Tutorial 22: Adding Simulink Signals to Record Frame Movements

2016-12-18: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

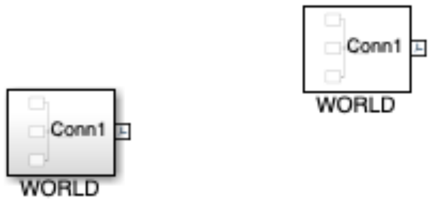
- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

Motivation for this tutorial: (Originally SolidGeometry 3.1 required)

2. Creating a new SimMechanics System

```
smbNewSystem ('SG_LIB_EXP_22');             % Creates the mechansim diagramm
smbDrawNow;
```

```
Creating temporary directory '/Users/timlueth/Desktop/tmp_SG_LIB_EXP_22/'
```

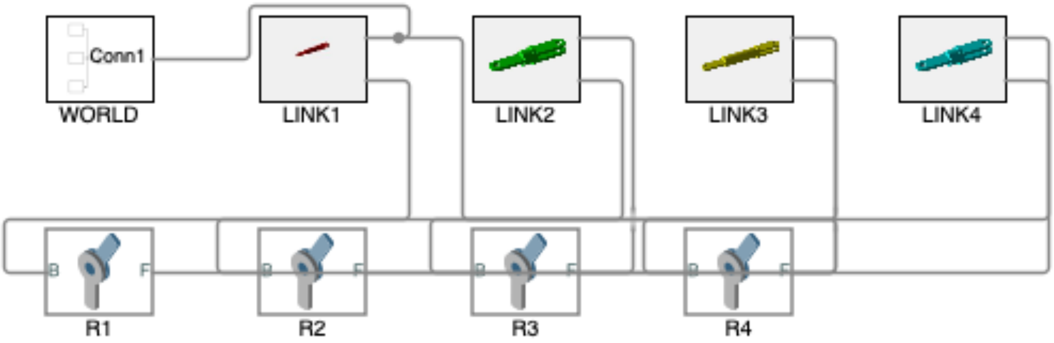


3. Create two links with length 50 and 80 and one or two mounting holes

```
SG1=SGmodelLink(80,'',1,2);                 % Creates a long rod with flange
SG2=SGmodelLink(50,'',1,2);                 % Creates a short rod with flange
```

4. Create SimMechanics models for the four links and four joints in different colors

```
smbCreateSG (SG1,'LINK1','r');                % Add long rod as LINK1
smbCreateSG (SG2,'LINK2','g');                % Add short rod as LINK2
smbCreateSG (SG1,'LINK3','y');                % Add long rod as LINK3
smbCreateSG (SG2,'LINK4','c');                % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
smbCreateConnection('WORLD.ORIGIN','LINK1.B'); % Connect Linkage to World Frame
smbDrawNow;
```



5. Create a video of the movements

smbVideoSimulation(3); % Show a 3 seconds video

6. Analyze the simulation for 3 Seconds

The result of a simulation is a strucutre that contains SimMultiBody states (xout) and recorded Simulink signals (sim). If there are no Simulink signals, sout is empty.

```
simOut=smbSimulate(3)
```

```
simOut = 
  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node] 
                   tout: [189x1 double] 
                   xout: [1x1 Simulink.SimulationData.Dataset] 

     SimulationMetadata: [1x1 Simulink.SimulationMetadata] 
           ErrorMessage: [0x0 char] 
```

The states contain the parameter = angles/velocity of the joints

```
xout = simOut.get('xout')
```

```
xout = 
Simulink.SimulationData.Dataset 'xout' with 8 elements

                              Name                     BlockPath
                              _______________________  ________________
    1  [1x1 State]            SG_LIB_EXP_22.R1.Rz.q    SG_LIB_EXP_22/R1
    2  [1x1 State]            SG_LIB_EXP_22.R1.Rz.w    SG_LIB_EXP_22/R1
    3  [1x1 State]            SG_LIB_EXP_22.R2.Rz.q    SG_LIB_EXP_22/R2
    4  [1x1 State]            SG_LIB_EXP_22.R2.Rz.w    SG_LIB_EXP_22/R2
    5  [1x1 State]            SG_LIB_EXP_22.R3.Rz.q    SG_LIB_EXP_22/R3
    6  [1x1 State]            SG_LIB_EXP_22.R3.Rz.w    SG_LIB_EXP_22/R3
    7  [1x1 State]            SG_LIB_EXP_22.R4.Rz.q    SG_LIB_EXP_22/R4
    8  [1x1 State]            SG_LIB_EXP_22.R4.Rz.w    SG_LIB_EXP_22/R4

  - Use braces { } to access, modify, or add elements using index.
```

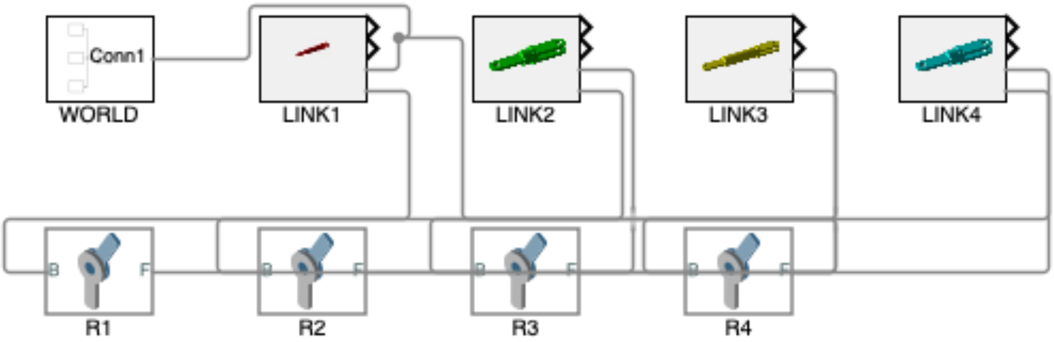
```
% There is no Simulink signals yet
% sout = simOut.get('sout')
```

7. Create Simulink signals for all the frames of the four links

```
smbAddFrameSensor ('LINK1.RF');
smbAddFrameSensor ('LINK2.RF');
smbAddFrameSensor ('LINK3.RF');
smbAddFrameSensor ('LINK4.RF');
```

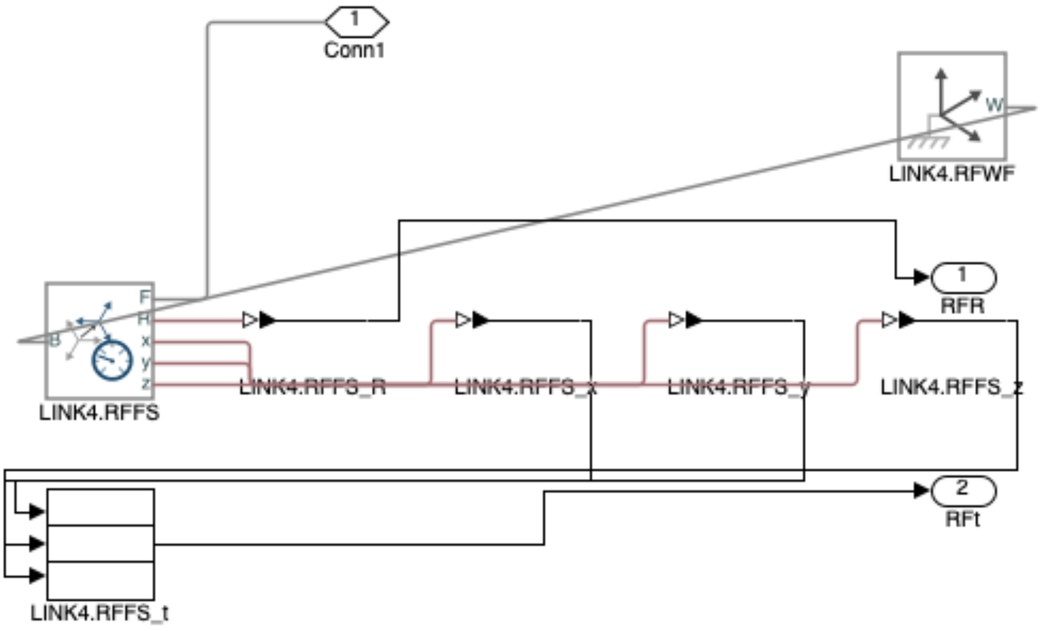
Now, all links have simulink signals and signal output for R and T of the reference frame

```
smbDrawNow;
```



The model of link4 is extendend by a transformation sensor

```
smbDrawNow ('LINK4.RF_T');
```



8. Simulate and record those signals too

```
simOut=smbSimulate(3)

return
smbVideoSimulation(3);
```

```
simOut = 
  Simulink.SimulationOutput:
                  simlog: [1x1 simscape.logging.Node] 
                    sout: [1x1 Simulink.SimulationData.Dataset] 
                    tout: [189x1 double] 
                    xout: [1x1 Simulink.SimulationData.Dataset] 

     SimulationMetadata: [1x1 Simulink.SimulationMetadata] 
           ErrorMessage: [0x0 char] 
```

The states contain the parameter = angles/velocity of the joints

```
xout = simOut.get('xout')
```

TheSimulink signals are related to the reference rotation and translation

```
sout = simOut.get('sout')
T1=smbTofSimOut(simOut,'LINK1.RF'); VL1=squeeze(T1(1:3,4,:))';
T2=smbTofSimOut(simOut,'LINK2.RF'); VL2=squeeze(T2(1:3,4,:))';
```

```
T3=smbTofSimOut(simOut,'LINK3.RF'); VL3=squeeze(T3(1:3,4,:))';
T4=smbTofSimOut(simOut,'LINK4.RF'); VL4=squeeze(T4(1:3,4,:))';
SGfigure; axis on; view(0,90); grid on;
VLplot(VL1,'r.-');
VLplot(VL2,'g.-');
VLplot(VL3,'y.-');
VLplot(VL4,'c.-');
drawnow;
```

Final remarks on toolbox version and execution date

```
VLFLlicense
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016b on 2016-12-18*
- __________, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2023a