTLFile("mesh.stl")} \% \text{load mesh}
\]

\[
\text{% add geomElement with original points}
\text{myGeomElementMesh3D}
\{
   \text{geom_element_type} = "GeomMesh3D"
   \text{MeshData.triangles} = \text{STL.triangles}
   \text{MeshData.points} = \text{STL.points}
   \text{Graphics.RGB_color} = [0.2,0.2,0.8]
\}
\text{nGeom1} = \text{AddGeomElement(myGeomElementMesh3D)}
\]

\[
\text{% transform points}
\text{vec} = [0,50,0] \% \text{translation}
\text{A} = [0.75,0,0;0,0.75,0;0,0,0.75] \% \text{scaling}
\text{points=TransformPoints(STL.points,vec,A)}
\]

\[
\text{% add geomElement with transformed points}
\text{myGeomElementMesh3D.MeshData.points} = \text{points}
\text{myGeomElementMesh3D.Graphics.RGB_color} = [0.2,0.8,0.2]
\text{nGeom2} = \text{AddGeomElement(myGeomElementMesh3D)}
\]

3.15.18 ComputeInertia

This command computes the mass, moment of inertia, volume and center of mass based on the information about the geometry and the material of a body

Parameters:
The parameter of this command is an ElementDataContainer, with the following entries:

- density or material\_number (one of these 2 has to be set!)

- One of the following options to define the geometry:

  - MeshData.triangles and MeshData.points
    both entries are Matrices with 3 columns
- Cube.body_dimensions

return values:
The return value is an ElementDataContainer with 4 entries: volume, mass, moment_of_inertia and center_of_mass

Example

see file ComputeInertia.txt

\%
\% simple example with a cube
\my_data
{
  density = 7850
  Cube.body_dimensions = [1.0,0.1,0.1]
}
CI1 = ComputeInertia(my_data)
Print(CI1)

\%
\% example with a mesh
\STL = ReadSTLFile("mesh.stl")
\Material1
{
  material_type= "Material"
  Solid.density= 7850
}
n = AddMaterial(Material1)

\my_data2
{
  material_number = n
  MeshData
  {
    triangles = STL.triangles
    points = STL.points
  }
}
CI2 = ComputeInertia(my_data2)
Print(CI2)

3.15.19 Sum

This command adds two components of the same type ( scalar, vector or matrix ).

Parameters:
The parameters of this command are
1. 1st summand, either scalar, vector or matrix
2. 2nd summand, either scalar, vector or matrix
3.15. COMMAND

- **return values:**
The return value is the sum of the two inputs.

**Example**

see file Sum.txt

Scalar = 1.5
Vector2D = [1,2]
Matrix2D = [0,1;2,0]
s = Sum(Scalar,Scalar) % 3
v = Sum(Vector2D,Vector2D) % [2,4]
m = Sum(Matrix2D,Matrix2D) % [0,2;4,0]

3.15.20 Product

This command multiplies two components of the type (scalar, vector or matrix) when the operation is defined.

**Parameters:**
The parameters of this command are

1. 1\textsuperscript{st} factor, either scalar, vector or matrix

2. 2\textsuperscript{nd} factor, either scalar, vector or matrix

product of two vectors is always computed as scalar product for vector times Matrix the vector is automatically transposed if required -

**return values:**
The return value is the product of the two inputs.

**Example**

see file Product2.txt

Scalar = 1.5
Vector2D = [1,2]
Matrix2D = [0,1;2,0]
s1 = Product(Scalar,Scalar) % 2.25
v1 = Product(Scalar,Vector2D) % [1.5,3]
m1 = Product(Scalar,Matrix2D) % [0,1.5;3,0]
s2 = Product(Vector2D,Vector2D)
v2 = Product(Vector2D,Scalar)
m2 = Product(Matrix2D,Scalar) % [0,1.5;3,0]
3.15.21 Transpose

This command transposes a matrix or vector.

Parameters:
The parameters of this command are

1. vector or matrix to be transposed

- return values:
The return value is a matrix or a vector.

Example

see file Transpose.txt

\[
\text{Vector2D} = [1,2] \\
a = \text{Transpose}(%\: \text{Vector2D}) \quad \% \quad [1;2] \\
b = \text{Transpose}(a) \quad \% \quad [1,2]
\]

3.15.22 CrossProduct

This command computes the cross product of two vectors.

Parameters:
The parameters of this command are

1. 1\textsuperscript{st} vector (2D or 3D)
2. 2\textsuperscript{nd} vector (2D or 3D)

for two 3D vectors the return value is also a 3D vector. For two 2D vectors the return value is a scalar. -

return values:
The return value is the scalar cross product.

Example

see file CrossProduct.txt

\[
\text{v1} = [1,2,3] \\
\text{v2} = [2,3,4] \\
\text{C1} = \text{CrossProduct}(\text{v1},\text{v2}) \quad \% \quad [-1, 2,-1] \\
\text{C2} = \text{CrossProduct}(\text{v2},\text{v1}) \quad \% \quad [ 1,-2, 1]
\]

3.15.23 for

This command starts a FOR loop for the subsequent block.

Parameters:
The parameters of this command are

1. 1\textsuperscript{st} define and initialize loop variable ("i=1")
2. $2^{nd}$ loop condition ($i<5$)

3. $3^{rd}$ loop increment ($i=i+1$)

the command must be followed by a container for the loop code.

**return values:**
The return value is the number of iterations.

---

**Example**

see file LoopCond.txt

```verbatim
%% Test 1
Test1 % general function and tree correctness
{
  sum = 0
  for(i=1,i<11,i=i+1)
  {
    sum = sum + i
  }
  Print("Test1: ")
  Print(sum)
  Print(" (55)\n")
  if(sum==55)
  {
    Print("TEST 1 PASSED \n")
  }
}

%% Test2  % nesting loops
%% Test2
%
for(i=1,i<5,i=i+1)
{
  for(j=1,j<5,j=j+1)
  {
    Mass3D
    {
      element_type = "Mass3D"
      Physics.mass= 1
      Initialization.initial_position= [i,j, 0]
      Graphics.RGB_color = [1,1,1]
    }
    if(i==j)
    {
      if(i==1)
      {
        Mass3D.Graphics.RGB_color = [0,0,0]
      }
      if(i==2)
      {  
```
{  
  Mass3D.Graphics.RGB_color = [1,0,0]  
  
  if(i==3)  
  {  
    Mass3D.Graphics.RGB_color = [0,1,0]  
    
  }  
  if(i==4)  
  {  
    Mass3D.Graphics.RGB_color = [0,0,1]  
  }  
  
  elnr = AddElement(Mass3D)  
  Print("Added Element ")  
  Print(elnr)  
  Print(" to MBS\n")  
  
}  

3.15.24 if  

This command evaluates an IF condition for the subsequent block.  

Parameters:  

The parameters of this command are  

1. 1st condition ("i<10")  

the command must be followed by a container for the conditional code -  

return values:  

The return value is the 1 for true and 0 for false.  

Example  

see file LoopCond.txt  

%% Test 1  
Test1  
  % general function and tree correctness  
  {  
    sum = 0  
    for(i=1,i<11,i=i+1)  
    {  
      sum = sum + i  
    }  
    Print("Test1: ")  
    Print(sum)  
    Print(" (55)\n")  
    if(sum==55)  
    {  
      Print("TEST 1 PASSED \n")  
    }  
}
%% Test2 % nesting loops
\%
\%
\%
\{
\ for(i=1,i<5,i=i+1)
\{
\ for(j=1,j<5,j=j+1)
\{
\ Mass3D
\{
\ element_type = "Mass3D"
\Physics.mass= 1
\ Initialization.initial_position= [i,j, 0]
\ Graphics.RGB_color = [1,1,1]
\}
\ if(i==j)
\{
\ if(i==1)
\{
\ Mass3D.Graphics.RGB_color = [0,0,0]
\ }
\ if(i==2)
\{
\ Mass3D.Graphics.RGB_color = [1,0,0]
\ }
\ if(i==3)
\{
\ Mass3D.Graphics.RGB_color = [0,1,0]
\ }
\ if(i==4)
\{
\ Mass3D.Graphics.RGB_color = [0,0,1]
\ }
\)
\ e1nr = AddElement(Mass3D)
\ Print("Added Element ")
\ Print(e1nr)
\ Print(" to MBS\n")
\}
\}
\}\}
\}
\}
\}
\}

3.15.25 GenerateNewMesh

This command generates a Handle to a Mesh Object for further operations.

Parameters:
The parameters of this command are

1. 1st parameter EDC to overwrite the default properties

the return vaule of the command MUST be assigned to a new variable(handle)
overwritable entries in the properties EDC are:

- mesh_name
- mesh_type: may be StructuralMesh (default) or SolidMesh
- compute_surface: set to 1 to automatically compute surface of the mesh

return values:
The return value is a special EDC (Handle) that must be assigned to a new variable.

Example
see file MeshGenerateMesh.txt

```
meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)
Mesh1.AddMeshToMBS(1)

meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)
Mesh2.AddMeshToMBS(1)
```

3.15.26 GenerateBeam
This command generates a Beam within a Mesh Object.
Parameters:
The parameters of this command are

1. 1st parameter EDC containing the beam properties

entries in the properties EDC are:

- component_name
- component_type: Linear(default)
- Generation.P1 - position of left outer node
- Generation.P2 - position of right outer node
- Generation.material_number - number of the material to be used for the beam elements (Beam3DProperties)
3.15. COMMAND

- Generation.discretization - number of the beam elements
- Generation.GeometricNonlinearityStatus - gnls of the generated elements

**return values:**
The return value is the component number of the newly generated beam within the mesh.

**Example**

see file MeshGenerateBeam.txt

```plaintext
meshparameters
{
  mesh_type = "StructuralMesh"
  mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

beamproperties
{
  material_type = "Beam3DProperties"
  cross_section_size = [0.1,0.1]
  EA = 2e9
  EIy = 2e6
  EIz = 2e6
  GJkx = 2e6
}
mnr_beam = AddBeamProperties(beamproperties)

beamparameters
{
  Generation.P1 = [0. 0. 0.]
  Generation.P2 = [1. 2. 3.]
  Generation.material_number = mnr_beam
  Generation.discretization = 4
}
Mesh1.GenerateBeam(beamparameters)
Mesh1.AddMeshToMBS(1)
```

3.15.27 GeneratePlate

This command generates a Plate within a Mesh Object.

**Parameters:**
The parameters of this command are

1. 1st parameter EDC containing the plate properties

entries in the properties EDC are:
• component_name
• component_type: Quadrilateral(default)
• Generation.P1 - position of first node
• Generation.P2 - position of outer node along first direction
• Generation.P3 - position of outer node along second direction
• Generation.P4 - position of node opposite to the first node
• Generation.material_number - number of the material to be used for the plate elements
• Generation.discretization - number of the plate elements both directions - enter as 2-component vector
• Generation.thickness - thickness for the plate elements
• Generation.GeometricNonlinearityStatus - gnls of the generated elements

- return values:
The return value is the component number of the newly generated plate within the mesh.

Example
see file MeshGeneratePlate.txt

meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

platematerial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
    Solid.plane = yes
    Solid.plane_stress = yes
}
mnr_plate = AddMaterial(platematerial)

plateparameters
{
    component_name = "tile_1"
    Generation.P1 = [ 0., 0., 0.]
    Generation.P2 = [ 2., 0., 0.]
    Generation.P3 = [ 0., 2., 0.]
}
3.15. COMMAND

Generation.P4 = [2., 2., 0.]
Generation.material_number = mnr_plate
Generation.discretization = [2,2]
Generation.thickness = 0.1
}
Mesh1.GeneratePlate(plateparameters)

Mesh1.AddMeshToMBS(1)

3.15.28 GenerateBlock

This command generates a Block within a Mesh Object.

Parameters:
The parameters of this command are

1. 1st parameter EDC containing the block properties

   entries in the properties EDC are:
   - component_name
   - component_type: Block (default)
   - Generation.P1 .. P8 - position of corner nodes
   - Generation.material_number - number of the material
   - Generation.discretization - number of the block elements all three directions
   - Generation.GeometricNonlinearityStatus - gnls of the generated elements

return values:
The return value is the component number of the newly generated block within the mesh.

Example

see file MeshGenerateBlock.txt

meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation
    {
        P1 = [-1.,-1.,-1.]
        P2 = [ 1.,-1.,-1.]
        P3 = [-1., 1.,-1.]
        P4 = [ 1., 1.,-1.]
        P5 = [-1.,-1., 1.]
        P6 = [ 1.,-1., 1.]
        P7 = [-1., 1., 1.]
        P8 = [ 1., 1., 1.]
    }
    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

3.15.29 GenerateCylinder

This command generates a Cylinder within a Mesh Object.

**Parameters:**
The parameters of this command are

1. 1st parameter EDC containing the cylinder properties

It is assumed that the cylinder rotates around the z-axis the y component for all points should be 0.0 entries in the properties EDC are:

- component_name
- component_type: Cylinder(default)
- Generation.P1 - inner point of the base
- Generation.P2 - outer point of the base
- Generation.P3 - inner point of the top
- Generation.P4 - outer point of the top
- Generation.material_number - number of the material
- Generation.discretization - number of the cylinder elements in radial, tangential and axial direction.
- Generation.GeometricNonlinearityStatus - gns of the generated elements
**3.15. COMMAND**

- **return values:**
The return value is the component number of the newly generated cylinder within the mesh.

**Example**

see file MeshGenerateCylinder.txt

```plaintext
meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

cylinderparameters
{
  component_name = "FullCylinder"
  component_type = "Cylinder"
  Generation
  {
    P1 = [0.0,0.0,0.0]
    P2 = [1.0,0.0,-.1]
    P3 = [0.0,0.0,0.8]
    P4 = [0.8,0.0,0.7]
  }
  Generation.Material_number = mnr_block
  Generation.discretization = [2,2,2]
}
Mesh2.GenerateCylinder(cylinderparameters)

cylinderparameters
{
  component_name = "HollowCylinder"
  Generation
  {
    P1 = [0.8,0.0,2.0]
    P2 = [1.1,0.0,2.0]
    P3 = [0.8,0.0,3.0]
    P4 = [1.2,0.0,3.0]
  }
}
Mesh2.GenerateCylinder(cylinderparameters)

Mesh2.AddMeshToMBS(1)

### 3.15.30 LoadMesh

This command loads a mesh from an external file into a Mesh Object.

**Parameters:**

The parameters of this command are

1. 1\textsuperscript{st} operation code defining the file format
2. 2\textsuperscript{nd} filename of the file containing the mesh

Currently implemented file formats and thus allowed values for the 1\textsuperscript{st} parameter are:

- Neutral3D
- Netgen2D
- STL

**Return values:**

The return value is the component number of the newly generated component containing the external mesh.

**Example**

see file MeshLoadMesh.txt

```plaintext
meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

Mesh2.LoadMesh("Neutral3D","blox.txt")

Mesh2.AddMeshToMBS(1)
```
3.15. COMMAND

3.15.31 WriteMesh

This command loads a mesh from an external file into a Mesh Object.

**Parameters:**
The parameters of this command are

1. 1*st* component number or zero for all
2. 2*nd* operation code defining the file format
3. 3*rd* filename of the file containing the mesh

currently implemented file formats and thus allowed values for the 1*st* parameter are

- Neutral3D

**Return values:**
The return value is the component number of the newly generated component containing the external mesh.

**Example**

see file MeshLoadMesh.txt

```plaintext
meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

Mesh2.LoadMesh("Neutral3D","blox.txt")

Mesh2.AddMeshToMBS(1)

3.15.32 Transform

This command applies a transformation on an entire Mesh or a single Mesh component.

**Parameters:**
The parameters of this command are

1. 1*st* parameter EDC containing the component number
2. *2nd* parameter EDC containing the transformation parameters

multiple entries for a single call are allowed and are processed in the same sequence as stated below. Entries in the transformation parameters EDC are:

- **scale** - scaling vector
- **rotate** - rotation vector - rotation angles around global axis, processed in sequence (x, y, z)
- **translate** - translation vector
- **name** - name of the component
- **Generation.GeometricNonlinearityStatus** - gnls for all nodes below - -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

- **return values:**
The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshTransformMesh.txt

```plaintext
meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.matnr = mnr_block
    Generation.discretization = [2,2,2]
}
compnr = Mesh2.GenerateBlock(blockparameters)
```

transformation
{

3.15. COMMAND

```cpp
name = "pullme"
scale = [1. 2. .5]
rotate = [0. -0.5 0.]
translate = [2. 0. 0]
}
Mesh2.Transform(1, transformation)
Mesh2.GenerateBlock(blockparameters)
Mesh2.AddMeshToMBS(1)
```

3.15.33 Distort

This command applies a distortion on an entire Mesh or a single Mesh component.

**Parameters:**
The parameters of this command are

1. 1\textsuperscript{st} parameter EDC containing the component number
2. 2\textsuperscript{nd} parameter EDC containing the distortion parameters

entries in the distortion parameters EDC are:

- fx - function to compute new x-coordinate - string, use (x,y,z) as function parameters
- fy - function to compute new y-coordinate - string, use (x,y,z) as function parameters
- fz - function to compute new z-coordinate - string, use (x,y,z) as function parameters
- name - name of the component
- Generation.GeometricNonlinearityStatus - gnls for all nodes below - -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

when no function defined the coordinate is not changed.

**return values:**
The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshDistort.txt

```cpp
meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)
```

```cpp
blockmaterial
{
    material_type = "Material"
```
 CHAPTER 3. HOTINT REFERENCE MANUAL

Solid.density = 7850
Solid.youngs_modulus = 2.1e11
Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.matnr = mnr_block
    Generation.discretization = [6,6,6]
}
comprnr = Mesh2.GenerateBlock(blockparameters)

distortion
{
    name = "halfpipe"
    fx = "x-2"
    fy = "y*(2*(x-0.5)^2+0.5"
    fz = "z"
}
Mesh2.Distort(1,distortion)

Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

3.15.34 Modify

This command applies a modification on an entire Mesh or a single Mesh component. A filter list can be used.

Parameters:
The parameters of this command are
1. 1\textsuperscript{st} parameter EDC containing the component number
2. 2\textsuperscript{nd} parameter EDC containing the modify parameters

entries in the modify parameters EDC are:
• entity - string - valid entries are: "NodeType","ElementType","MaterialNumber","GeometricNonlinearity"
• newValue - various, matching the entity... - e.g. for entity ElementType any valid ElementTypeString
• filter - integer - index number of a Set added to the system or zero
• name - name of the component

return values:
The return value is the component number of the newly generated component (usually the same number as the input component number)
Example
see file MeshModify.txt

3.15.35 Linear2Quadratic
This command converts elements for an entire Mesh or a single Mesh component.

Parameters:
The parameters of this command are
1. 1st parameter EDC containing the component number
2. 2nd parameter EDC containing the additional parameters

Entries in the additional parameters EDC are:

- name - name of the component
- Generation.GeometricNonlinearityStatus - gnls for all nodes below -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

Return values:
The return value is the component number of the newly generated component (usually the same number as the input component number)

Example
see file MeshLinear2Quadratic.txt

meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
  component_name = "TwoCubed"
  component_type = "Block"
3.15.36 SplitHexes

This command converts hexahedrals to (5)tetrahedrals, (3)pyramids or (2)prisms for an entire Mesh or a single Mesh component.

**Parameters:**
The parameters of this command are

1. 1st parameter EDC containing the component number
2. 2nd parameter EDC containing the conversion parameters
   - split - number of items to split the hexahedral into - may be (2), (3), (5)
   - name - name of the component
   - Generation.GeometricNonlinearityStatus - gnls for all nodes below - -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

**Return values:**
The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshSplitHexes.txt

```plaintext
meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)
```
3.15. COMMAND

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.matnr = mnr_block
    Generation.discretization = [2,2,2]
}

Mesh2.GenerateBlock(blockparameters)
transformationparameters.translate = [0 -2 0]
Mesh2.Transform(1,transformationparameters)
splitparameters.split=5
splitparameters.name = "Tets"
Mesh2.SplitHexes(1,splitparameters)

Mesh2.GenerateBlock(blockparameters)
transformationparameters.translate = [0 0 0]
Mesh2.Transform(2,transformationparameters)
splitparameters.split=3
splitparameters.name = "Pyramids"
Mesh2.SplitHexes(2,splitparameters)

Mesh2.GenerateBlock(blockparameters)
transformationparameters.translate = [0 2 0]
Mesh2.Transform(3,transformationparameters)
splitparameters.split=2
splitparameters.name = "Prisms"
Mesh2.SplitHexes(3,splitparameters)

Mesh2.AddMeshToMBS(1)

3.15.37 Ren
e
This command applies a refinement on an entire Mesh or a single Mesh component.
Parameters:
The parameters of this command are
1. 1st parameter EDC containing the component number
2. 2nd parameter EDC containing the refinement parameters

entries in the distortion parameters EDC are:

- Generation.level - list of subdivision levels for all nodes of the subordinate component
- Generation.method - string defining the method: default 0 for all directions. 1,2 or 3 to skip single local direction
- name - name of the component
- Generation.GeometricNonlinearityStatus - gns for all nodes below - -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain
- **return values:**
The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshRefine.txt

```plaintext
meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

platematerial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_plate = AddMaterial(platematerial)

plateparameters
{
    component_name = "6tiles"
    Generation.P1 = [ 0., 0., 0.]
    Generation.P2 = [ 3., 0., 0.]
    Generation.P3 = [ 0., 2., 0.]
    Generation.P4 = [ 3., 2., 0.]
    Generation.material_number = mnr_plate
    Generation.discretization = [3,2]
    Generation.thickness = 0.1
}
mnr1 = Mesh1.GeneratePlate(plateparameters)

refine
{
    Generation.level = [1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0] % seed for 12 nodes of the 6-element pattern
    name = "pattern"
}
Mesh1.Refine(mnr1,refine)

extrudeparameters.axis = 3
Mesh1.Extrude(mnr1,extrudeparameters)

Mesh1.AddMeshToMBS(1)
```
3.15. COMMAND

3.15.38 Rotate

This command rotates a subordinate mesh around a given axis (2D -> 3D)

**Parameters:**

The parameters of this command are

1. 1\textsuperscript{st} parameter EDC containing the component number
2. 2\textsuperscript{nd} parameter EDC containing the conversion parameters
   - axis\_number - rotates around the axis 1..x 2..y 3..z
   - angular\_segments - number of elements around the circumflex
   - total\_angle - 360 for a full rotation

Additionally, the following parameters may be specified for the extruded component

- thickness - thickness for the plates when extruding curves
- name - name of the component
- Generation.GeometricNonlinearityStatus - gnls for all nodes below - -1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

**Return values:**

The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshRotate.txt

```plaintext
meshparameters
{
  mesh\_type = "SolidMesh"
  mesh\_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material\_type = "Material"
  Solid.density = 7850
  Solid.youngs\_modulus = 2.1e11
  Solid.poisson\_ratio = 0.3
}
%% source has 7 domains
for(i=1,i<7,i=i+1)
{
  mnr\_block = AddMaterial(blockmaterial)
}
```
Mesh2.LoadMesh("Netgen2D","trigs.txt")

rotationparameters.angular_segments=16
rotationparameters.total_angle=360
rotationparameters.name = "spinner"
Mesh2.Rotate(1,rotationparameters)

Mesh2.AddMeshToMBS(1)

### 3.15.39 Mirror

This command mirrors a subordinate mesh at a given plane

**Parameters:**
The parameters of this command are

1. 1\textsuperscript{st} parameter EDC containing the component number
2. 2\textsuperscript{nd} parameter EDC containing the mirror parameters

Entries in the mirror parameters EDC are:

- plane - 1..x=0, 2..y=0, 3..z=0
- name - name of the component
- Generation.GeometricNonlinearityStatus - gnls for all nodes below -- 1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

**Return values:**
The return value is the component number of the newly generated component (usually the same number as the input component number)

**Example**

see file MeshMirror.txt

```python
meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)
```
3.15. COMMAND

```plaintext
3.15. COMMAND

```cylinderparameters
{
    component_name = "HollowCylinder"
    component_type = "Cylinder"
    Generation.P1 = [0.8,0.0,2.0]
    Generation.P2 = [1.1,0.0,2.0]
    Generation.P3 = [0.8,0.0,3.0]
    Generation.P4 = [1.2,0.0,3.0]
    Generation.material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh2.GenerateCylinder(cylinderparameters)

Mesh2.Mirror(1,1)
mirrorparams.plane = 2
mirrorparams.name = "onthewall"
Mesh2.Mirror(1,mirrorparams)

Mesh2.AddMeshToMBS(1)

3.15.40 Extrude

This command extrudes a subordinate mesh along a given axis (1D->2D, 2D->3D)
Parameters:
The parameters of this command are
1. 1st parameter EDC containing the component number
2. 2nd parameter EDC containing the conversion parameters
   • axis_number - extrude along the axis 1..x 2..y 3..z
   • discretization - number of elements along the axis
   • total_extrusion - extrusion distance
   • name - name of the component
   • Generation.GeometricNonlinearityStatus - gnls for all nodes below --1..don't change 0..Linear 1..NonlinearSmallStrain 2..NonlinearLargeStrain

- return values:
The return value is the component number of the newly generated component (usually the same number as the input component number)

Example

see file MeshExtrude.txt
CHAPTER 3. HOTINT REFERENCE MANUAL

meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

platematerial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
    Solid.plane = yes
    Solid.plane_stress = yes
}
mnr = AddMaterial(platematerial)

beamparameters.Generation.P1 = [0. 0. 0.]
beamparameters.Generation.P2 = [2. 0. 0.]
beamparameters.Generation.discretization = 2
Mesh1.GenerateBeam(beamparameters)

extrudeparameters.axis_number = 2
extrudeparameters.name = "pi*z*z*a"
Mesh1.Extrude(1,extrudeparameters)

MeshAsVar = Mesh1.AddMeshToMBS(1)

3.15.41  AddMeshToMBS

This command generates an instance of the mesh in the MBS.

Parameters:
There are no parameters for this command, for compatibility please enter a dummy '0' at this position.

return values:
The return value is the Element Data container of the added mesh (as shown in the Edit Mesh Menu).

Example
see file MeshAddMeshToMBS.txt

meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

beamproperties
3.15. COMMAND

```plaintext
{
  material_type = "Beam3DProperties"
  cross_section_size = [0.1,0.1]
  EA  = 2e9
  EIy = 2e6
  EIz = 2e6
  GJkx = 2e6
}
mnr_beam = AddBeamProperties(beamproperties)

beamparameters
{
  Generation.P1 = [0. 0. 0.]
  Generation.P2 = [1. 2. 3.]
  Generation.matnr = mnr_beam
  Generation.discretization = 4
  Generation.element_size = 0.5
}
Mesh1.GenerateBeam(beamparameters)

MeshAsModelEDCVariable = Mesh1.AddMeshToMBS(1)

3.15.42 GetNodesInBox

This command returns a list of nodes (registered to the mesh) in a given box.

Parameters:
The parameters of this command are

1. 1st parameter EDC containing the box (defined by two corner)

entries in the properties EDC are:

- P1 - position of corner 1
- P2 - position of corner 2

- return values:
The return value is a list of node numbers.

Example

see file GetNodesInBox.txt

meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)
blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation
    {
        P1 = [-1.,-1.,-1.]
        P2 = [ 1.,-1.,-1.]
        P3 = [-1., 1.,-1.]
        P4 = [ 1., 1.,-1.]
        P5 = [-1.,-1., 1.]
        P6 = [ 1.,-1., 1.]
        P7 = [-1., 1., 1.]
        P8 = [ 1., 1., 1.]
    }
    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

boxparameters.P1 = [-0.05,-0.05,-0.05]
boxparameters.P2 = [ 0.05, 0.05, 0.05]

meshpicked = Mesh2.GetNodesInBox(boxparameters)

3.15.43 GetNodesInCylinder

This command returns a list of nodes (registered to the mesh) in a given cylinder or cylinder shell.

**Parameters:**
The parameters of this command are

1. 1\textsuperscript{st} parameter EDC containing the cylinder (defined by axis and two radii)

entries in the properties EDC are:
- P1 - position of bottom center
- P2 - position of top center
- Rout - outer shell radius
3.15. COMMAND

- Rin - inner shell radius, default 0 for full cylinder

return values:
The return value is a list of node numbers.

Example

see file GetNodesInCylinder.txt

meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

cylinderparameters
{
    component_name = "FullCylinder"
    component_type = "Cylinder"

    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,1]
}
Mesh2.GenerateCylinder(cylinderparameters)
Mesh2.AddMeshToMBS(1)

delta = 1e-3
shellparameter.P1 = [0 0 0.9]
shellparameter.P2 = [0 0 1.1]
shellparameter.Rout = 1+delta

allupperhalf = Mesh2.GetNodesInCylinder(shellparameter)
% for given discretization [2,2,1] -> 20..38

shellparameter.P1 = [0 0 -0.1]
shellparameter.P2 = [0 0 1.1]
shellparameter.Rin = 0.75-delta

outerlayers = Mesh2.GetNodesInCylinder(shellparameter)
% for given discretization [2,2,1] -> 10..19, 29..38
3.15.44 GetNodesInSphere

This command returns a list of nodes (registered to the mesh) in a given sphere or spherical shell.

**Parameters:**
The parameters of this command are

1. 1st parameter EDC containing the sphere (defined by center and two radii)

Entries in the properties EDC are:

- P1 - position of center
- Rout - outer shell radius
- Rin - inner shell radius, default 0 for full cylinder

**Return values:**
The return value is a list of node numbers.

**Example**

see file GetNodesInSphere.txt

```plaintext
meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)
Mesh2.AddMeshToMBS(1)

delta = 1e-3
```
shellparameter\_P1 = \[0.5 0.5 0.5\] \% center of cube

shellparameter\_Rout = 0.75 +delta \% this excludes the corners

edgest = sqrt(2)/2 < .71

cornerdist = sqrt(3)/2 > 0.86

allbutcorners = Mesh2.GetNodesInSphere(shellparameter)
\% for given discretization [2,2,2] -> 1..27 without \{1,3,7,9, 19,21,25,27\}

shellparameter\_Rout = 0.5 +delta

shellparameter\_Rin = 0.5 -delta

allfacecenters = Mesh2.GetNodesInSphere(shellparameter)
\% for given discretization [2,2,2] -> 5, 11,13,15,17, 23

\%shellparameter

### 3.15.45 GetNodesInFunction

This command returns a list of nodes (registered to the mesh) in a region where a given function \( f(x,y,z) > 0 \).

**Parameters:**

The parameters of this command are

1. 1\(^{st}\) parameter EDC containing the function

entities in the properties EDC are:

- function - definition of the function as single string, dependencies on \(x,y,z\) as node coordinates are assumed.

- **return values:**

The return value is a list of node numbers.

**Example**

see file GetNodesInFunction.txt

meshparameters
{
  mesh\_type = "SolidMesh"
  mesh\_name = "Mesh2"
}

Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
  material\_type = "Material"
  Solid.density = 7850
  Solid.youngs\_modulus = 2.1e11
  Solid.poisson\_ratio = 0.3
}

mnr\_block = AddMaterial(blockmaterial)
blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

functionparameter.function = "x+y+z < 1.2"
cuttingplane111 = Mesh2.GetNodesInFunction(functionparameter)

% for given discretization [2,2,2] → 1,2,3, 4,5, 7, 10,11, 13, 19

delta = 0.001
functionparameter.function = "((1-delta)<(x+y))&&((x+y)<(1+delta))"
pickplane110at1 = Mesh2.GetNodesInFunction(functionparameter)
% for given discretization [2,2,2] → 3,5,7, 12,14,16, 21,23,25

%%% equivalent to GetNodesInSphere example 2
functionparameter.function = "((0.495^2)<((x-.5)^2+(y-.5)^2+(z-.5)^2)) && (((x-.5)^2+(y-.5)^2+(z-.5)^2)<(0.505^2))"
allfacecenters = Mesh2.GetNodesInFunction(functionparameter)
% for given discretization [2,2,2] → 5, 11,13,15,17, 23

3.15.46 GetNodePos

This command returns the global position of a given node.

Parameters:
The parameters of this command are

1. global node number

- return values:
The return value is a 3D position vector.

Example

see file GetNodePosition.txt

meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
Solid.density = 7850
Solid.youngs_modulus = 2.1e11
Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
  component_name = "TwoCubed"
  component_type = "Block"
  Generation.Material_number = mnr_block
  Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

node1pos = Mesh2.GetNodePos(1)
node8pos = Mesh2.GetNodePos(8)

3.15.47 GetFacesFromNodes

This command returns a list of elements and face numbers (registered to the mesh) that can be built from nodes in the input.

Parameters:
The parameters of this command are
1. 1st parameter: a set of global node numbers
2. 2nd parameter: a set of elements to process, use 0 for all elements

return values:
The return value is a container with a list of element numbers and a list of face numbers.

3.15.48 GlueMesh

This command glues mesh together.

Parameters:
The parameters of this command are
1. 1st parameter EDC containing the component numbers
2. 2nd parameter EDC containing additional parameters for the glue operation

entries in the properties EDC are:
  • usepenalty - set to one for penalty constraints, 0(default) for lagrangian

return values:
The return value is the component number of the MeshGlue-Component (usually the lowest of the input component numbers)
Example

See file MeshGlueMesh.txt

```plaintext
meshparameters
{
  mesh_type = "StructuralMesh"
  mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

platematerial
{
  material_type = "Material"
  Solid.density = 7850
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
  Solid.plane = yes
  Solid.plane_stress = yes
}
mnr_plate = AddMaterial(platematerial)

plateparameters
{
  component_name = "tile_1"
  Generation.P1 = [ 0., 0., 0.]
  Generation.P2 = [ 2., 0., 0.]
  Generation.P3 = [ 0., 2., 0.]
  Generation.P4 = [ 2., 2., 0.]
  Generation.matnr = mnr_plate
  Generation.discretization = [1,1]
  Generation.thickness = 0.1
}
Mesh1.GeneratePlate(plateparameters)

plateparameters.component_name = "tile_2"
plateparameters.Generation.P1 = [-2., 0., 0.]
plateparameters.Generation.P2 = [ 0., 0., 0.]
plateparameters.Generation.P3 = [-2., 2., 0.]
plateparameters.Generation.P4 = [ 0., 2., 0.]
Mesh1.GeneratePlate(plateparameters)

plateparameters.component_name = "tile_3"
plateparameters.Generation.P1 = [ 0.,-2., 0.]
plateparameters.Generation.P2 = [ 2.,-2., 0.]
plateparameters.Generation.P3 = [ 0., 0., 0.]
plateparameters.Generation.P4 = [ 2., 0., 0.]
Mesh1.GeneratePlate(plateparameters)

plateparameters.component_name = "tile_4"
plateparameters.Generation.P1 = [ 0., 0., 0.]
plateparameters.Generation.P2 = [ 0., 0.,-2.]
plateparameters.Generation.P3 = [ 0., 2., 0.]
```
3.15. Command

plateparameters.Generation.P4 = [0., 2., -2.]
Mesh1.GeneratePlate(plateparameters)

transformation
{
    scale = [1., 1., 1.]
    rotate = [0., -0.5, 0.]
}
Mesh1.Transform(4, transformation)

glueparams.name = "superglue"
Mesh1.GlueMesh(0, glueparams)
Mesh1.AddMeshToMBS(1)

3.15.49 GetLocalPosOfGlobalPos

This command returns the local position of a global position on a specified element.

Parameters:
The parameters of this command are
1. (integer) number of the element
2. (vector3D) global position

Return values:
The return value is a 3D vector.

Example
see file GetLocalPositionPfGlobalPosition.txt

3.15.50 GetElementsInBox

This command returns a list of elements (registered to the mesh) in a given box.

Parameters:
The parameters of this command are
1. 1st parameter EDC containing the box (defined by two corners)

Entries in the properties EDC are:
• P1 - position of first corner
• P2 - position of second corner

Return values:
The return value is a list of element numbers.
3.15.51 GetElementAtPosition

This command returns a list of elements and local positions (registered to the mesh) at a given position.

Parameters:
The parameters of this command are

1. 1st parameter EDC containing the position entries in the properties EDC are:

   - P1 - position to look at

return values:
The return value is a list of element numbers and a list of local positions numbers.

Example

see file GetElementAtPosition.txt

```plaintext
meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "Mesh2"
}
Mesh2 = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation
    {
        P1 = [-1.,-1.,-1.]
        P2 = [ 1.,-1.,-1.]
        P3 = [-1., 1.,-1.]
    }
```
3.15. COMMAND

```plaintext
P4 = [ 1., 1.,-1.]
P5 = [-1.,-1., 1.]
P6 = [ 1.,-1., 1.]
P7 = [-1., 1., 1.]
P8 = [ 1., 1., 1.]
}
Generation.Material_number = mnr_block
Generation.discretization = [2,2,2]
}
Mesh2.GenerateBlock(blockparameters)

Mesh2.AddMeshToMBS(1)

boxparameters.P1 = [-0.05,-0.05,-0.05]
boxparameters.P2 = [ 0.05, 0.05, 0.05]

meshpicked = Mesh2.GetElementAtPosition(boxparameters)

3.15.52 GenerateNewPlot

This command generates a Handle to a PlotTool Object for further operations.

Parameters:
The parameters of this command are

1. 1\textsuperscript{st} parameter EDC to overwrite the default properties

the return value of the command MUST be assigned to a new variable (handle)

overwrite entries in the properties EDC are:

- layout_file: may be StructuralMesh (default) or SolidMesh

- return values:
The return value is a special EDC (Handle) that must be assigned to a new variable.

Example

see file PlotGenreatePlot.txt

3.15.53 ExportToFile

This command saves the content PlotTool Object for further operations.

Parameters:
The parameters of this command are

1. 1\textsuperscript{st} parameter EDC containing entries for the possible export formats

available entries in the properties EDC are:

- width: width in pixels, use actual window size if not specified
• height: height in pixels, use actual window size if not specified

• export_file_jpg: generates a .jpg file "name.jpg"

• export_file_png: generates a .png file "name.jpg"

• export_file_bmp: generates a .bmp file "name.jpg"

- return values:
The return value is the number of the generated files.

Example
see file ExportPlot.txt

3.15.54 Close
This command closes the content PlotTool Object.
Parameters:
The parameters of this command are -
return values:
The return value is the filename of the generated file.

Example
see file ExportPlot.txt

3.15.55 DoesEntryExist
This command checks if the specified entry exists.
Parameters:
The parameters of this command are

1. 1st parameter: string with the name and tree of the entry to check

- return values:
returns 0 if the entry does not exist, returns 1 if the entry exists, returns 2 if the entry is an EDC
Example

see file Container.txt

Root.NodeL.NodeL.path = [ 0 0 ]
Root.NodeL.NodeR.path = [ 0 1 ]
Root.NodeR.path = [ 1 ]

Leaf = "Root.NodeL.NodeR"
flag_exist = DoesEntryExist(Leaf)

str = StrCat(Leaf,".path")
p = GetByName(str)

newpath = [1 0]
SetByName("Root.NodeR.NodeL.path",newpath)

null = 0
SetByName("Root.NodeL",null)

3.15.56 GetByName

This command lets you get any (existing) EDC entry by name.

Parameters:
The parameters of this command are

1. 1st parameter: string with the name and tree of the entry to get

- return values:
returns the entry associated with the string.

Example

see file Container.txt

Root.NodeL.NodeL.path = [ 0 0 ]
Root.NodeL.NodeR.path = [ 0 1 ]
Root.NodeR.path = [ 1 ]

Leaf = "Root.NodeL.NodeR"
flag_exist = DoesEntryExist(Leaf)

str = StrCat(Leaf,".path")
p = GetByName(str)

newpath = [1 0]
SetByName("Root.NodeR.NodeL.path",newpath)

null = 0
SetByName("Root.NodeL",null)
3.15.57  SetByName

This command lets you set any EDC entry by name, the name contains the absolute treename.  

**Parameters:**  
The parameters of this command are

1. 1\textsuperscript{st} parameter: string with the name and tree of the entry to set  
2. 2\textsuperscript{nd} parameter: the variable that should be assigned  

---

**return values:**  
has no return value.

---

**Example**

see file Container.txt

Root.NodeL.NodeL.path = [0 0]  
Root.NodeL.NodeR.path = [0 1]  
Root.NodeR.path = [1]  

Leaf = "Root.NodeL.NodeR"  
flag_exist = DoesEntryExist(Leaf)

str = StrCat(Leaf, ".path")  
p = GetByName(str)

newpath = [1 0]  
SetByName("Root.NodeR.NodeL.path", newpath)

null = 0  
SetByName("Root.NodeL", null)

3.15.58  Compare

This command compares two strings.  

**Parameters:**  
The parameters of this command are

1. 1\textsuperscript{st} parameter: string A  
2. 2\textsuperscript{nd} parameter: string B  

---

**return values:**  
returns 0 if both strings are identical, returns $>0$ or $<0$ otherwise indicating which string has higher value .
3.15. COMMAND

Example

see file strings.txt

str = "string"
strA = "string A"
strB = "string B"
str1 = "1"
str2 = "2"

mone = Compare(strA,strB)
pone = Compare(strB,strA)
zero = Compare(str,str)

stringAB = StrCat(strA,strB)
string1 = StrCat(str,str1)
string2 = StrCat(str,str2)
string12 = StrCat(string1,str2)

3.15.59 StrCat

This command joins two strings together

Parameters:
The parameters of this command are

1. 1st parameter: string A
2. 2nd parameter: string B

You can also use integer or double values instead of the strings. Inline definition of strings, e.g. StrCat("this is a ","test"), do not work properly. The spaces are not taken into account correctly! -

return values:
returns a single string - strA+strB.

Example

see file strings.txt

str = "string"
strA = "string A"
strB = "string B"
str1 = "1"
str2 = "2"

mone = Compare(strA,strB)
pone = Compare(strB,strA)
zero = Compare(str,str)

stringAB = StrCat(strA,strB)
string1 = StrCat(str,str1)
string2 = StrCat(str,str2)
string12 = StrCat(string1,str2)
3.15.60  Zeros

This command sets a vector or matrix variable to a given dimension and sets all entries to 0

**Parameters:**
The parameters of this command are

1. 1\(^{\text{st}}\) parameter: length of the vector or first dimension of the matrix

2. 2\(^{\text{nd}}\) parameter: second dimension of the matrix

**return values:**
returns a vector or matrix variable

**Example**

see file lists.txt

```matlab
mat22 = Zeros(2,4)   % matrix
vec3 = Zeros(3,1)    % vector
vec3t = Zeros(1,3)   % matrix

vec3[1] = 1
mat22[1,1] = 11
mat22[1,2] = 12

squares = Zeros(10,1)
for(i=1,i<11,i=i+1)
{
    squares[i] = i*i
}

c = cols(vec3)
r = rows(vec3)

c2 = cols(vec3t)
r2 = rows(vec3t)

c3 = cols(mat22)
r3 = rows(mat22)

mat23 = [ 1 2
             3 4
             5 6]

Sensorpositions = [-0.0781,0.1457,0;-0.1033,0.1953,0;-0.1252,0.2384,0;-0.2647,0.1908,0;-0.2411,0.2199,0;-0.2142,0.2461,0
i=2
x= Sensorpositions[i,1]
y= Sensorpositions[i,2]
CurPos=[x,y,0]
```
3.15. COMMAND

3.15.61 IntArrayOp

This command allows operations on up to two integer arrays, length of an array may be one, some operations only require one array to operate on.

Parameters:
The parameters of this command are

1. 1st parameter: type of operation as string
2. 2nd parameter: first array operand
3. 3rd parameter: second array operand

allowed operations are:

1. operation 'Append' or 'Add' - adds the second array to the tail of the first array
2. operation 'Union', 'Inter' or 'Diff' - computes union, intersection and difference set of the two arrays
3. operation 'Asc' and 'Desc' - sorts the array, only the first input is processed
4. operation 'Find' - returns a list of index numbers where numbers of the second array occur in the first
5. operation 'Unique' removes multiple entries in the first array
6. operation 'AddConst', 'MulConst'
7. operation 'Sequence'

return values:
returns a vector variable

Example

see file ArrayOps.txt

primes = [ 2 3 5 7 11 ]
fibs = [ 1 1 2 3 5 8 ]
squares = [ 1 4 9 16 25 ]
even = [ 2 4 6 8 10 ]
odd = [ 1 3 5 7 9 ]
empty = [ ]

res1 = IntArrayOp("Append",primes,13)
res2 = IntArrayOp("Add",0,fibs)

res3 = IntArrayOp("Union",odd,even)
res4 = IntArrayOp("Inter",primes,fibs)
res5 = IntArrayOp("Diff",squares,odd)

res6 = IntArrayOp("Desc",even,0)
res7 = IntArrayOp("Unique",fibs,0)

res8 = IntArrayOp("Append",empty,even)
res9 = IntArrayOp("Append",odd,empty)

res10 = IntArrayOp("Find",squares,odd)
res11 = IntArrayOp("Find",primes,even)
res12 = IntArrayOp("Find",fibs,7)

res13 = IntArrayOp("AddConst",even,8)
res14 = IntArrayOp("MultConst",odd,2)
res15 = IntArrayOp("AddArrays",odd,even)

res16 = IntArrayOp("Sequence",1,10)
res17 = IntArrayOp("Sequence",4,-4)

empty = [1337] % otherwise Popup for len 0 vector variable

3.15.62 Timer

This command returns the current system time in milliseconds

Parameters:
The parameters of this command are

1. 0..no output, 1.. line in output window and log file

- return values:
returns an integer variable

Example

see file Timer.txt

tic = Timer(0)

for(i=1,i<=1000,i=i+1)
{
    Print("still counting... ")
    Print(i)
    Print("...
")
}
toc = Timer(0)

spent = (toc-tic)
Print("time passed: ")
Print(spent)
Print("\n")

now = Timer(1)
3.15. COMMAND

3.15.63 AddSet

Adds a set to the system. See the description of the set above in order to get the available options.

Parameters:
The parameter of this command is an ElementDataContainer with the data of the set. ATTENTION: the entry set_type must exist!

return values:
The return value of this command is the number of the set in the MBS.

Example

see file AddSet.txt

myRigid
{
  element_type = "Rigid3D"  %specification of element type.
}
AddElement(myRigid)
myRigid.Initialization.initial_position = [1, 0, 0]
AddElement(myRigid)
Set1
{
  set_name = "SetOfElements1"
  set_type = "ElementSet"
  element_numbers = [1, 2]
}
nSet1 = AddSet(Set1)

3.15.64 AccessSet

This command makes a set accessible for script by copying it to Model variables

Parameters:
The parameters of this command are

1. 1st parameter: a setnumber

return values:
returns the EDC of the set

Example

see file AccessSet.txt

meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "Mesh"
Mesh = GenerateNewMesh(meshparameters)

blockmaterial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_block = AddMaterial(blockmaterial)

blockparameters
{
    component_name = "TwoCubed"
    component_type = "Block"
    Generation.Material_number = mnr_block
    Generation.discretization = [2,2,2]
}
Mesh.GenerateBlock(blockparameters)

Mesh.AddMeshToMBS(1)

SetDiag
{
    set_name = "nodes_diagonal"
    set_type = "GlobalNodeSet"
    global_node_numbers = [1 14 27]
}
nSetDiag = AddSet(SetDiag)

diagonal = AccessSet(nSetDiag)

3.15.65 GenerateConstraints

This command generates constraints for the given set

Parameters:
The parameters of this command are

1. 1st parameter: a set of global node numbers (more types will be added)
2. 2nd parameter: parameters for the constraints

entries in the properties EDC are:

- mode: 'ground' or 'pair'(default)
- type: type string for the constraint to use

return values:
returns a list of element numbers for the generated constraints
Example

see file GenerateConstraints.txt

```%
%%===================================================================%
%% Define Material(s)
%%===================================================================
blockmaterial
{
  material_type = "Material"
  name = "BlockMaterial"
  Solid.density = 7800
  Solid.youngs_modulus = 2.1e11
  Solid.poisson_ratio = 0.3
}
mnr_steelhomogen = AddMaterial(blockmaterial)

%%===================================================================%
%% Generate Mesh
%%===================================================================
meshparameters
{
  mesh_type = "SolidMesh"
  mesh_name = "theMesh"
}
theMesh = GenerateNewMesh(meshparameters)

blockparameters
{
  component_name = "UpperBlock"
  component_type = "Block"
  Generation.Material_number = mnr_steelhomogen
  Generation.discretization = [2 2 2]
}
theMesh.GenerateBlock(blockparameters)

transformation.translate = [0.2 0 1]
theMesh.Transform(1,transformation)

blockparameters.component_name = "LowerBlock"
blockparameters.Generation.discretization = [3 3 3]
theMesh.GenerateBlock(blockparameters)

%% add to mbs
MeshAsVar = theMesh.AddMeshToMBS(1)

%% connect - bottom nodes of upper block ('source') to elements of lower block ('target')
NodeSet.set_type = "GlobalNodeSet"
NodeSet.global_node_numbers = [1,2,3,4,5,6,7,8,9]  %% nodes at bottom of upper block
nNodeSet = AddSet(NodeSet)

ElemSet.set_type = "ElementSet"
ElemSet.element_numbers = MeshAsVar.LowerBlock.list_of_elements```
nElemSet = AddSet(ElemSet)

PJTemplate
{
    element_type = "PointJoint"
    Graphics.draw_size = 0.1
}
constraintparams.template = PJTemplate
constraintparams.type = "PointJoint"
constraintparams.mode = "find"
constraintparams.findfilter = nElemSet
nConstr = GenerateConstraints(nNodeSet, constraintparams)

%% ground constraints
CConstrTemplate
{
    element_type = "CoordinateConstraint"
    Coordinate1.element_number = 1
    Graphics.draw_size = 0.1
}
constraintparameter.type = "CoordinateConstraint"
constraintparameter.mode = "ground"
constraintparameter.template = CConstrTemplate
constraintparameter.coordinate1 = 3
NodeSet.global_node_numbers = [28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]
nNodeSet = AddSet(NodeSet)
 cnrsZ = GenerateConstraints(nNodeSet, constraintparameter)

constraintparameter.coordinate1 = 2
NodeSet.global_node_numbers = [28, 29, 30, 31]
nNodeSet = AddSet(NodeSet)
 cnrsX = GenerateConstraints(nNodeSet, constraintparameter)

constraintparameter.coordinate1 = 1
NodeSet.global_node_numbers = [28, 32, 36, 40]
nNodeSet = AddSet(NodeSet)
 cnrsY = GenerateConstraints(nNodeSet, constraintparameter)

%% topload
NodeSet.global_node_numbers = [19, 20, 21, 22, 23, 24, 25, 26, 27]
nNodeSet = AddSet(NodeSet)
facet6 = theMesh.GetFacesFromNodes(nNodeSet, 0)

ElemSet2.set_type = "ElementSet"
ElemSet2.element_numbers = [5, 6, 7, 8]
nElemSet = AddSet(ElemSet2)

AreaLoad
{
    name = "TopLoad"
    load_type = "AreaLoad"
3.15. COMMAND

```plaintext
pressure = 1000
    element_numbers = facet6.element_numbers
    element_facet_numbers = facet6.element_facet_numbers
}`
lnr = AddLoad(AreaLoad)
AssignLoad(nElemSet,lnr)

3.15.66 GenerateSensors

This command generates sensors for the given set

**Parameters:**
The parameters of this command are

1. 1st parameter: a set of global node numbers (more types will be added)
2. 2nd parameter: parameters for the sensors

Entities in the properties EDC are:

- template - a template container containing SensorType etc...

**Return values:**
returns a list of sensor numbers for the generated sensors

**Example**

see file GenerateSensors.txt

```plaintext
blockmaterial
{
    material_type = "Material"
    name = "BlockMaterial"
    Solid.density = 7800
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_steelhomogen = AddMaterial(blockmaterial)

meshparameters
{
    mesh_type = "SolidMesh"
    mesh_name = "theMesh"
}
theMesh = GenerateNewMesh(meshparameters)

discrx = 4
discry = 3
discrz = 2
blockparameters
{
    component_name = "UnitBlock"
```
```plaintext
component_type = "Block"
Generation.Material_number = mnr_steelhomogen
Generation.discretization = [discrx discry discrz]
}
theMesh.GenerateBlock(blockparameters)
MeshAsVariable = theMesh.AddMeshToMBS(1)

offset_top = (discrx+1)*(discry+1)*(discrz)
nnrs = [ 1+offset_top (1+discrx)+offset_top 1+discry*(1+discrx)+offset_top (1+discry)*(1+discrx)
      Set1
      {
        set_name = "CornerNodes"
        set_type = "GlobalNodeSet"
        global_node_numbers = nnrs
      }
nSet1 = AddSet(Set1)

sensorparameter.template.sensor_type = "FVElementSensor"
sensorparameter.template.field_variable = "displacement"
sensorparameter.template.component = "z"

snrs = GenerateSensors(nSet1,sensorparameter)

3.15.67 AssignMaterial
This command sets the material number of all elements of the element-set to a given number
Parameters:
The parameters of this command are
1. 1st parameter: a set of elements
2. 2nd parameter: material number to be assigned

return values:
returns 0 or an error code

Example
see file AssignMaterial.txt

beam_material
{
  material_type = "Beam3DProperties"
  cross_section_size = [0.1,0.1]
}
AddBeamProperties(beam_material)
AddBeamProperties(beam_material)
node
{
  node_type = "Node3DRxyz"
```
3.15. COMMAND

n1 = AddNode(node)
node.Geometry.reference_position = [1,0,0]

n2 = AddNode(node)
beam
{
    element_type = "LinearBeam3D"
    Physics.material_number = 1
    Geometry.node_1 = n1
    Geometry.node_2 = n2
}

nBeam = AddElement(beam)

Set1
{
    set_name = "ElementSet"
    set_type = "ElementSet"
    element_numbers = [1]
}

nSet1 = AddSet(Set1)

AssignMaterial(nSet1,2)  \% set with number nSet - assign material 2
AssignMaterial("ElementSet",1)  \% set with name "ElementSet" - assign material 1
AssignMaterial(1,2)  \% first set - assign material 2

3.15.68 AssignLoad

This command adds a load to all elements of the element-set

Parameters:
The parameters of this command are

1. 1st parameter: a set of elements

2. 2nd parameter: load number to be added or "ClearAll" to remove all loads

- return values:
  returns 0 or an error code

Example

see file AssignLoad.txt

beam_material
{
    material_type = "Beam3DProperties"
    cross_section_size = [0.1,0.1]
}

AddBeamProperties(beam_material)
AddBeamProperties(beam_material)
node{
   node_type = "Node3DRxyz"
}
n1 = AddNode(node)
node.Geometry.reference_position = [1,0,0]
n2 = AddNode(node)
beam{
   element_type = "LinearBeam3D"
   Physics.material_number = 1
   Geometry.node_1 = n1
   Geometry.node_2 = n2
}
nBeam = AddElement(beam)

Gravity{
   load_type = "Gravity"
   name = "Gravity"
   direction = 3
   gravity_constant = -9.81
}
gravnr = AddLoad(Gravity)

Set1{
   set_name = "ElementSet"
   set_type = "ElementSet"
   element_numbers = [1]
}
nSet1 = AddSet(Set1)

AssignLoad(nSet1,1)  \% set with number nSet - assign load 1
AssignLoad("ElementSet","Gravity")  \% set with name "ElementSet" - assign load named "Gravity"

3.15.69  ChangeProperties

This command changes properties of the elements of the set

**Parameters:**
The parameters of this command are

1. 1\textsuperscript{st} parameter: a set of elements or global nodes
2. 2\textsuperscript{nd} parameter: EDC containing substitute parameters EDC

**Return values:**
returns 0 or an error code
Example

see file ChangeProperties.txt

meshparameters
{
    mesh_type = "StructuralMesh"
    mesh_name = "Mesh1"
}
Mesh1 = GenerateNewMesh(meshparameters)

platematerial
{
    material_type = "Material"
    Solid.density = 7850
    Solid.youngs_modulus = 2.1e11
    Solid.poisson_ratio = 0.3
}
mnr_plate = AddMaterial(platematerial)

plateparameters
{
    component_name = "tile_1"
    Generation.P1 = [ 0., 0., 0.]
    Generation.P2 = [ 2., 0., 0.]
    Generation.P3 = [ 0., 2., 0.]
    Generation.P4 = [ 2., 2., 0.]
    Generation.matnr = mnr_plate
    Generation.discretization = [3,3]
    Generation.thickness = 0.1
}
Mesh1.GeneratePlate(plateparameters)

MeshAsVariable = Mesh1.AddMeshToMBS(1)

Set1
{
    set_name = "SomeElements"
    set_type = "ElementSet"
    element_numbers = [1,3,7,9]
}
nSet1 = AddSet(Set1)
invisible.Graphics.show_element = 0
ChangeProperties(nSet1,invisible)

Set2
{
    set_name = "AllNodes"
    set_type = "GlobalNodeSet"
    global_node_numbers = MeshAsVariable.list_of_nodes
}
nSet2 = AddSet(Set2)
initvel.Initialization.node_initial_values = [0,0,0, 0,0,0, 0,0,0, 0,0,1, 0,0,0, 0,0,0, 0,0,0]
ChangeProperties(nSet2,initvel)

### 3.15.70 SetInitialCondition

This command sets the initial condition of all members of the element or node set.

**Parameters:**

The parameters of this command are

1. 1st parameter: a set of elements or global nodes
2. 2nd parameter: index of initial value
3. 3rd parameter: expression of the value. Can contain:
   - "x" for the global reference position of the node or element
   - "y" for the global reference position of the node or element
   - "z" for the global reference position of the node or element

### 3.15.71 OpenCompiledModel

This command loads a model from the compiled models

**Parameters:**

The parameters of this command are

1. 1st parameter: string with the name of the model to load

**return values:**

returns 0 for fail, 1 for successful load
3.16 Options

These options are available:

- SolverOptions 3.16.1
- LoggingOptions 3.16.2
- GeneralOptions 3.16.3
- ViewingOptions 3.16.4
- PlotToolOptions 3.16.5

SolverOptions can be saved in the GUI separately of the other HOTINT options.

3.16.1 SolverOptions

Data objects of SolverOptions:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverOptions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SolverOptions.start_time</td>
<td>double</td>
<td>0</td>
<td>Starting time of simulation, usually 0; for static and timeint solver</td>
</tr>
<tr>
<td>SolverOptions.end_time</td>
<td>double</td>
<td>10</td>
<td>Final simulation time; for static and timeint solver</td>
</tr>
<tr>
<td>SolverOptions.do_static_computation</td>
<td>bool</td>
<td>0</td>
<td>Do only static computation; velocities and acceleration terms are ignored; system may not have kinematic degrees of freedom.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.max_step_size</td>
<td>double</td>
<td>0.001</td>
<td>Maximal step size of timeint solver.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.min_step_size</td>
<td>double</td>
<td>0.0001</td>
<td>Minimal step size of timeint solver.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.max_index</td>
<td>integer</td>
<td>2</td>
<td>Maximum index which solver the solver needs to handle</td>
</tr>
<tr>
<td>SolverOptions.Timeint.tableau_name</td>
<td>string</td>
<td>&quot;LobattoIIA&quot;</td>
<td>Runge Kutta tableau chosen</td>
</tr>
<tr>
<td>SolverOptions.Timeint.max_stages</td>
<td>integer</td>
<td>2</td>
<td>Number of stages for simulation, max. stages for variable order.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.min_stages</td>
<td>integer</td>
<td>1</td>
<td>Min. stages for variable order.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.automatic_stepsize_control</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... Full adaptive stepsize selection of timeint is (not) active.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.init_step_size</td>
<td>double</td>
<td>0.01</td>
<td>Initial stepsize for timeint.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.absolute_accuracy</td>
<td>double</td>
<td>0.01</td>
<td>Absolute accuracy, for full adaptive timeint.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.relative_accuracy</td>
<td>double</td>
<td>1</td>
<td>Relative accuracy, for full adaptive timeint.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.variable_order</td>
<td>integer</td>
<td>0</td>
<td>1(0) ... Variable order algorithm is (not) active.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.do_implicit_integration</td>
<td>bool</td>
<td>1</td>
<td>1 .. Use implicit integration, 0 .. use explicit integration.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.reset_after_simulation</td>
<td>bool</td>
<td>1</td>
<td>Reset start time and initial values after each simulation.</td>
</tr>
<tr>
<td>SolverOptions.Timeint.assume_constant_mass_matrix</td>
<td>bool</td>
<td>0</td>
<td>Experimental version of constant mass matrix (WARNING: experimental only)</td>
</tr>
</tbody>
</table>
### SolverOptions.Static

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_load_inc</td>
<td>double</td>
<td>1e-012</td>
<td>Minimal increment.</td>
</tr>
<tr>
<td>max_load_inc</td>
<td>double</td>
<td>1</td>
<td>Maximum load increment.</td>
</tr>
<tr>
<td>init_load_inc</td>
<td>double</td>
<td>1</td>
<td>Initial load increment.</td>
</tr>
<tr>
<td>load_inc_up</td>
<td>double</td>
<td>2</td>
<td>Increase load increment if success very often.</td>
</tr>
<tr>
<td>load_inc_down</td>
<td>double</td>
<td>2</td>
<td>Decrease load increment if no success.</td>
</tr>
<tr>
<td>increase_load_inc_steps</td>
<td>integer</td>
<td>1</td>
<td>If increase_load_inc_steps successful steps -&gt; leads to increase of load increment.</td>
</tr>
<tr>
<td>spring_regularisation_parameter</td>
<td>double</td>
<td>0</td>
<td>Spring-type regularisation parameter to stabilize almost kinematic systems during static comp.</td>
</tr>
<tr>
<td>use_tolerance_relax_factor</td>
<td>integer</td>
<td>0</td>
<td>Enables/disables [1/0] the use of the relaxation factor on the tolerance goal (discontinuous accuracy) within static comp. Relaxation depends on load factor (0..1).</td>
</tr>
<tr>
<td>max_tolerance_relax_factor</td>
<td>double</td>
<td>10</td>
<td>Upper bound for relaxation factor on the tolerance goal (discontinuous accuracy).</td>
</tr>
<tr>
<td>experimental_sparse_jacobian</td>
<td>bool</td>
<td>1</td>
<td>Experimental: optimized (low memory) sparse Jacobian matrix.</td>
</tr>
</tbody>
</table>

### SolverOptions.Newton

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relative_accuracy</td>
<td>double</td>
<td>1e-008</td>
<td>Relative accuracy for Newton method</td>
</tr>
<tr>
<td>absolute_accuracy</td>
<td>double</td>
<td>100</td>
<td>Absolute accuracy for Newton method</td>
</tr>
<tr>
<td>num_diff_parameter</td>
<td>double</td>
<td>1e-007</td>
<td>Numerical differentiation parameter</td>
</tr>
<tr>
<td>use_central_diff_quotient</td>
<td>bool</td>
<td>1</td>
<td>Use central difference quotient for numerical differentiation (slower).</td>
</tr>
<tr>
<td>use_modified_newton</td>
<td>bool</td>
<td>1</td>
<td>Use modified Newton (approximated Jacobian, much faster).</td>
</tr>
<tr>
<td>max_modified_newton_steps</td>
<td>integer</td>
<td>12</td>
<td>Max. modified Newton steps.</td>
</tr>
<tr>
<td>max_restart_newton_steps</td>
<td>integer</td>
<td>15</td>
<td>Max. modified Newton steps after restart.</td>
</tr>
<tr>
<td>max_full_newton_steps</td>
<td>integer</td>
<td>25</td>
<td>Max. full Newton steps.</td>
</tr>
<tr>
<td>use_trust_region</td>
<td>bool</td>
<td>0</td>
<td>0...do not use trust region; 1.use line search algorithm for newton's method, usually not necessary.</td>
</tr>
<tr>
<td>trust_region_division</td>
<td>double</td>
<td>0.1</td>
<td>Increment for line search.</td>
</tr>
<tr>
<td>low_contractivity_tolerance</td>
<td>double</td>
<td>0.7</td>
<td>Used in modified Newton: if ratio error over last error violates this bound more than twice, then Jacobian is recomputed.</td>
</tr>
<tr>
<td>high_contractivity_tolerance</td>
<td>double</td>
<td>2</td>
<td>Used in modified Newton: if ratio error over last error violates this bound more than twice, then switch to classical Newton method.</td>
</tr>
</tbody>
</table>

### SolverOptions.Eigensolver

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>do_eigenmode_computation</td>
<td>bool</td>
<td>0</td>
<td>This overwrites the dostaticcomputation flag and activates eigenmode computation on button START.</td>
</tr>
<tr>
<td>reuse_last_eigenvectors</td>
<td>bool</td>
<td>0</td>
<td>Reuse eigenvectors from last computation (faster, but might be eigenvectors from different system).</td>
</tr>
</tbody>
</table>
### 3.16. OPTIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverOptions.Eigensolver.n_eigvals</td>
<td>integer</td>
<td>3</td>
<td>Number of eigenvalues and eigenmodes to be computed for sparse iterative methods.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.max_iterations</td>
<td>integer</td>
<td>1000</td>
<td>Maximum number of iterations for iterative eigenvalue solver.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.solver_type</td>
<td>integer</td>
<td>0</td>
<td>Solver type for eigenvalue computations: 0.direct (LAPACK), 1.Arnoldi (Matlab), 2.LOBPCG (HotInt).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.n_zero_modes</td>
<td>integer</td>
<td>0</td>
<td>Number of zero eigenvalues (convergence check).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.use_n_zero_modes</td>
<td>bool</td>
<td>0</td>
<td>Check convergence for zero eigenvalues.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.use_preconditioning</td>
<td>bool</td>
<td>0</td>
<td>Use preconditioner ( \text{inv}(K + \lambda M) ).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.accuracy</td>
<td>double</td>
<td>(1e-10)</td>
<td>Tolerance for iterative Eigenvalue solver.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.preconditioner_lambda</td>
<td>double</td>
<td>1</td>
<td>(\lambda) for preconditioner ( \text{inv}(K + \lambda M) ).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.eigenmodes_scaling_factor</td>
<td>double</td>
<td>1</td>
<td>Scaling factor for the eigenmodes.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.eigenmodes_normalization_mode</td>
<td>integer</td>
<td>0</td>
<td>0 (standard)... (\max(v) = 1), 1... (v'v = 1).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.linearize_about_actual_solution</td>
<td>bool</td>
<td>0</td>
<td>Use actual solution as configuration for linearization of (K/M).</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.use_gyroscopic_terms</td>
<td>bool</td>
<td>0</td>
<td>Use gyroscopy terms for Eigenvalue computation.</td>
</tr>
<tr>
<td>SolverOptions.Eigensolver.eigval_outp_format_flag</td>
<td>integer</td>
<td>3</td>
<td>(1).. eigenfreq., (2).. eigenvec., (4).. eigenfreq. in Hz (otherwise in rad/s).</td>
</tr>
<tr>
<td>SolverOptions.Linalg.use_sparse_solver</td>
<td>bool</td>
<td>0</td>
<td>Sparse Jacobian and sparse solver is (not) activated.</td>
</tr>
<tr>
<td>SolverOptions.Linalg.undetermined_system</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... Solve system which is overdetermined (least squares solution) or underdetermined (minimum norm solution) via LAPACK routine dgels.</td>
</tr>
<tr>
<td>SolverOptions.Linalg.estimated_condition_number</td>
<td>double</td>
<td>(1e+012)</td>
<td>Used for considering equations to be linearly dependent when solving undetermined systems. Use together with option undetermined_system.</td>
</tr>
<tr>
<td>SolverOptions.Discontinuous.absolute_accuracy</td>
<td>double</td>
<td>0.0001</td>
<td>Accuracy for discontinuous problems (plasticity, contact, friction, ...).</td>
</tr>
<tr>
<td>SolverOptions.Discontinuous.max_iterations</td>
<td>integer</td>
<td>8</td>
<td>Max. number of iterations for discontinuity problems.</td>
</tr>
<tr>
<td>SolverOptions.Discontinuous.ignore_max_iterations</td>
<td>bool</td>
<td>0</td>
<td>Continue anyway if error goal is not reached after max discontinuous iterations.</td>
</tr>
<tr>
<td>SolverOptions.Solution.write_solution</td>
<td>bool</td>
<td>1</td>
<td>((0)(1)) ... (Don’t) write results to file.</td>
</tr>
<tr>
<td>SolverOptions.Solution.write_solution_every_x_step</td>
<td>integer</td>
<td>1</td>
<td>Write solution every xx steps.</td>
</tr>
<tr>
<td>SolverOptions.Solution.immediately_write_file</td>
<td>bool</td>
<td>1</td>
<td>(1) ... SLOW: immediately write data to file with ‘(\leq) flush’ (no buffering), (0)=FAST.</td>
</tr>
<tr>
<td>SolverOptions.Solution.always_replace_files</td>
<td>bool</td>
<td>0</td>
<td>(1) = always replace files, (0) = append solution to files.</td>
</tr>
<tr>
<td>SolverOptions.Solution.SolutionFile.write_solution_file_header</td>
<td>bool</td>
<td>1</td>
<td>Write solution file header.</td>
</tr>
<tr>
<td>SolverOptions.Solution.SolutionFile. solution_file_header_comment</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Comment written in solution file header.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>SolverOptions.Solution.SolutionFile.output_format</td>
<td>integer</td>
<td>0</td>
<td>(0) fixed point, (1) scientific with exp, (2) floating point notation in solution files</td>
</tr>
<tr>
<td>SolverOptions.Solution.SolverFile.ParameterFile.write_final_sensor_values</td>
<td>bool</td>
<td>1</td>
<td>Write final sensor values into parameter file.</td>
</tr>
<tr>
<td>SolverOptions.Solution.store_solution_state</td>
<td>integer</td>
<td>0</td>
<td>Store final solution state in file.</td>
</tr>
<tr>
<td>SolverOptions.Solution.store_solution_state_name</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Filename for final solution state storage.</td>
</tr>
<tr>
<td>SolverOptions.Solution.load_solution_state</td>
<td>integer</td>
<td>0</td>
<td>Load initial configuration from file.</td>
</tr>
<tr>
<td>SolverOptions.Solution.load_solution_state_name</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Filename for initial configuration.</td>
</tr>
<tr>
<td>SolverOptions.Solution.Sensor.postproc_compute_eigenvalues</td>
<td>bool</td>
<td>0</td>
<td>Compute eigenvalues in postprocessing.</td>
</tr>
<tr>
<td>SolverOptions.Solution.Sensor.output_precision</td>
<td>integer</td>
<td>17</td>
<td>Decimal precision for the floating-point values in solution files</td>
</tr>
<tr>
<td>SolverOptions.Element.store_finite_elements_matrices</td>
<td>bool</td>
<td>1</td>
<td>Store intermediate matrices for finite elements (faster, but uses huge memory).</td>
</tr>
<tr>
<td>SolverOptions.Element.element_wise_jacobian</td>
<td>bool</td>
<td>1</td>
<td>Jacobian is computed only for each element, taking into account known couplings.</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.activate</td>
<td>bool</td>
<td>0</td>
<td>Do multiple computations by varying a parameter in a certain range.</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.geometric</td>
<td>bool</td>
<td>0</td>
<td>Vary parameter geometrically (a<em>x, a</em>a<em>x, a</em>a<em>a</em>x,...).</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.start_value</td>
<td>double</td>
<td>0</td>
<td>Start value for parameter variation.</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.end_value</td>
<td>double</td>
<td>0</td>
<td>Final value for parameter variation.</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.arithmetic_step</td>
<td>double</td>
<td>1</td>
<td>Arithmetic step size for parameter variation.</td>
</tr>
<tr>
<td>SolverOptions.ParameterVariation.geometric_step</td>
<td>double</td>
<td>2</td>
<td>Geometric factor for parameter variation.</td>
</tr>
</tbody>
</table>
### 3.16 OPTIONS

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.MBS_EDC_variable_name</th>
<th>string</th>
<th>&quot;MBS_EDC_variable_name&quot;</th>
<th>Path and variable name in MBS EDC which shall be varied in parameter variation.</th>
</tr>
</thead>
</table>

#### SolverOptions.ParameterVariation.Var2

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.activate</th>
<th>bool</th>
<th>0</th>
<th>Do multiple computations by varying a parameter in a certain range.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.geometric</th>
<th>bool</th>
<th>0</th>
<th>Vary parameter geometrically ((a^x, a^{a^x}, a^{a^{a^x}}, \ldots)).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.start_value</th>
<th>double</th>
<th>0</th>
<th>Start value for parameter variation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.end_value</th>
<th>double</th>
<th>0</th>
<th>Final value for parameter variation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.arithmetic_step</th>
<th>double</th>
<th>1</th>
<th>Arithmetic step size for parameter variation.</th>
</tr>
</thead>
</table>

|------------------------------------------------------|---------|---|-------------------------------------------|

<table>
<thead>
<tr>
<th>SolverOptions.ParameterVariation.Var2.MBS_EDC_variable_name</th>
<th>string</th>
<th>&quot;MBS_EDC_variable_name&quot;</th>
<th>Path and variable name in MBS EDC which shall be varied in parameter variation.</th>
</tr>
</thead>
</table>

#### SolverOptions.Optimization

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.activate</th>
<th>bool</th>
<th>0</th>
<th>Do multiple computations by genetic optimization of parameter(s) in a certain range.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.run_with_nominal_parameters</th>
<th>bool</th>
<th>0</th>
<th>(01\ldots) (Don't) perform single simulation with nominal parameters.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.sensors</th>
<th>integer</th>
<th>0</th>
<th>Define sensor number(s) here: (the sum of) the end value(s) of the sensor signal time history is defined as cost function. The use of more than one sensor is planned.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.restart</th>
<th>bool</th>
<th>0</th>
<th>(01\ldots) (Don't) continue parameters optimization based on existing parameter file. 0: create new parameter file, 1: append to existing parameter file.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.method</th>
<th>string</th>
<th>&quot;Genetic&quot;</th>
<th>Genetic: optimize using random parameters, best parameters are further tracked.</th>
</tr>
</thead>
</table>

#### SolverOptions.Optimization.Parameters

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.number_of_params</th>
<th>integer</th>
<th>0</th>
<th>Number of parameters to optimize.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_name1</th>
<th>string</th>
<th>&quot;param_name1&quot;</th>
<th>Parameter name.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_minval1</th>
<th>double</th>
<th>0</th>
<th>Lower limit of parameter.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_maxval1</th>
<th>double</th>
<th>0</th>
<th>Upper limit of parameter.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_name2</th>
<th>string</th>
<th>&quot;param_name2&quot;</th>
<th>Parameter name.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_minval2</th>
<th>double</th>
<th>0</th>
<th>Lower limit of parameter.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_maxval2</th>
<th>double</th>
<th>0</th>
<th>Upper limit of parameter.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SolverOptions.Optimization.Parameters.param_name3</th>
<th>string</th>
<th>&quot;param_name3&quot;</th>
<th>Parameter name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Name</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval3</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval3</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name4</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval4</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval4</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name5</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval5</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval5</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name6</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval6</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval6</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name7</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval7</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval7</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name8</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval8</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval8</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name9</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval9</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval9</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_name10</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_minval10</td>
<td>double</td>
<td>0</td>
<td>Lower limit of parameter.</td>
</tr>
<tr>
<td>SolverOptions.Optimization.Parameters.param_maxval10</td>
<td>double</td>
<td>0</td>
<td>Upper limit of parameter.</td>
</tr>
</tbody>
</table>

SolverOptions.Optimization.Newton

- **random_starting_values**: bool, 0; set to 1 if the 'surviving_population_size' best value(s) of shooting with 'initial_population_size' different parameter sets should be used as starting values for Newton's method.

- **param_epsilon_abs**: double, 1e-006; Absolute value D for numerical computation of dx=K*x + D (==>f'(x) = df/dx).

- **param_epsilon_rel**: double, 0.0001; Relative value K for numerical computation of dx=K*x + D (==>f'(x) = df/dx).
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverOptions.Optimization. Newton.use_param_limits</td>
<td>integer</td>
<td>0</td>
<td>0...no limit of optimized parameter values, 1...use param_[min</td>
</tr>
<tr>
<td>SolverOptions.Optimization. Newton.initial_population_size</td>
<td>integer</td>
<td>1</td>
<td>Shooting: number of initial parameter sets initial evaluations of the cost function (randomly between range Parameters.minval and Parameters.maxval)</td>
</tr>
<tr>
<td>SolverOptions.Optimization. Newton.surviving_population_size</td>
<td>integer</td>
<td>1</td>
<td>Number of starting parameter sets for Newton’s method. This number defines how often Newton’s method should be started.</td>
</tr>
</tbody>
</table>

**SolverOptions.Optimization.Genetic**

| SolverOptions.Optimization. Genetic.initial_population_size | integer | 20 | Size of initial trial values; also used for random Newton initialization. |
| SolverOptions.Optimization. Genetic.surviving_population_size | integer | 10 | Size of values which are further tracked; also used for random Newton initialization. |
| SolverOptions.Optimization. Genetic.range_reduction_factor | double | 0.5 | Reduction of range of possible mutations. |
| SolverOptions.Optimization. Genetic.randomizer_initialization | double | 0 | Initialization of random function. |
| SolverOptions.Optimization. Genetic.min_allowed_distance_factor | double | 0.5 | Set to value greater than zero (distance is allowed radius of (hyper-)sphere in the normed parameter space (min=0)). Only the best parameter in the inner of the (hyper-)sphere is fertile. |

**SolverOptions.Sensitivity**

| SolverOptions.Sensitivity. activate | integer | 0 | (0)1...(Don’t) analyze sensitivity of sensor values with respect to parameters. |
| SolverOptions.Sensitivity. num_diff_parameter_absolute | double | 0.0001 | Absolute value D for computation of df/dx, dx=K*x+D. |
| SolverOptions.Sensitivity. num_diff_parameter_relative | double | 0.0001 | Relative factor K for computation of df/dx, dx=K*x+D. |
| SolverOptions.Sensitivity.use_final_sensor_values | bool | 0 | (0)1...(Don’t) use final sensor values. |
| SolverOptions.Sensitivity.use_optimization_parameters | bool | 0 | 1|(0) ... (Don’t) get parameters from Optimization.Parameters. |

**SolverOptions.Sensitivity.Parameters**

<p>| SolverOptions.Sensitivity. Parameters.number_of_params | integer | 0 | Number of parameters. |
| SolverOptions.Sensitivity. Parameters.param_name1 | string | &quot;&quot; | Parameter name. |
| SolverOptions.Sensitivity. Parameters.param_name2 | string | &quot;&quot; | Parameter name. |</p>
<table>
<thead>
<tr>
<th>Data object</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver.general_information</td>
<td>bool</td>
<td>0</td>
<td>Print general solver information. This includes: Newtons relative error goal, contractivity, iteration error, and qualitative information about Jacobian-updates, as well as iteration error and number of newton iterations at each post newton step, and post newton iterations at each time step.</td>
</tr>
<tr>
<td>Solver.newton_iteration_jacobi_condition</td>
<td>bool</td>
<td>0</td>
<td>Print condition number of Jacobi matrix in Newtons method whenever it is updated.</td>
</tr>
<tr>
<td>Solver.newton_iteration_jacobi_matrix</td>
<td>bool</td>
<td>0</td>
<td>Print Jacobi matrix of Newtons method whenever it is updated.</td>
</tr>
<tr>
<td>Solver.newton_iteration_residual_vector</td>
<td>bool</td>
<td>0</td>
<td>Print iterated residual vector at each Newton step.</td>
</tr>
<tr>
<td>Solver.newton_iteration_solution_vector</td>
<td>bool</td>
<td>0</td>
<td>Print iterated solution vector at each Newton step.</td>
</tr>
<tr>
<td>Solver.post_newton_iteration_data_vector</td>
<td>bool</td>
<td>0</td>
<td>Print data vector at each nonlinear iteration step.</td>
</tr>
<tr>
<td>Solver.step_solution_vector_increment</td>
<td>bool</td>
<td>0</td>
<td>Print solution increment of each step (dynamic simulation: time step, static simulation: load step).</td>
</tr>
<tr>
<td>Solver.step_solution_vector</td>
<td>bool</td>
<td>0</td>
<td>Print solution vector of each step (dynamic simulation: time step, static simulation: load step).</td>
</tr>
</tbody>
</table>

3.16.2 LoggingOptions

Data objects of LoggingOptions:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDCParser.general_information</td>
<td>bool</td>
<td>0</td>
<td>Print general information on parsed objects (e.g., while reading modeldata or configuration files).</td>
</tr>
<tr>
<td>EDCParser.output_level</td>
<td>integer</td>
<td>6</td>
<td>0.no output; 1.necessary output (Errors, start/end simulation); 2.almost necessary output (Warnings); 3.multiple simulation output (parameter variation/optimization); 4.simulation output (solver); 5.extended output (useful information); 6.complete information; 7.debug level 1; 8.debug level 2; 9.max output.</td>
</tr>
<tr>
<td>output_precision_double</td>
<td>integer</td>
<td>8</td>
<td>number of significant digits of a double in output window and logfile.</td>
</tr>
</tbody>
</table>
3.16. OPTIONS

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output_precision_vector</td>
<td>integer</td>
<td>6</td>
<td>number of significant digits of a vector in output window and logfile.</td>
</tr>
<tr>
<td>output_precision_matrix</td>
<td>integer</td>
<td>10</td>
<td>number of significant digits of a matrix in output window and logfile.</td>
</tr>
<tr>
<td>max_error_messages</td>
<td>integer</td>
<td>100</td>
<td>Number of displayed error messages.</td>
</tr>
<tr>
<td>max_warning_messages</td>
<td>integer</td>
<td>100</td>
<td>Number of displayed warning messages.</td>
</tr>
<tr>
<td>computation_output_every_x_seconds</td>
<td>double</td>
<td>2</td>
<td>Write computation output every x seconds; notice: if solver logs are printed, then this option does not take effect.</td>
</tr>
<tr>
<td>write_mass_and_stiffness_matrix</td>
<td>bool</td>
<td>0</td>
<td>Write the initial mass and stiffness matrices in Matlab format to files Mmat.dat and Kmat.dat, in Matlab directory.</td>
</tr>
<tr>
<td>default_log_filename</td>
<td>string</td>
<td>&quot;hotint.log&quot;</td>
<td>Default filename for hotint log file.</td>
</tr>
<tr>
<td>critical_log_file_size</td>
<td>double</td>
<td>10</td>
<td>critical log file size, after which a warning is displayed; in megabytes.</td>
</tr>
<tr>
<td>file_output_level</td>
<td>integer</td>
<td>7</td>
<td>0...no output; 1..necessary output (Errors, start/end simulation); 2..almost necessary output (Warnings); 3..multiple simulation output (parameter variation/optimization); 4..simulation output (solver); 5..extended output (useful information); 6..complete information; 7..debug level 1; 8..debug level 2; 9..max output.</td>
</tr>
</tbody>
</table>

3.16.3 GeneralOptions

Data objects of GeneralOptions:

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application.close_application_when Finished</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Application.show_hotint_window</td>
<td>bool</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Application.start_computation_automatically</td>
<td>bool</td>
<td>0</td>
<td>immediately start computation on program start</td>
</tr>
<tr>
<td>Application.slim_menu</td>
<td>integer</td>
<td>0</td>
<td>0..full menu, otherwise several menu items removed.</td>
</tr>
<tr>
<td>Application.reload_last_model</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Application.activate_autosave</td>
<td>bool</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Application.capture_final_frame</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Paths</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paths.application_path</td>
<td>string</td>
<td>&quot;D:\cppclean 3_2010\HotInt_V1_clean\HotIntx64\Release&quot;</td>
<td>Path of the application.</td>
</tr>
<tr>
<td>Paths.hotint_input_data_path</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Path of Hotint Input Data file.</td>
</tr>
<tr>
<td>Paths.relative_paths_relative_to_application</td>
<td>bool</td>
<td>1</td>
<td>1.. relative paths are relative to hotint.exe(application_path)</td>
</tr>
<tr>
<td>Paths.single_image_path</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Path to store single images (record frame dialog)</td>
</tr>
<tr>
<td>Paths.video_image_path</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Path to store video images series (record frame dialog)</td>
</tr>
</tbody>
</table>
### 3.16.4 ViewingOptions

**Data objects of ViewingOptions:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Animation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Animation.animate_from_beginning</td>
<td>bool</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Animation.animate_every_N_frame</td>
<td>integer</td>
<td>1</td>
<td>Animation frames: show every N’th frame at animation.</td>
</tr>
<tr>
<td>Animation.animate_deformation</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### 3.16. OPTIONS

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Animation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>animate_deformation_once</td>
<td>bool</td>
<td>0</td>
<td>1 ... animate deformation (eigenmodes) only for one cycle - for recording; 0 ... endless animate</td>
</tr>
<tr>
<td><strong>Animation.RecordSingleFrames</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>record</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don't) record frames</td>
</tr>
<tr>
<td>show_frame_numbers</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don't) show frame numbers in images;</td>
</tr>
<tr>
<td>record_every_x_frame</td>
<td>integer</td>
<td>1</td>
<td>record every x frames</td>
</tr>
<tr>
<td>single_file_name</td>
<td>string</td>
<td>&quot;snapshot&quot;</td>
<td>name of the single frame file without extensions</td>
</tr>
<tr>
<td>video_file_name</td>
<td>string</td>
<td>&quot;frame&quot;</td>
<td>name of the video frame file without extensions and number</td>
</tr>
<tr>
<td>default_image_format</td>
<td>integer</td>
<td>0</td>
<td>format of the exported file (default setting for radiobutton) 0..JPG, 1..PNG, 2..BMP</td>
</tr>
<tr>
<td>include_output_window</td>
<td>bool</td>
<td>0</td>
<td>includes the output window to the screenshot</td>
</tr>
<tr>
<td>max_one_frame_per_timestep</td>
<td>bool</td>
<td>1</td>
<td>prevent multiple frames of the same time step (de-activate for saving e.g. eigenmodes)</td>
</tr>
<tr>
<td><strong>Misc</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>redraw_frequency</td>
<td>integer</td>
<td>4</td>
<td>Redraw frequency: 0..off, 1..draw last frame, 2..100sec, 3..20sec, 4..2sec, 5..200ms, 6..50ms, 7..20ms, 8..every 10 frames, 9..every frame</td>
</tr>
<tr>
<td>global_line_thickness</td>
<td>double</td>
<td>1</td>
<td>Global_line_thickness (coord system, etc.) ****.</td>
</tr>
<tr>
<td>global_point_size</td>
<td>double</td>
<td>2</td>
<td>Global point size (coord system, grid, etc.) ****.</td>
</tr>
<tr>
<td>show_3D_text_in_front</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don't) show 3D texts in front</td>
</tr>
<tr>
<td>axes_position</td>
<td>integer</td>
<td>0</td>
<td>position of axes: bottom left (0), bottom right (1), top right (2), top left (3), center (4), no axes (5)</td>
</tr>
<tr>
<td>lock_rotation</td>
<td>bool</td>
<td>0</td>
<td>lock rotation of model (for 2D models)</td>
</tr>
<tr>
<td><strong>GeomElements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>line_thickness</td>
<td>double</td>
<td>2</td>
<td>GeoElement (outline) line thickness ****.</td>
</tr>
<tr>
<td><strong>Origin</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>show</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don't) draw coordinate system in origin (X0, Y0, Z0).</td>
</tr>
<tr>
<td>size_of_origin</td>
<td>double</td>
<td>0.5</td>
<td>Size of origin.</td>
</tr>
<tr>
<td><strong>Grid</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>show</td>
<td>integer</td>
<td>0</td>
<td>Show Grid and Background planes (add up), 1=XY, 2=XZ, 4=YZ.</td>
</tr>
<tr>
<td>pos_x</td>
<td>double</td>
<td>0</td>
<td>X-position for intersection point of planes</td>
</tr>
<tr>
<td>pos_y</td>
<td>double</td>
<td>0</td>
<td>Y-position for intersection point of planes</td>
</tr>
<tr>
<td>pos_z</td>
<td>double</td>
<td>0</td>
<td>Z-position for intersection point of planes</td>
</tr>
<tr>
<td>size_1</td>
<td>double</td>
<td>2</td>
<td>X-size of background plane</td>
</tr>
<tr>
<td>size_2</td>
<td>double</td>
<td>2</td>
<td>Y-size of background plane</td>
</tr>
<tr>
<td>size_3</td>
<td>double</td>
<td>2</td>
<td>Z-size of background plane</td>
</tr>
<tr>
<td>step_1</td>
<td>double</td>
<td>0.1</td>
<td>Grid discretization X-direction</td>
</tr>
<tr>
<td>step_2</td>
<td>double</td>
<td>0.1</td>
<td>Grid discretization Y-direction</td>
</tr>
<tr>
<td>step_3</td>
<td>double</td>
<td>0.1</td>
<td>Grid discretization Z-direction</td>
</tr>
<tr>
<td><strong>Grid.Colors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transparency_factor</td>
<td>double</td>
<td>0.1</td>
<td>Transparency factor for the background planes</td>
</tr>
<tr>
<td>plane1_col_r</td>
<td>double</td>
<td>0.85</td>
<td>Red color channel for XY plane</td>
</tr>
<tr>
<td>plane1_col_g</td>
<td>double</td>
<td>0.85</td>
<td>Green color channel for XY plane</td>
</tr>
<tr>
<td>plane1_col_b</td>
<td>double</td>
<td>0.85</td>
<td>Blue color channel for XY plane</td>
</tr>
<tr>
<td>plane2_col_r</td>
<td>double</td>
<td>0.95</td>
<td>Red color channel for XZ plane</td>
</tr>
<tr>
<td>plane2_col_g</td>
<td>double</td>
<td>0.95</td>
<td>Green color channel for XZ plane</td>
</tr>
<tr>
<td>plane2_col_b</td>
<td>double</td>
<td>0.95</td>
<td>Blue color channel for XZ plane</td>
</tr>
<tr>
<td>plane3_col_r</td>
<td>double</td>
<td>0.95</td>
<td>Red color channel for YZ plane</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Type</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Grid.Colors.plane3.col_g</td>
<td>double</td>
<td>0.95</td>
<td>Green color channel for YZ plane</td>
</tr>
<tr>
<td>Grid.Colors.plane3.col_b</td>
<td>double</td>
<td>0.95</td>
<td>Blue color channel for YZ plane</td>
</tr>
<tr>
<td>CuttingPlane.1.activate</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... Use (Don’t use) cutting plane.</td>
</tr>
<tr>
<td>CuttingPlane.1.normal_X</td>
<td>double</td>
<td>1</td>
<td>Cutting plane normal-X</td>
</tr>
<tr>
<td>CuttingPlane.1.normal_Y</td>
<td>double</td>
<td>0</td>
<td>Cutting plane normal-Y</td>
</tr>
<tr>
<td>CuttingPlane.1.normal_Z</td>
<td>double</td>
<td>0</td>
<td>Cutting plane normal-Z</td>
</tr>
<tr>
<td>CuttingPlane.1.distance</td>
<td>double</td>
<td>0</td>
<td>Cutting plane distance</td>
</tr>
<tr>
<td>CuttingPlane.2.activate</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... Use (Don’t use) cutting plane.</td>
</tr>
<tr>
<td>CuttingPlane.2.normal_X</td>
<td>double</td>
<td>1</td>
<td>Cutting plane 2 normal-X</td>
</tr>
<tr>
<td>CuttingPlane.2.normal_Y</td>
<td>double</td>
<td>0</td>
<td>Cutting plane 2 normal-Y</td>
</tr>
<tr>
<td>CuttingPlane.2.normal_Z</td>
<td>double</td>
<td>0</td>
<td>Cutting plane 2 normal-Z</td>
</tr>
<tr>
<td>CuttingPlane.2.distance</td>
<td>double</td>
<td>0</td>
<td>Cutting plane 2 distance</td>
</tr>
<tr>
<td>CuttingPlane.cut_bodies</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) cut bodies.</td>
</tr>
<tr>
<td>CuttingPlane.cut_bodies_alshapes</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) cut alternative shapes of bodies.</td>
</tr>
<tr>
<td>CuttingPlane.cut_ground</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) cut background.</td>
</tr>
<tr>
<td>CuttingPlane.cut_whole_scene_by_open_gl</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don’t) use OpenGL for handling cutting planes.</td>
</tr>
<tr>
<td>StandardView.angle_rot_axis_1</td>
<td>integer</td>
<td>1</td>
<td>Rotation axis for standard view angle_1 (rotation axis 1, 2 or 3).</td>
</tr>
<tr>
<td>StandardView.angle_rot_axis_2</td>
<td>integer</td>
<td>2</td>
<td>Rotation axis for standard view angle_2 (rotation axis 1, 2 or 3).</td>
</tr>
<tr>
<td>StandardView.angle_rot_axis_3</td>
<td>integer</td>
<td>3</td>
<td>Rotation axis for standard view angle_3 (rotation axis 1, 2 or 3).</td>
</tr>
<tr>
<td>StandardView.angle_1</td>
<td>double</td>
<td>0</td>
<td>Standard view angle_1</td>
</tr>
<tr>
<td>StandardView.angle_2</td>
<td>double</td>
<td>0</td>
<td>Standard view angle_2</td>
</tr>
<tr>
<td>StandardView.angle_3</td>
<td>double</td>
<td>0</td>
<td>Standard view angle_3</td>
</tr>
<tr>
<td>Bodies.Rigid.show_outline</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) show bodies outline.</td>
</tr>
<tr>
<td>Bodies.Rigid.show_faces</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) show bodies faces.</td>
</tr>
<tr>
<td>Bodies.Rigid.line_thickness</td>
<td>double</td>
<td>1</td>
<td>Rigid body (outline) line thickness **** not used yet.</td>
</tr>
<tr>
<td>Bodies.Rigid.draw_center_of_gravity</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) draw center of gravity</td>
</tr>
<tr>
<td>Bodies.Rigid.draw_resolution</td>
<td>integer</td>
<td>12</td>
<td>Draw resolution for Rigid3D</td>
</tr>
<tr>
<td>Bodies.Rigid.COG_sizefactor</td>
<td>double</td>
<td>1</td>
<td>Cog_factor for Rigid3D (default: 1).</td>
</tr>
<tr>
<td>Bodies.show_element_numbers</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don’t) show element body numbers.</td>
</tr>
<tr>
<td>Bodies.show_local_frame</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don’t) show local body frame.</td>
</tr>
<tr>
<td>Bodies.transparent</td>
<td>bool</td>
<td>1</td>
<td>1(0) ... (Don’t) draw bodies transparent.</td>
</tr>
<tr>
<td>Bodies.local_frame_size</td>
<td>double</td>
<td>0</td>
<td>Body local frame size</td>
</tr>
<tr>
<td>Bodies.deformation_scale_factor</td>
<td>double</td>
<td>1</td>
<td>Deformation scale factor.</td>
</tr>
<tr>
<td>Bodies.scale_rigid_body_displacements</td>
<td>integer</td>
<td>0</td>
<td>1(0) ... (Don’t) use deformation scale factor in animation.</td>
</tr>
<tr>
<td>Bodies.show_velocity_vector</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don’t) show velocity vector, e.g. for particles.</td>
</tr>
<tr>
<td>Bodies.velocity_vector</td>
<td>bool</td>
<td>0</td>
<td>1(0) ... (Don’t) show velocity vector for particles only.</td>
</tr>
<tr>
<td>Bodies.velocity_vector_scaling_mode</td>
<td>integer</td>
<td>1</td>
<td>1: constant scaling (a), 2: linear scaling (ax), 3: exponential scaling (a(1-exp(-x/b))).</td>
</tr>
</tbody>
</table>
### Options

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bodies. velocity_vector_scaling_a</td>
<td>double</td>
<td>1</td>
<td>Magnification factor; e.g., if velocity_vector_scaling_mode == 2: velocity vector length = ( v \times \text{velocity_vector_scaling}_a ).</td>
</tr>
<tr>
<td>Bodies. velocity_vector_scaling_b</td>
<td>double</td>
<td>1</td>
<td>Knee factor; e.g., if velocity_vector_scaling_mode == 3: velocity vector length = ( \text{velocity_vector_scaling}_a \times (1 - \exp(-v / \text{velocity_vector_scaling}_b)) ).</td>
</tr>
<tr>
<td>Bodies. velocity_vector_scaling_thickness</td>
<td>double</td>
<td>1</td>
<td>Thickness scaling factor; independent from mode.</td>
</tr>
<tr>
<td>Bodies.Particles. displacement_scale_factor</td>
<td>double</td>
<td>1</td>
<td>Factor for scaling the displacements.</td>
</tr>
<tr>
<td>Bodies.Particles. draw_size_factor</td>
<td>double</td>
<td>1</td>
<td>Factor for adjusting the size of particles while drawing.</td>
</tr>
<tr>
<td>Bodies.Particles. draw_every_nth</td>
<td>integer</td>
<td>1</td>
<td>Draw every n-th particle only.</td>
</tr>
</tbody>
</table>

### Finite Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiniteElements.Contour. activate</td>
<td>bool</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. max_stress_active</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. min_stress_active</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. min_stress</td>
<td>double</td>
<td>0</td>
<td>Value of min. stress.</td>
</tr>
<tr>
<td>FiniteElements.Contour. post_processing_variable_name</td>
<td>string</td>
<td>&quot;&quot;</td>
<td>Name of the field variable, which is currently selected for contour plotting.</td>
</tr>
<tr>
<td>FiniteElements.Contour. variable_range_auto_update</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. color_tiling</td>
<td>integer</td>
<td>10</td>
<td>Color tiling (used for FE-color texture).</td>
</tr>
<tr>
<td>FiniteElements.Contour. label_precision</td>
<td>integer</td>
<td>3</td>
<td>Number of digits for the numbers in label.</td>
</tr>
<tr>
<td>FiniteElements.Contour. plot_interpolated</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. grey_mode</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. invert_colors</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. nonlinear_color_legend</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. hide_legend</td>
<td>bool</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FiniteElements.Contour. axis_tiling</td>
<td>integer</td>
<td>16</td>
<td>Axis tiling (for element face and outline, beams and plates).</td>
</tr>
<tr>
<td>FiniteElements.Contour. resolution_axis</td>
<td>integer</td>
<td>8</td>
<td>Axis resolution: contour plot resolution along axis, beams and plates.</td>
</tr>
<tr>
<td>FiniteElements.Contour. resolution_cross_section</td>
<td>integer</td>
<td>4</td>
<td>Cross-section resolution: contour plot resolution at cross-section, beams and plates.</td>
</tr>
<tr>
<td>FiniteElements.Contour. resolution_solid_elements</td>
<td>integer</td>
<td>2</td>
<td>Contour plot resolution for solid finite elements.</td>
</tr>
</tbody>
</table>

### Finite Elements Nodes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiniteElements.Nodes. show</td>
<td>bool</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>FiniteElements.Nodes.</code> show_node_numbers</td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) show node numbers.</td>
</tr>
<tr>
<td><code>FiniteElements.Nodes.</code> node_resolution</td>
<td>integer</td>
<td>3</td>
<td>Node resolution for drawing.</td>
</tr>
<tr>
<td><code>FiniteElements.Nodes.</code> node_size</td>
<td>double</td>
<td>0.001</td>
<td>Draw node size.</td>
</tr>
<tr>
<td><code>FiniteElements.Nodes.</code> show_velocity_vector</td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) draw vector in direction of velocity.</td>
</tr>
<tr>
<td><code>FiniteElements.Mesh.</code> show</td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) show mesh of finite element.</td>
</tr>
<tr>
<td><code>FiniteElements.Mesh.</code> draw_flat_elements</td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) draw Plate elements flat, only mid-plane (view from top only).</td>
</tr>
<tr>
<td><code>FiniteElements.Mesh.</code> draw_only_surface_elements</td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) draw surface elements only.</td>
</tr>
<tr>
<td><code>FiniteElements.Mesh.</code> element_line_thickness</td>
<td>double</td>
<td>1</td>
<td>Finite element line thickness (outline of 2D and 3D beam, plate).</td>
</tr>
<tr>
<td><code>FiniteElements.Mesh.</code> shrinking_factor</td>
<td>double</td>
<td>1</td>
<td>Shrinking factor.</td>
</tr>
<tr>
<td><code>Connectors.show_constraints</code></td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) show joints/connectors.</td>
</tr>
<tr>
<td><code>Connectors.show_control_elements</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) draw control elements in 3D Window.</td>
</tr>
<tr>
<td><code>Connectors.show_constraint_numbers</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) show constraint number.</td>
</tr>
<tr>
<td><code>Connectors.show_faces</code></td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) show constraint faces -- show constraint faces.</td>
</tr>
<tr>
<td><code>Connectors.transparent</code></td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) draw constraints transparent.</td>
</tr>
<tr>
<td><code>Connectors.draw_outline</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) draw constraints outline **** --&gt; faces is IOption 114.not used yet.</td>
</tr>
<tr>
<td><code>Connectors.line_thickness</code></td>
<td>double</td>
<td>1</td>
<td>Constraint (outline) line thickness ****.not used yet.</td>
</tr>
<tr>
<td><code>Connectors&gt;Contact.</code> show_contact_as_circle</td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) draw circles at contact of bodies.</td>
</tr>
<tr>
<td><code>Connectors&gt;Contact.</code> show_contact_points</td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) show contact points.</td>
</tr>
<tr>
<td><code>Connectors.global_draw_scalar_size</code></td>
<td>double</td>
<td>0.1</td>
<td>global scalar constraint draw size (e.g. radius)</td>
</tr>
<tr>
<td><code>Connectors.global_draw_resolution</code></td>
<td>double</td>
<td>16</td>
<td>global constraint draw resolution</td>
</tr>
<tr>
<td><code>ConnectorsAutoSize</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] Autogenerate a global scalar constraint draw size.</td>
</tr>
<tr>
<td><code>Loads.show_loads</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) show loads.</td>
</tr>
<tr>
<td><code>Loads.arrow_size</code></td>
<td>double</td>
<td>0.1</td>
<td>Size of arrow for drawing of loads.</td>
</tr>
<tr>
<td><code>Loads.color_red</code></td>
<td>double</td>
<td>0.6</td>
<td>Red-value for drawing of loads (use values between 0. and 1.).</td>
</tr>
<tr>
<td><code>Loads.color_green</code></td>
<td>double</td>
<td>0.6</td>
<td>Green-value for drawing of loads (use values between 0. and 1.).</td>
</tr>
<tr>
<td><code>Loads.color_blue</code></td>
<td>double</td>
<td>0</td>
<td>Blue-value for drawing of loads (use values between 0. and 1.).</td>
</tr>
<tr>
<td><code>Sensors.show_sensors</code></td>
<td>bool</td>
<td>0</td>
<td>1[0] ... (Don’t) show sensors.</td>
</tr>
<tr>
<td><code>Sensors.transparent</code></td>
<td>bool</td>
<td>1</td>
<td>1[0] ... (Don’t) draw sensors transparent.</td>
</tr>
<tr>
<td><code>Sensors.sensor_origin_size</code></td>
<td>double</td>
<td>0.2</td>
<td>Sensor origin size.</td>
</tr>
<tr>
<td><code>OpenGL.enable_lighting</code></td>
<td>bool</td>
<td>1</td>
<td>OpenGL lighting.</td>
</tr>
</tbody>
</table>
### 3.16. OPTIONS

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><strong>Type</strong></th>
<th><strong>Value</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL.smooth_model</td>
<td>bool</td>
<td>1</td>
<td>OpenGL SMOOTH ShadeModel smooth.</td>
</tr>
<tr>
<td>OpenGL.immediate_apply_dialog</td>
<td>bool</td>
<td>1</td>
<td>Immediate apply in OpenGL dialog.</td>
</tr>
<tr>
<td>OpenGL.global_culling</td>
<td>integer</td>
<td>0</td>
<td>OpenGL cull (means: exclude) 1=front 2=back or 3=both views on faces of polygons; 0=don’t cull any view.</td>
</tr>
<tr>
<td>OpenGL.global_transparency</td>
<td>double</td>
<td>0.8</td>
<td>Global transparency for SetColor, 1=no translucency, 0=fully transparent.</td>
</tr>
<tr>
<td>OpenGL.material_shininess</td>
<td>double</td>
<td>60</td>
<td>Material shininess (0.128).</td>
</tr>
<tr>
<td>OpenGL.material_color_intensity</td>
<td>double</td>
<td>1</td>
<td>Material specular color intensity.</td>
</tr>
<tr>
<td>OpenGL.Light1.enable</td>
<td>bool</td>
<td>1</td>
<td>OpenGL enable light1.</td>
</tr>
<tr>
<td>OpenGL.Light1.use_light_position</td>
<td>bool</td>
<td>0</td>
<td>OpenGL light1 mode (0=standard, 1=use light position).</td>
</tr>
<tr>
<td>OpenGL.Light1.ambient</td>
<td>double</td>
<td>0.25</td>
<td>Light1 ambient parameter.</td>
</tr>
<tr>
<td>OpenGL.Light1.diffuse</td>
<td>double</td>
<td>0.4</td>
<td>Light1 diffuse parameter.</td>
</tr>
<tr>
<td>OpenGL.Light1.specular</td>
<td>double</td>
<td>0.4</td>
<td>Light1 specular parameter.</td>
</tr>
<tr>
<td>OpenGL.Light1.pos_x</td>
<td>double</td>
<td>1</td>
<td>Light1 posx.</td>
</tr>
<tr>
<td>OpenGL.Light1.pos_y</td>
<td>double</td>
<td>1</td>
<td>Light1 posy.</td>
</tr>
<tr>
<td>OpenGL.Light1.pos_z</td>
<td>double</td>
<td>-1</td>
<td>Light1 posz.</td>
</tr>
<tr>
<td>OpenGL.Light2.enable</td>
<td>bool</td>
<td>1</td>
<td>OpenGL enable light2.</td>
</tr>
<tr>
<td>OpenGL.Light2.use_light_position</td>
<td>bool</td>
<td>0</td>
<td>OpenGL light2 mode (0=standard, 1=use light position).</td>
</tr>
<tr>
<td>OpenGL.Light2.ambient</td>
<td>double</td>
<td>0.25</td>
<td>Light2 ambient parameter.</td>
</tr>
<tr>
<td>OpenGL.Light2.diffuse</td>
<td>double</td>
<td>0.4</td>
<td>Light2 diffuse parameter.</td>
</tr>
<tr>
<td>OpenGL.Light2.specular</td>
<td>double</td>
<td>0</td>
<td>Light2 specular parameter.</td>
</tr>
<tr>
<td>OpenGL.Light2.pos_x</td>
<td>double</td>
<td>0</td>
<td>Light2 posx.</td>
</tr>
<tr>
<td>OpenGL.Light2.pos_y</td>
<td>double</td>
<td>3</td>
<td>Light2 posy.</td>
</tr>
<tr>
<td>OpenGL.Light2.pos_z</td>
<td>double</td>
<td>2</td>
<td>Light2 posz.</td>
</tr>
<tr>
<td>ApplicationWindow.rect_left</td>
<td>integer</td>
<td>250</td>
<td>left coordinate of application window</td>
</tr>
<tr>
<td>ApplicationWindow.rect_top</td>
<td>integer</td>
<td>50</td>
<td>top coordinate of application window</td>
</tr>
<tr>
<td>ApplicationWindow.rect_width</td>
<td>integer</td>
<td>700</td>
<td>width of application window</td>
</tr>
<tr>
<td>ApplicationWindow.rect_height</td>
<td>integer</td>
<td>700</td>
<td>left coordinate of application window</td>
</tr>
<tr>
<td>DataManager.dialog_open</td>
<td>bool</td>
<td>1</td>
<td>open data manager on startup</td>
</tr>
<tr>
<td>DataManager.store_data_to_files</td>
<td>bool</td>
<td>0</td>
<td>if checked, then solution data (for data manager) is stored in files, instead of memory; these files are located in subdirectory solution_data of GeneralOptions.Paths.sensor_output_path.</td>
</tr>
<tr>
<td>DataManager.store_data_every</td>
<td>double</td>
<td>0.01</td>
<td>Store data with data-manager, redraw and create animations: # -4 == once at endtime, -2 == always, -1 == at max stepsize, 0 = never, x.x = at every time x.x.</td>
</tr>
<tr>
<td>DataManager.special_output</td>
<td>integer</td>
<td>0</td>
<td>Store a single special output file, available: 1 I-DEAS Format: stresses at element nodes, 2 VTK Format, 3 I-DEAS and VTK format</td>
</tr>
<tr>
<td>OutputWindow.dialog_open</td>
<td>bool</td>
<td>1</td>
<td>open output dialog on startup</td>
</tr>
<tr>
<td>OutputWindow.stored_width</td>
<td>integer</td>
<td>200</td>
<td>stored width of output dialog</td>
</tr>
<tr>
<td>OutputWindow.enable_output_text</td>
<td>bool</td>
<td>1</td>
<td>enable output text in output dialog</td>
</tr>
<tr>
<td>View3D.Centre_point</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 3.16.5 PlotToolOptions

**Data objects of PlotToolOptions:**

<table>
<thead>
<tr>
<th>Data name</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_redraw</td>
<td>bool</td>
<td>0</td>
<td>1/0 (Don't) redraw in regular intervals</td>
</tr>
<tr>
<td>auto_redraw_interval</td>
<td>double</td>
<td>30</td>
<td>redraw every x seconds</td>
</tr>
<tr>
<td>auto_rescale</td>
<td>bool</td>
<td>1</td>
<td>1/0 (Don't) rescale to fully fit the whole data when updated</td>
</tr>
<tr>
<td>title_size_factor</td>
<td>double</td>
<td>1.25</td>
<td>factor to in-/decrease font size of title in respect to axis font</td>
</tr>
<tr>
<td>ticks_size_factor</td>
<td>double</td>
<td>0.7</td>
<td>factor to in-/decrease font size of ticks in respect to axis font</td>
</tr>
<tr>
<td>line_thickness_border</td>
<td>integer</td>
<td>4</td>
<td>line thickness (border) in logical points</td>
</tr>
<tr>
<td>line_thickness_factor</td>
<td>double</td>
<td>1</td>
<td>scaling factor for all plotted lines</td>
</tr>
<tr>
<td>status_bar_info</td>
<td>bool</td>
<td>0</td>
<td>1/0 (Don't) show status bar information.</td>
</tr>
<tr>
<td>open_automatically</td>
<td>bool</td>
<td>0</td>
<td>1/0 (Don't) open PlotTool when HOTINT opens</td>
</tr>
</tbody>
</table>
### 3.16. OPTIONS

<table>
<thead>
<tr>
<th>layout_file</th>
<th>string</th>
<th>m</th>
<th>layout file that is loaded when PlotTool is automatically loaded</th>
</tr>
</thead>
</table>

**DataPoints**

| DataPoints.flag_draw_every_nth | bool | 0 | 1(0) .. (Don't) skip several datapoints in draw routine |
| DataPoints.draw_every_nth     | integer | 100 | draw every nth datapoint (0 for every) |
| DataPoints.flag_mark_every_nth | bool | 0 | 1(0) .. (Don't) skip marking of several datapoints in draw routine |
| DataPoints.mark_every_nth     | integer | 100 | mark every nth datapoint (0 for every) |
| DataPoints.vertical_marker    | bool | 0 | 1(0) .. (Don't) mark current time in plot with a special marker |
| DataPoints.draw_only_to_time  | bool | 0 | 1(0) .. (Don't) draw the data only up to the time from datamanager |
| DataPoints.use_time_interval  | bool | 0 | 1(0) .. (Don't) use t_min and t_max as boundaries for drawing in plottool |
| DataPoints.t_min              | double | 0 | Lower boundary for time interval plot. Only used if use_time_interval = 1. |
| DataPoints.t_max              | double | 0 | Upper boundary for time interval plot. Only used if use_time_interval = 1. |

**View**

| View.initial_size_horizontal | integer | 640 | initial size of the CView holding the plot |
| View.initial_size_vertical   | integer | 480 | initial size of the CView holding the plot |
| View.plot_horizontal         | integer | 3000 | size in logical units for the plot - fixed aspect ratio |
| View.plot_vertical           | integer | 2000 | size in logical units for the plot - fixed aspect ratio |
| View.distance_left           | double | 15 | surplus in %plotwidth from left border of the plot to left border of the window |
| View.distance_top            | double | 15 | surplus in %plotheight from upper border of the plot to upper border of the window |
| View.distance_bottom         | double | 20 | surplus in %plotheight from lower border of the plot to lower border of the window |
| View.distance_right          | double | 15 | surplus in %plotheight from lower border of the plot to lower border of the window |

**Watches**

| Watches.initial_size_horizontal | integer | 300 | initial size of the CView holding the plot |
| Watches.initial_size_vertical   | integer | 200 | initial size of the CView holding the plot |

**Axis**

| Axis.draw_at_origin            | bool | 0 | 1(0) .. (Don't) draw axis at origin |
| Axis.label_major               | bool | 1 | 1(0) .. (Don't) write labels for major ticks |
| Axis.label_minor               | bool | 1 | 1(0) .. (Don't) write labels for minor ticks |
| Axisoverridesize              | double | 3 | percentage the axes are longer than the graph |
| Axis.ticksizesize             | double | 2 | size in percent of major ticks, minor are half size |
| Axis.minor_ticks_x            | integer | 0 | minor ticks for x-axis |
| Axis.minor_ticks_y            | integer | 0 | minor ticks for y-axis |
| Axis.digits_x_labels          | integer | 3 | maximum digits for x-axis labels |
| Axis.digits_y_labels          | integer | 3 | maximum digits for y-axis labels |

**Grid**

| Grid.shading                  | double | 0.5 | Linecolor of grid lines: black if 0, white if 1, and grey scales in between |
| Grid.linetype_major_x         | integer | 2 | Linetype for major gridlines, x axis (0 = no line, 1 = solid, 2 = dash, 3 = dot) |
| Grid.linetype_minor_x         | integer | 3 | Linetype for minor gridlines, x axis (0 = no line, 1 = solid, 2 = dash, 3 = dot) |
| Grid.linetype_major_y         | integer | 2 | Linetype for major gridlines, y axis (0 = no line, 1 = solid, 2 = dash, 3 = dot) |
| Grid.linetype_minor_y         | integer | 3 | Linetype for minor gridlines, y axis (0 = no line, 1 = solid, 2 = dash, 3 = dot) |

**Legend**
| Legend.show   | bool  | 0 | 1[0] .. (Don’t) draw axis at origin |
| Legend.left  | double | 75 | position in % of the legend’s left border |
| Legend.right | double | 100 | position in % of the legend’s right border |
| Legend.top   | double | 100 | position in % of the legend’s upper border |
| Legend.bottom| double | 75 | position in % of the legend’s lower border |

### SavePicture

| SavePicture.filename   | string  | "snap" | filename for the picture without extensions |
| SavePicture.size_horizontal | integer | 1600 | size in pixels of the saved BMP |
| SavePicture.size_vertical   | integer | 1200 | size in pixels of the saved BMP |
| SavePicture.jpg_quality   | integer | 10 | quality setting for the JPG encoder |
| SavePicture.store_jpg   | bool  | 1 | 1[0] .. (Don’t) store image as jpg |
| SavePicture.store_png   | bool  | 0 | 1[0] .. (Don’t) store image as png |
| SavePicture.store_bmp   | bool  | 0 | 1[0] .. (Don’t) store image as bmp |
| SavePicture.store_emf   | bool  | 1 | 1[0] .. (Don’t) store image as emf |
Bibliography


