XML GUIDE
FOR DUALSPHYSICS

COUPLING WITH PROJECT CHRONO
SPECIAL: CHRONO

July 2020

DualSPHysics team

http://dual.sphysics.org

DualSPHysics has been coupled with Project Chrono: https://projectchrono.org/

Source code of Project Chrono can be found at https://github.com/projectchrono/chrono

Only some of the functionalities and modules are available in this coupling:

COLLISIONS:
- non-smooth contacts (NSC)
- smooth contacts (SMC)

RESTRICTIONS:
- linear spring with stiffness and damping
- spring using Coulomb damping
- hinge along an axis
- spherical hinge on a point
- sliding along an axis
- pulley

XML file: Special-Chrono

```
<special>
    <chrono>
        <savedata value="0.01" />
        <schemescale value="1" />
        <collision activate="true">
        <bodyfixed id="domain" mkbound="0" modelfile="domain.obj" />
        <bodyfloating id="box1" mkbound="1" modelfile="box1.obj" />
        <bodyfloating id="box2" mkbound="2" modelfile="box2.obj" />
        <link linearspring idbody1="domain" idbody2="box1">
        <link coulombdamping  idbody1="domain" idbody2="box1">
        <link hinge idbody1="box1" idbody2="box2">
        <link spheric idbody1="box1" idbody2="box2">
        <link pointline idbody1="box1">
        <link spheric idbody1="box2">
        <link pulley idbody1="box1" idbody2="box2">
    </chrono>
</special>
```

savedata: Saves CSV with data exchange between DualSPHysics and Project Chrono
ChronoBody_forces.csv, ChronoExchange_mkbound_1.csv, ChronoExchange_mkbound_2.csv
schemescale: Creates VTK file with the initial scheme of Chrono objects using the given scale
CfgChrono_Scheme.vtk

collision: section to activate collisions
bodyfixed and bodyfloating: indicates the objets to be linked or to collide with each other
link_xxxx: section to define mechanical restrictions

```
<parameters>
    <parameter key="RigidAlgorithm" value="3" />
</parameters>
```

RigidAlgorithm: 1:SPH or 2:DEM or 3:Chrono
THIS IS IMPORTANT **TO ACTIVATE CRHONO**

XML file: Special-Chrono

Collisions

<u>In order to solve collisions with Project Chrono we need:</u>

1) To activate Chrono in "parameters" using RigidAlgorithm:3

```
<parameters>
    <parameter key="RigidAlgorithm" value="3" />
</parameters>
```

2) To define "properties" for the objects that are going to collide with each other

- Young modulus
- Poisson ration
- Restitution coefficient
- Kinetic Friction coefficient

```
<floatings>
    <floating mkbound="51" property="pvc" >
        <massbody value="2.7" />
    </floating>
</floatings>
```

3) To include the "collision" section in "special-chrono"

- To define a minimum distance of detection to solve collisions
- To choose the contact method type: non-smooth contacts or smooth contacts

```
<collision activate="true">
    <distancedp value="0.5" />
    <contactmethod value="0" />
</collision>
```

4) To create the geometries of the objects

geo.obj is created (in Case_out/chrono_objs) starting from VTK, PLY or STL files

normal vector of the faces are also computed

```
<bodyfloating id="cube" mkbound="51" modelfile="AutoActual" />
<bodyfixed id="tank" mkbound="0" modelfile="AutoActual" modelnormal="invert" />
```

XML file: Special-Chrono

Collisions

<u>In order to solve collisions with Project Chrono we need:</u>

The contact method type: non-smooth contacts or smooth contacts

```
<collision activate="true">
    <distancedp value="0.5" />
    <contactmethod value="0" />
</collision>
```

```
<contactmethod value="0" comment="Contact method type.
                              0:NSC (Non Smooth Contacts),
                              1:SMC (SMooth Contacts) (default=0)" />
```

0: Non-smooth contacts use these properties:
- Restitution coefficient
- Kinetic Friction coefficient

1: Smooth contacts use these properties :
- Restitution coefficient
- Kinetic Friction coefficient
- Poisson ration
- Young modulus

XML file: Special-Chrono

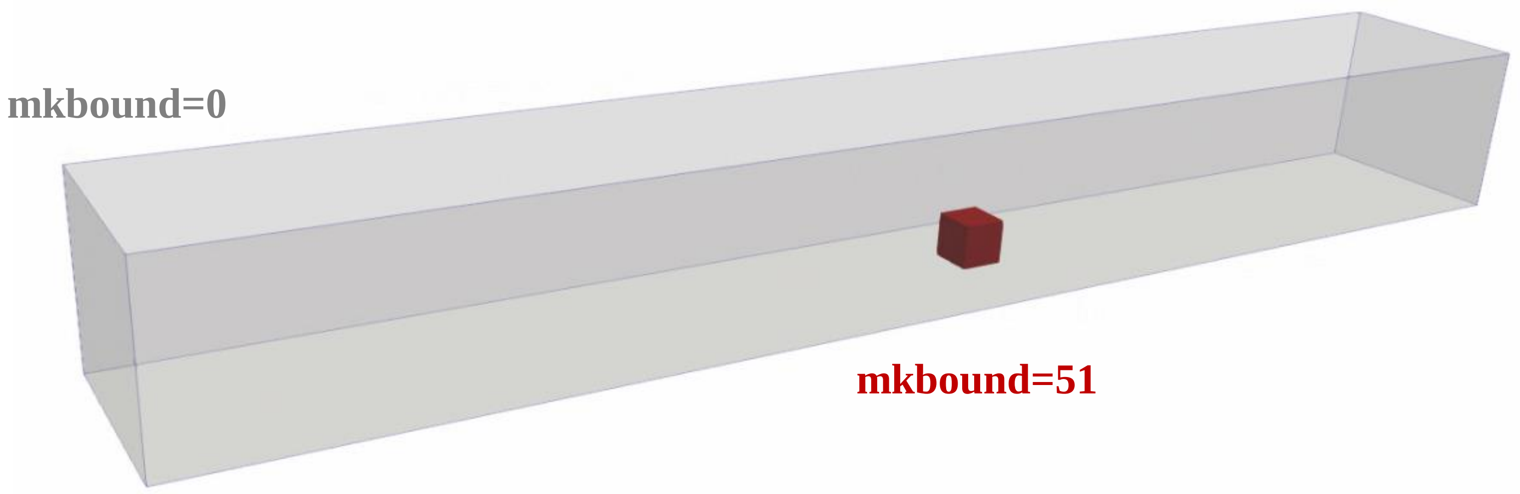
Collisions

mkbound=51 is a floating object made of “pvc”
mkbound=0 is a fixed tank made of “steel”

```xml
<floatings>
    <floating mkbound="51" property="pvc" >
        <massbody value="2.7" />
    </floating>
</floatings>

<properties>
    <propertyfile file="Floating_Materials.xml" path="materials" />
    <links>
        <link mkbound="0" property="steel" />
    </links>
</properties>
```

```xml
<property name="steel">
    <Young_Modulus value="210000000000.0" comment="Young Modulus (N/m2)" />
    <PoissonRatio value="0.35" comment="Poisson Ratio (-)" />
    <Restitution_Coefficient value="0.80" comment="Restitution Coefficient (-)" />
    <Kfric value="0.35" comment="Kinetic friction coefficient" />
</property>
<property name="pvc">
    <Young_Modulus value="3000000000.0" comment="Young Modulus (N/m2)" />
    <PoissonRatio value="0.30" comment="Poisson Ratio (-)" />
    <Restitution_Coefficient value="0.60" comment="Restitution Coefficient (-)" />
    <Kfric value="0.15" comment="Kinetic friction coefficient" />
</property>
```



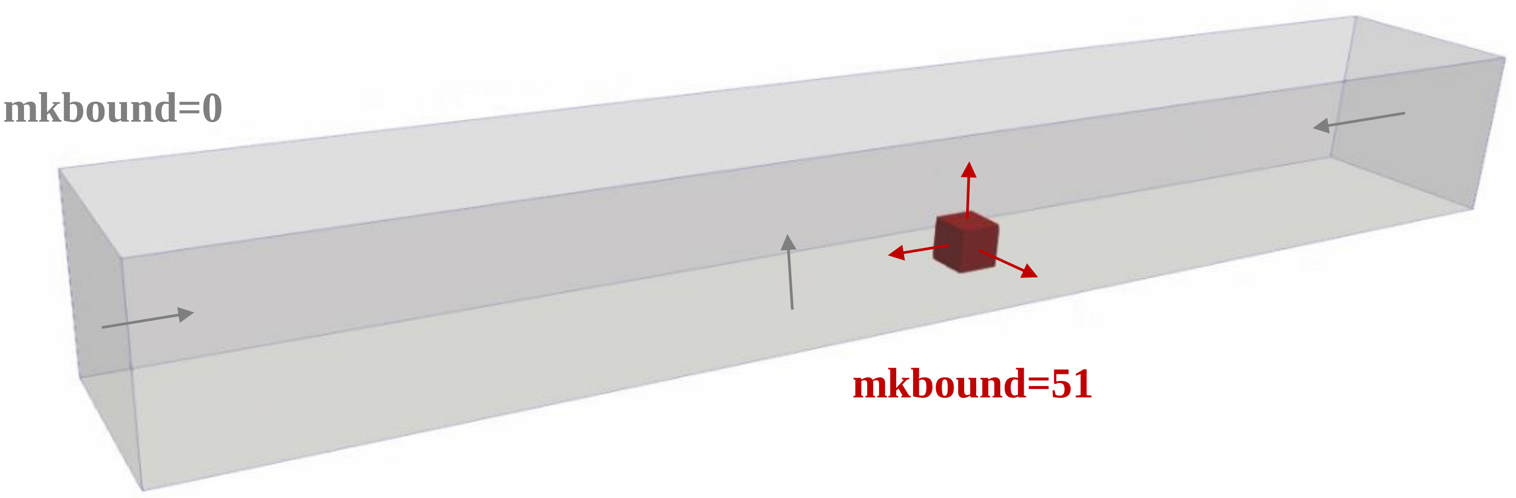
XML file: Special-Chrono

Collisions

activate:true To activate the functionality to solve collisions
distancedp:0.5 Allowed collision overlap according to “dp” (0.5*dp)
contactmethod:0 Contact method type. 0:NSC (Non Smooth Contacts), 1:SMC (SMooth Contacts)

```
<special>
    <chrono>
        <savedata value="0.01" />
        <schemescale value="1" />
        <collision activate="true">
            <distancedp value="0.5" />
            <contactmethod value="0" />
        </collision>
        <bodyfloating id="cube" mkbound="51" modelfile="AutoActual" />
        <bodyfixed id="tank" mkbound="0" modelfile="AutoActual" modelnormal="invert" />
    </chrono>
</special>
```

Creates the objects in **Case_out/chrono_objs**:
cube_mkb0051.obj
tank_mkb0000.obj

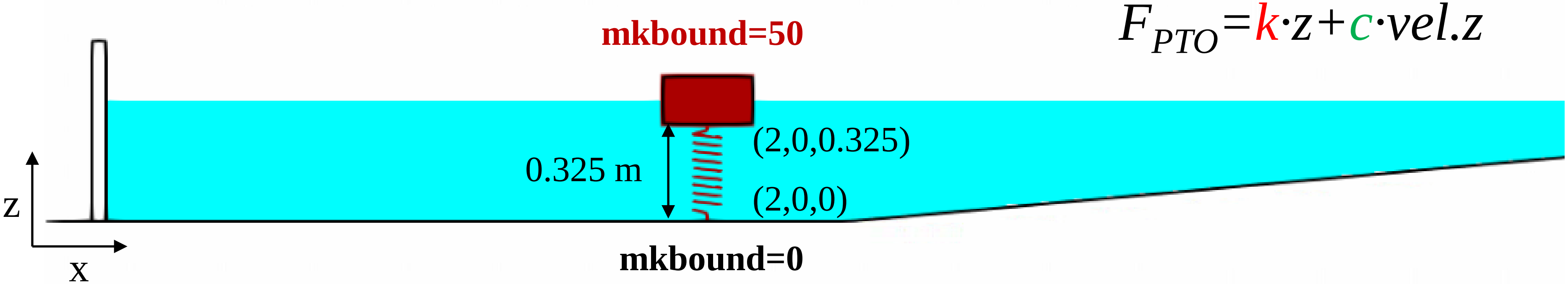


XML file: Special-Chrono

link_linearspring

Linear Spring

$$F_{PTO}=k\cdot z+c\cdot vel.z$$



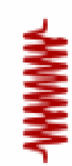

```xml
<chrono>
    <savedata value="0.05" />
    <schemescale value="1" />
    <bodyfixed id="Bottom" mkbound="0" />
    <bodyfloating id="Floater" mkbound="50" />
    <link_linearspring idbody1="Bottom" idbody2="Floater">
        <point_fb1 x="2.0" y="0.0" z="0.0" />
        <point_fb2 x="2.0" y="0.0" z="0.325" />
        <stiffness value="100.0" />
        <damping value="500.0" />
        <rest_length value="0.325" />
        <savevtk>
            <nside value="16" />
            <radius value="5.0" />
            <length value="3.0" />
        </savevtk>
    </link_linearspring>
</chrono>
```

A spring will connect two bodies:
- Body 1: “Bottom” (mkbound=0)
- Body 2: “Floater” (mkbound=50)

point_fb1: Point in body 1
point_fb2: Point in body 2
k*:* Stiffness [N/m]
c: Damping [Ns/m]
rest_length: Spring equilibrium length [m]

$$\boxed{F_{PTO}=k\cdot z+c\cdot vel.z}$$

VTK for visualisation: *Case_out/SpringVtk/Chrono_Springs_xxxx.vtk*
- **nside**: number of sections for each revolution
- **radius**: spring radius
- **length**: length of each revolution

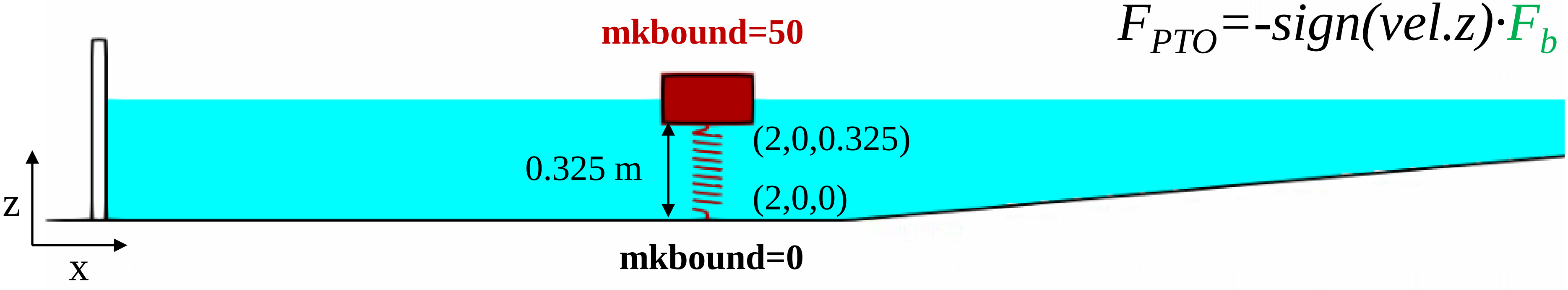


XML file: Special-Chrono

link_coulombdamping

Coulomb Spring

$$F_{PTO} = -sign(vel.z) \cdot F_b$$




```
<chrono>
    <savedata value="0.05" />
    <schemescale value="1" />
    <bodyfixed id="Bottom" mkbound="0" />
    <bodyfloating id="Floater" mkbound="50" />
    <link_coulombdamping idbody1="Bottom" idbody2="Floater">
        <point_fb1 x="2.0" y="0.0" z="0.0" />
        <point_fb2 x="2.0" y="0.0" z="0.325" />
        <damping value="10.0" />
        <rest_length value="0.325" />
        <savevtk>
            <nside value="16" />
            <radius value="5.0" />
            <length value="3.0" />
        </savevtk>
    </link_coulombdamping>
</chrono>
```

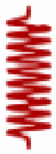
A spring will connect two bodies:
- Body 1: “Bottom” (mkbound=0)
- Body 2: “Floater” (mkbound=50)

point_fb1: Point in body 1
point_fb2: Point in body 2
F_b: Coulomb force [N]

rest_length: Spring equilibrium length [m]

VTK for visualisation: *Case_out/SpringVtk/Chrono_Springs_xxxx.vtk*

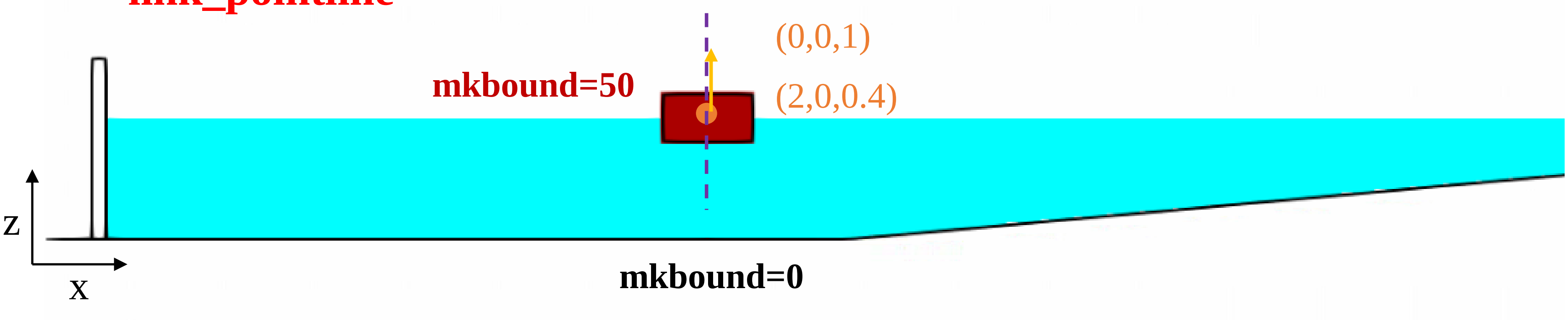
nside: number of sections for each revolution
radius: spring radius
length: length of each revolution



XML file: Special-Chrono

Sliding vector

link_pointline




```xml
<chrono>
    <savedata value="0.05" />
    <schemescale value="1"  />
    <bodyfixed id="Bottom" mkbound="0" />
    <bodyfloating id="Floater" mkbound="50" />
    <link_pointline idbody1="Floater">
        <slidingvector x="0" y="0" z="1" />
        <rotpoint x="2" y="0" z="0.4"  />
        <rotvector x="0" y="0" z="1" />
        <stiffness value="0" />
        <damping value="0" />
    </link_pointline>
</chrono>
```

The floating body “Floater” (mkbound=50) can slide along a given direction:

slidingvector: Sliding vector direction
rotpoint: Sliding point in the body
rotvector: Vector direction for rotation
stiffness: Torsional stiffness [Nm/rad]
damping: Torsional damping [Nms/rad]

Global directions are used

Different configurations may be possible:
- Sliding with a spherical joint is possible if **rotvector** is not defined
- Rotation fixed along the **slidingvector** can be defined using a second vector **rotvector2**

More examples in *examples\others\ChronoPointline*

XML file: Special-Chrono

link_spheric

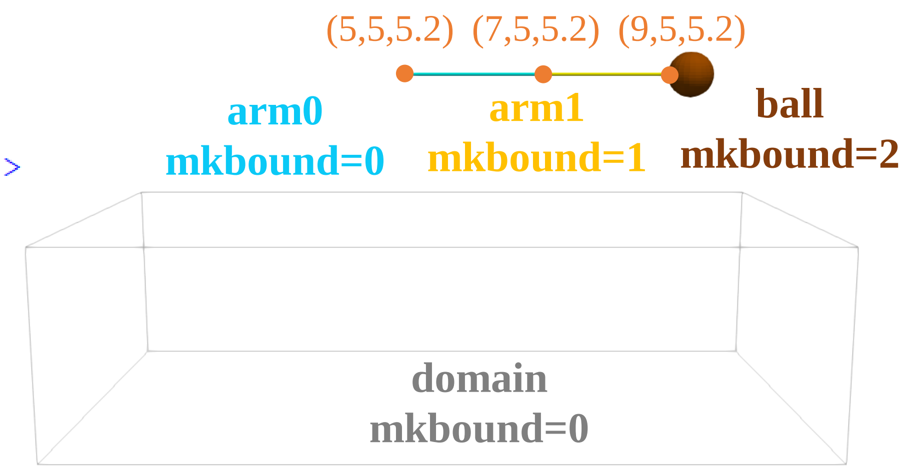
```
<chrono>
    <savedata value="0.02" />
    <schemescale value="1" />
    <bodyfixed id="domain" mkbound="10" />
    <bodyfloating id="arm" mkbound="0-1" />
    <bodyfloating id="ball" mkbound="2" />
    <link_spheric idbody1="arm0">
        <rotpoint x="5" y="5" z="5.2" />
        <stiffness value="0" />
        <damping value="0" />
    </link_spheric>
    <link_spheric idbody1="arm0" idbody2="arm1">
        <rotpoint x="7" y="5" z="5.2" />
        <stiffness value="0" />
        <damping value="0" />
    </link_spheric>
    <link_spheric idbody1="arm1" idbody2="ball">
        <rotpoint x="9" y="5" z="5.2" />
        <stiffness value="0" />
        <damping value="0" />
    </link_spheric>
</chrono>
```

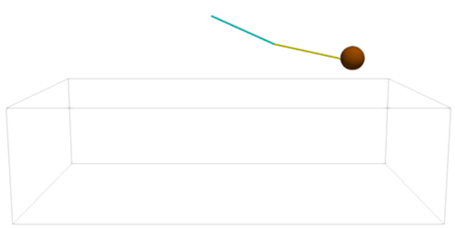
Spherical hinge

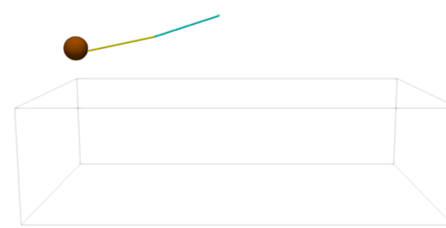
Three spherical links are defined between:

- “arm0” and the world
- “arm0” and “arm1”
- “arm1” and “ball”

rotpoint: Point for rotation
stiffness: Torsional stiffness [Nm/rad]
damping: Torsional damping [Nms/rad]







XML file: Special-Chrono

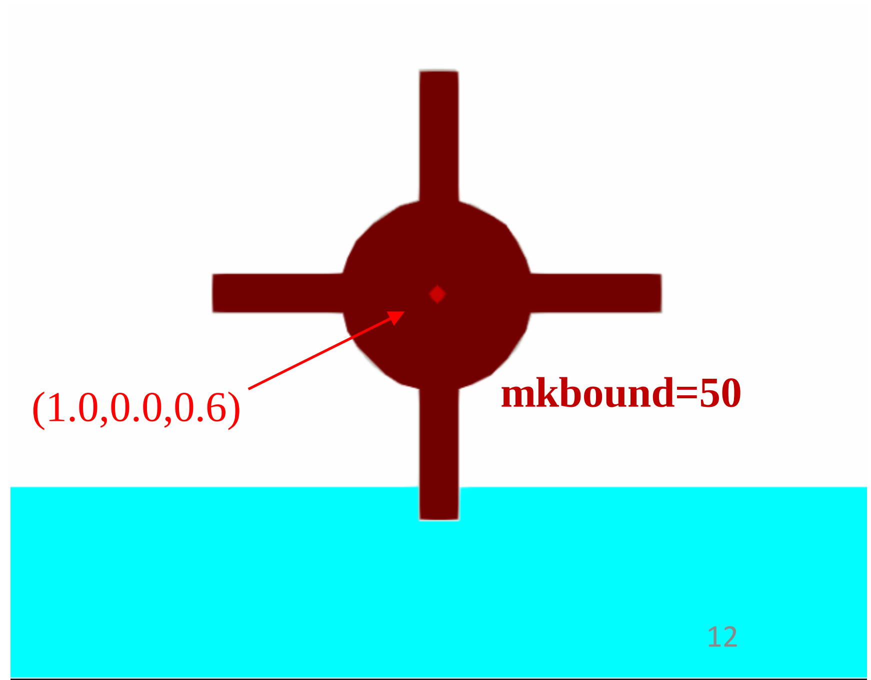
link_hinge

Hinge

$$M_{PTO}=k\cdot\theta+c\cdot vel.\theta$$

```xml
<chrono>
    <savedata value="0.01" />
    <schemescale value="1" />
    <bodyfixed id="domain" mkbound="0" />
    <bodyfloating id="wheel" mkbound="50" />
    <link_hinge idbody1="wheel">
        <rotpoint x="1.0" y="0" z="0.6" />
        <rotvector x="0.0" y="1.0" z="0.0" />
        <stiffness value="0.0" />
        <damping value="0.0" />
    </link_hinge>
</chrono>
```

A hinge is defined for “wheel” (mkbound=50)



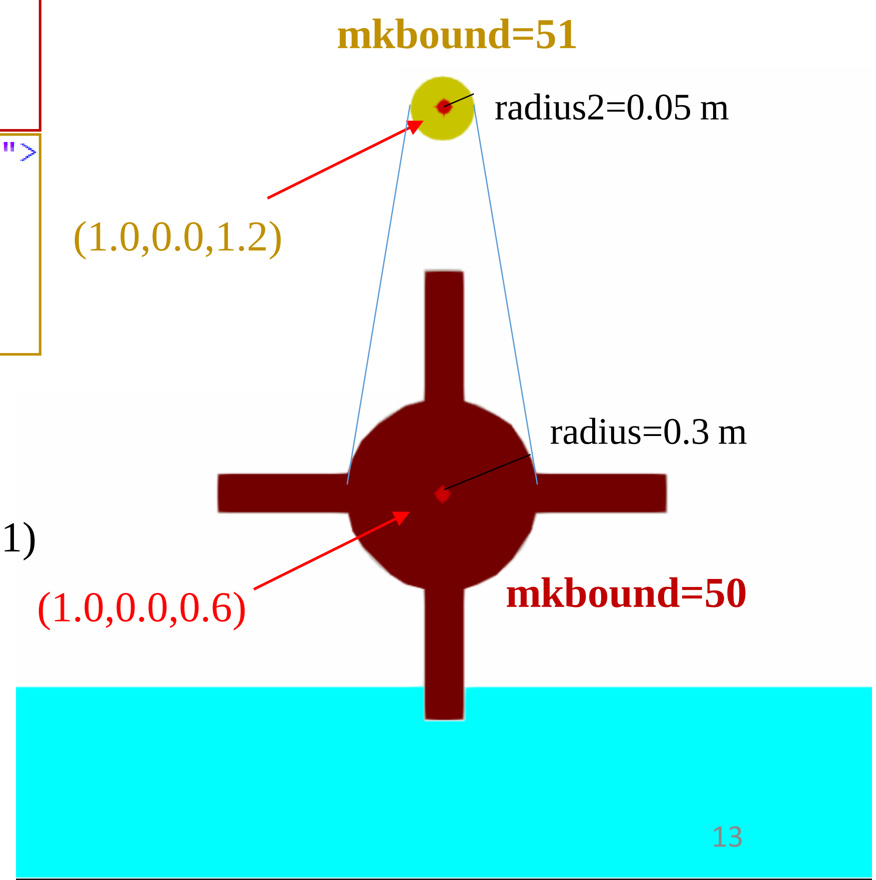
rotpoint: Point for rotation
rotvector: Vector direction for rotation
k: Torsional stiffness [Nm/rad]
c: Torsional damping [Nms/rad]

XML file: Special-Chrono

Pulley

link_pulley

```
<chrono>
    <savedata value="0.01" />
    <schemescale value="1" />
    <bodyfixed id="domain" mkbound="0" modelfile="AutoActual"/>
    <bodyfloating id="wheel" mkbound="50" modelfile="AutoActual"/>
    <bodyfloating id="gear" mkbound="51" modelfile="AutoActual"/>
    <link_hinge idbody1="wheel">
        <rotpoint x="1.0" y="0" z="0.6" />
        <rotvector x="0.0" y="1.0" z="0.0" />
        <stiffness value="0.0" />
        <damping value="0.0" />
    </link_hinge>
    <link_pulley idbody1="wheel" idbody2="gear">
        <rotpoint x="1.0" y="0" z="1.2" />
        <rotvector x="0.0" y="1.0" z="0.0" />
        <radius value="0.3" />
        <radius2 value="0.05" />/>
    </link_pulley>
</chrono>
```



A hinge is defined for “wheel” (mkbound=50)
A pulley connects “wheel” with “gear” (mkbound=51)

rotpoint: Point for rotation
rotvector: Rotation axis of the body
radius: Radius of “wheel”
radius2: Radius of “gear”