AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
MECHATRONICS ENGINEERING DEPARTMENT

Spring 2022

MCT 333: Mechatronic System Design

Lab Session 01: Linking Inventor and Simscape Multibody

Contents[*]

Background

Simscape™ Multibody™ Link is a plug-in that you install on your CAD application to export your CAD assembly models. The plug-in generates the files that you need to import the model into the Simscape Multibody environment (using the *smimport* function). You can install the plug-in on three CAD applications:

- SolidWorks®
- Autodesk Inventor®
- PTC® Creo™

The plug-in generates an XML file detailing the structure and properties of your CAD assembly and 3-D geometry files for visualizing the various CAD parts. You can then import the files into Simscape Multibody software, which parses the XML data and automatically generates an equivalent multibody model.

Exercise 1: Install Simscape Multibody Link Plug-In.

Before You Begin

You must have one of the supported CAD applications:

- Autodesk Inventor software
- PTC Creo software
- SolidWorks software

Your MATLAB and CAD installations must have the same system architecture—e.g., Windows 64-bit.

Step 1: Get the Installation Files

1. Go to the Simscape Multibody Link download page.
2. Follow the prompts on the download page.
3. Save the zip archive and MATLAB file in a convenient folder.

Select the file versions matching your MATLAB release number and system architecture—e.g., release R2016a and Win64 architecture. Do not extract the zip archive.

Step 2: Run the Installation Function

1. Run MATLAB as administrator.
2. Add the saved installation files to the MATLAB path.
   You can do this by entering *addpath('foldername')* at the MATLAB command prompt. Replace *foldername* with the name of the folder in which you saved the installation files—e.g., *C:\Temp*.
3. At the MATLAB command prompt, enter *install_addon('zipname')*.
   Replace *zipname* with the name of the zip archive—e.g., smlink.r2015b.win64.zip.

[*]Parts of this lab sheet are based on the Simscape Multibody documentation, provided by Mathworks

Step 3: Register MATLAB as an Automation Server

Each time you export a CAD assembly model, the Simscape Multibody Link plug-in attempts to connect to MATLAB. For the connection to occur, you must register MATLAB as an automation server. You can do this in two ways:

- In a MATLAB session running in administrator mode — At the command prompt, enter *regmatlabserver*
- In an MS-DOS window running in administrator mode — At the command prompt, enter matlab -regserver.

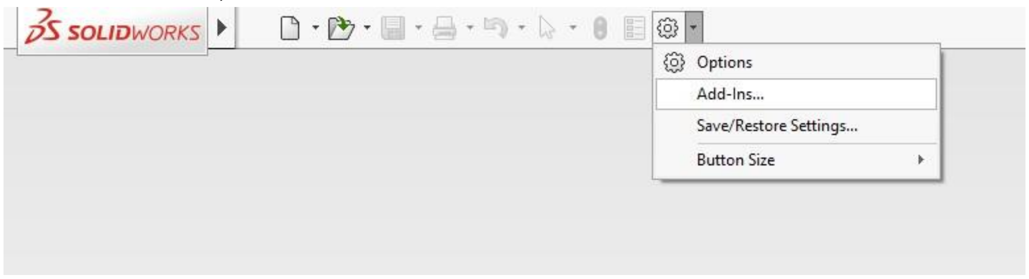
Step 4: Enable the Simscape Multibody Link Plug-In

Before you can export an assembly, you must enable the Simscape Multibody Link plug-in on your CAD application. To do this, see:

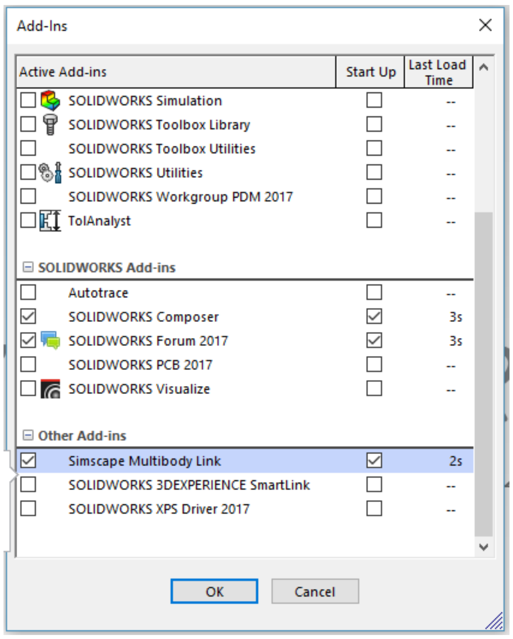
- **Enable Simscape Multibody Link Inventor Plug-In**
  1. At the MATLAB® command prompt, enter *smlink_linkinv*.
  2. A Simscape Multibody Link menu appears in the Inventor menu when you start or open a CAD assembly.

  If your computer has more than one Inventor application, the smlink_linkinv command adds the Simscape Multibody Link plug-in to all installations simultaneously.

- **Enable Simscape Multibody Link SolidWorks Plug-In**
  1. At the MATLAB® command prompt, enter *smlink_linksw*.
  2. Start SolidWorks.
  3. In the **Tools** menu, select **Add-Ins**.



  4. In the Add-Ins dialog box, select the **Simscape Multibody Link** check box. A Simscape Multibody Link menu appears in the SolidWorks menu bar when you start or open a CAD assembly.



Add-Ins

Active Add-ins	Start Up	Last Load Time
☐ SOLIDWORKS Simulation	☐	--
☐ SOLIDWORKS Toolbox Library	☐	--
☐ SOLIDWORKS Toolbox Utilities	☐	--
☐ SOLIDWORKS Utilities	☐	--
☐ SOLIDWORKS Workgroup PDM 2017	☐	--
☐ TolAnalyst	☐	--
⊟ SOLIDWORKS Add-ins		
☐ Autotrace	☐	--
☑ SOLIDWORKS Composer	☑	3s
☑ SOLIDWORKS Forum 2017	☑	3s
☐ SOLIDWORKS PCB 2017	☐	--
☐ SOLIDWORKS Visualize	☐	--
⊟ Other Add-ins		
☑ Simscape Multibody Link	☑	2s
☐ SOLIDWORKS 3DEXPERIENCE SmartLink	☐	--
☐ SOLIDWORKS XPS Driver 2017	☐	--

OK Cancel

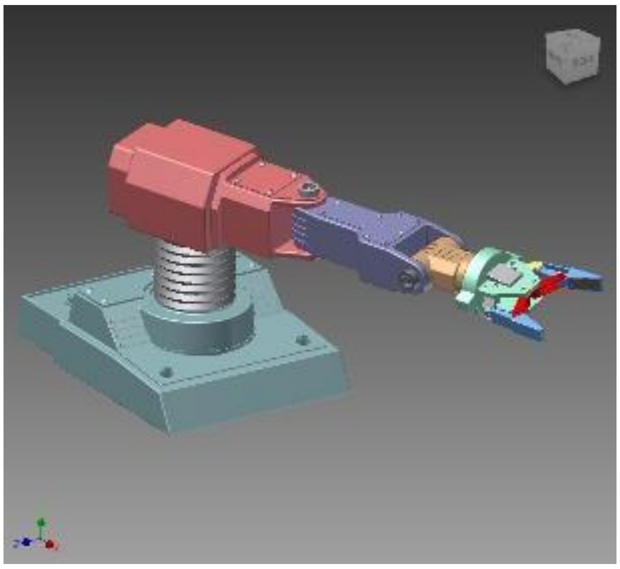
If your computer has more than one SolidWorks installation, the *smlink_linksw* command adds the Simscape Multibody Link plug-in to all installations simultaneously. However, you must select the Simscape Multibody Link check box in the Add-Ins dialog box individually for each installation you want to export CAD from.

Exercise 2: Export a CAD mechanism drawn on Inventor to Simscape Multibody.

In this example, you export an Autodesk Inventor® CAD assembly that represents a robot arm. The export procedure generates one XML file and a set of geometry files that you can import into Simscape™ Multibody™ to generate a new s model.

The example begins with a procedure to export the CAD assembly. Information on the robot CAD files and CAD Export follows the export procedure

The Robot Assembly Model



Open the Assembly Model

Before you can export the robot assembly, you must load the assembly into Inventor.

1. Open Inventor on your machine.
2. Select **File** > **Open**
3. Navigate to the file directory that contains the robot CAD files.

   Note: The directory that contains the robot CAD files for the Autodesk Inventor platform is

   <MATLAB Root>\toolbox\physmod\smlink\smlinkdemos\inventor\robot

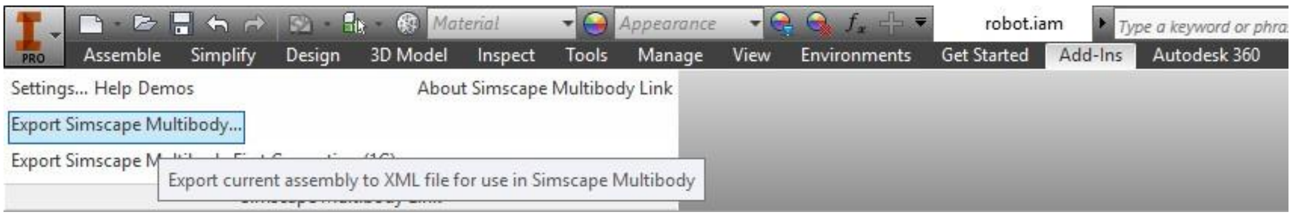
   Where <MATLAB Root> for R2016a is usually C:\Program files\MATLAB\R2016a
4. Select file robot.IAM.
5. Click **Open**.

The CAD platform opens the robot assembly.

Export the Model

Once you successfully open the robot CAD assembly in your Inventor installation, you can export the assembly in a format compatible with the latest Simscape Multibody technology:

1. In the Inventor toolbar, select **Add-Ins** > **Export Simscape Multibody**.



Export current assembly to XML file for use in Simscape Multibody

2. In the **File name** field of the **Save As** dialog box, enter *sm_robot* and select a suitable directory to export the files. For example, save it to the desktop under a folder called *robot*
3. Click **Save**.

   Note: Large CAD assemblies require more time to complete the export process. Allow up to a few minutes for the export process to complete.

Check the Exported Files

Confirm the following files exist in the export directory you specified:

- XML file—Provides structure of model and parameters of parts.
- Geometry files—Provide surface geometry of parts.

About the Example CAD Files

The CAD assembly files are present in your Simscape Multibody Link installation. You can access the files in the following directory:

<MATLAB Root>\matlab\toolbox\physmod\smlink\smlinkdemos\inventor\robot

Substitute <MATLAB Root> with the root directory of your MATLAB® installation.

Note: If you are not sure what your MATLAB root directory is, at the MATLAB command line enter matlabroot. MATLAB returns the root directory for your installation.

The \inventor\robot directory contains a set of CAD files that define each CAD part and CAD assembly. Part file names contain the file extension .IPT. Assembly file names contain the extension .IAM.

The robot assembly contains nine parts and two assemblies: robot.IAM and grip.IAM. File robot.IAM models the robot root assembly. File grip.IAM models the robot grip subassembly.

About CAD Export

The CAD export procedure generates one XML multibody description file and a set of geometry files. The XML file contains the structure of the assembly and the parameters that define each part. The geometry files define the 3-D surface of each part. Once the export procedure is complete, you can import the XML multibody description file into Simscape Multibody software. Simscape Multibody uses the file to automatically generate a new Simscape Multibody model.

Exercise 3: Import an xml of a CAD mechanism generated using the Link plugin into Simscape Multibody.

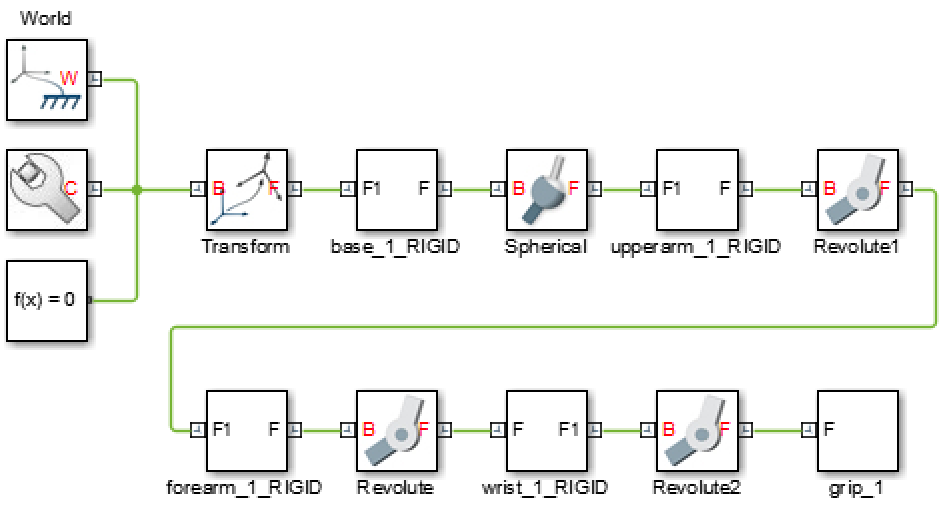
This example shows how to generate a Simscape™ Multibody™ model from a multibody description XML file using the *smimport* function. The example is based on a multibody description file that you exported in Exercise 2.

Import the Model

1. In MATLAB, navigate to the folder that you saved in Exercsise 2.
2. At the MATLAB command prompt, enter the command:

   smimport('sm_robot');

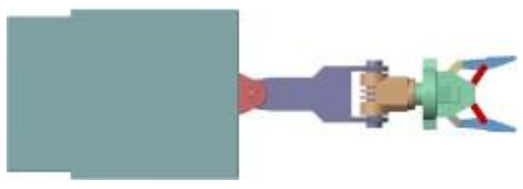
Simscape Multibody software generates the model described in the sm_robot.xml file using the default smimport function settings.



The blocks in the generated model are parameterized in terms of MATLAB variables. The numerical values of these variables are defined in a data file that is named *sm_robot.m* and stored in the same active folder as the generated model.

Visualize the Model

Update the diagram to visualize the model. You can do this from the Simulink® menu bar by selecting **Simulation** > **Update Diagram**. Mechanics Explorer opens with a static visualization of the robotic arm model in its initial configuration.



The default view convention in Mechanics Explorer differs from that in the CAD application used to create the original assembly model. Mechanics Explorer uses a Z-axis-up view convention while the CAD application uses a Y-axis-up view convention.

Change the view convention from the Mechanics Explorer toolstrip by setting the **View convention** parameter to Y up (XY Front). Then, select a standard view from the **View** > **Standard Views** menu to apply the new view convention.

Build on the Model

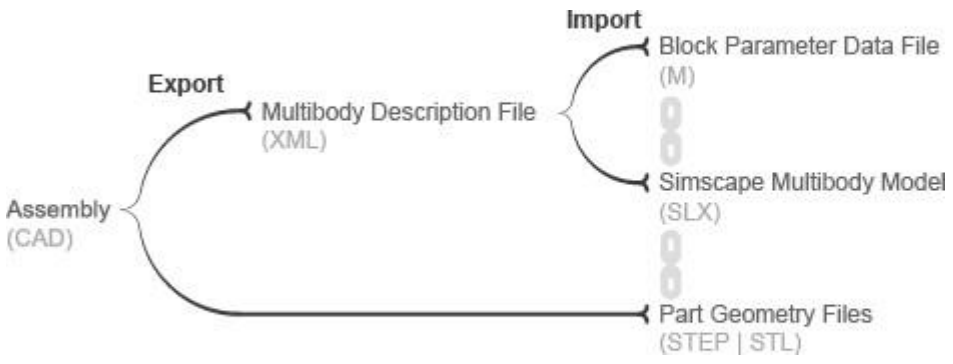
Try to simulate the model. Because the robotic arm lacks a control system, it simply flails under gravity. You can use Simulink blocks to create the control system needed to guide the robotic arm motion. A control system would convert motion sensing outputs into actuation inputs at the various joints. You can expose the sensing and actuation ports from the joint block dialog boxes.

Exercise 4: Translating a CAD Model to a Simscape Multibody model

You can translate a CAD model into an equivalent Simscape™ Multibody™ block diagram. The conversion relies on the *smimport* function featuring an XML multibody description file name as its central argument. The XML file passes to Simscape Multibody software the data it needs to recreate the original model—or an approximation of it if unsupported constraints exist in the model.

You translate a CAD model in two steps—export and import. The export step converts the CAD assembly model into an XML multibody description file and a set of STEP or STL part geometry files. The import step converts the multibody description and part geometry files into an SLX Simscape Multibody model and an M data file. The model obtains all block parameter inputs from the data file.

CAD Translation Steps

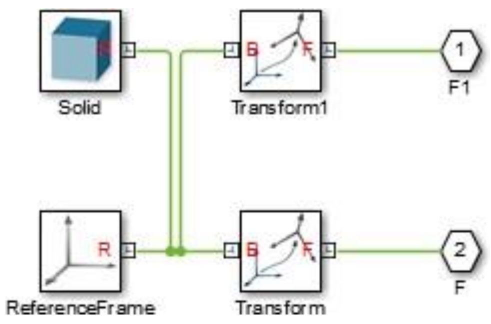


What's in a Translated Model?

The translated model represents the CAD parts—referred to as bodies in Simscape Multibody software—using Simulink® subsystems that comprise multiple Solid and Rigid Transform blocks. The Solid blocks provide the body geometries, inertias, and colors. The Rigid Transform blocks provide the frames with the required poses for connection between bodies.

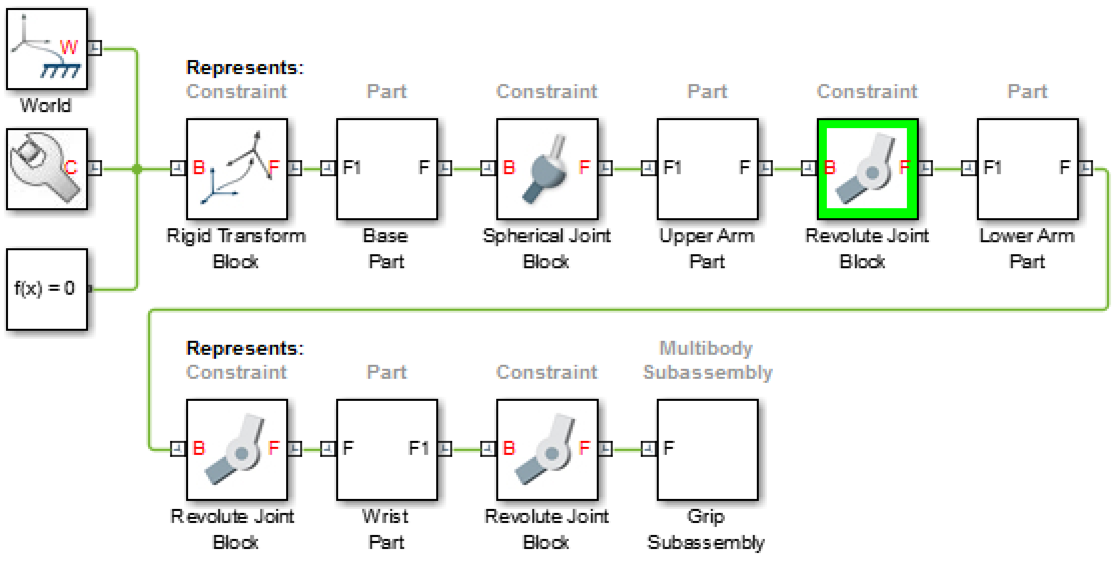
Consider the upper arm body of a CAD robotic arm model, shown in the figure. The Simulink subsystem for this body consists of one Solid block connected to a pair of Rigid Transform blocks. The Solid block provides the reference to the upper arm geometry file and the inertial properties derived from the CAD model. The Rigid Transform blocks provide the frames for connection to the robot base and lower arm bodies.

Simulink Subsystem Representing Upper Arm Body



CAD joints, constraints, and mates translate into Simscape Multibody software as combinations of joint and constraint blocks. In the CAD robotic arm example, the constraints between the upper arm and the lower arm translate into a Revolute Joint block. This block sits between the Simulink Subsystem blocks that represent the upper arm and lower arm bodies.

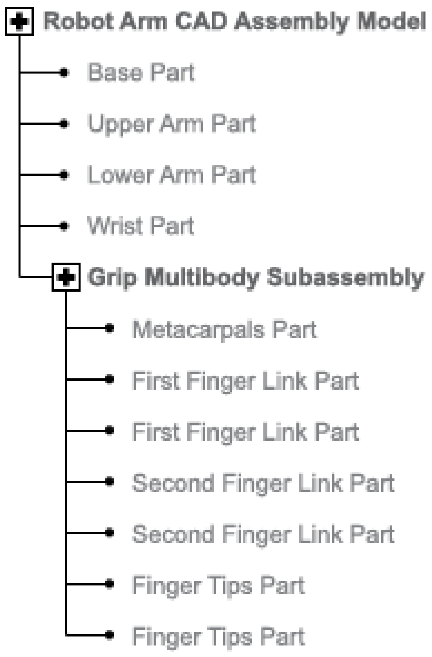
Simulink Subsystem Representing Upper Arm Body



Important:

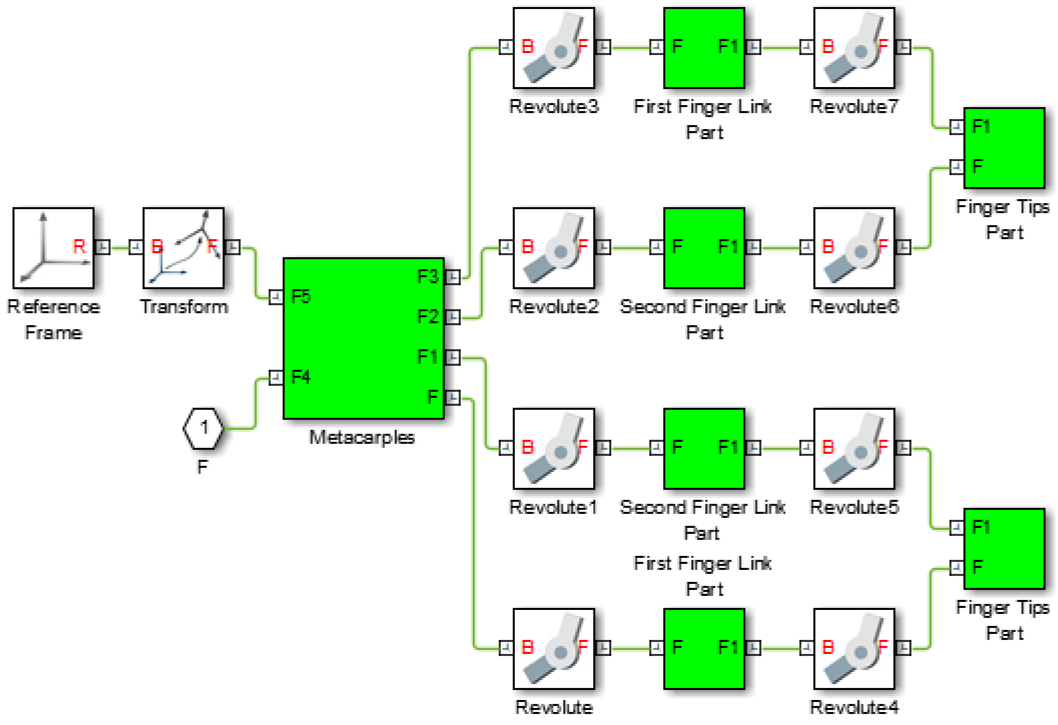
By default, the translated model preserves the structural hierarchy of the original CAD model. If the source model is a CAD model with multibody subassemblies, the subassemblies convert in Simscape Multibody software into multibody Simulink subsystems. Consider again the CAD robotic arm model. The model contains a grip multibody subassembly with seven bodies, shown schematically in the figure.

CAD Robotic Arm Model Hierarchy



During translation, the grip subassembly converts into a Simulink subsystem with seven Simulink subsystems, one for each body.

Multibody Simulink Subsystem with Body Simulink Subsystems



What's in a Data File?

Blocks in the translated model are parameterized in terms of MATLAB® variables defined in the data file. These variables are stored in structure arrays named after the various block types. The structure arrays are nested in a parent data structure named smiData or a custom string that you specify.

Consider an imported model with a data structure named smiData. If the model contains Revolute Joint blocks, the parameter data for these blocks is the structure array smiData.RevoluteJoint. This structure array contains a number of data fields, each corresponding to a different block parameter.

The structure array fields are named after the block parameters. For example, the position state target data for the Revolute Joint blocks is in a field named Rz_Position_Target. If the model has two Revolute Joint blocks, this field contains two entries—smiData.RevoluteJoint(1).Rz_Position_Target and smiData.RevoluteJoint(2).Rz_Position_target.

Each structure array index corresponds to a specific block in the imported model. The index assignments can change if you regenerate a data file from an updated XML multibody description file. The smimport function checks the prior data file, when specified, to ensure the index assignments remain the same.

Exporting a CAD Model

You can technically export a CAD assembly model from any CAD application. The Simscape Multibody Link CAD plug-in provides one means to export a model in a valid XML format. The plug-in is compatible with three desktop CAD applications: SolidWorks®, PTC® Creo™, and Autodesk® Inventor®. The plug-in generates not only the XML multibody description file but also any geometry files required for visualization in the final translated model.

If you use an unsupported CAD application and a URDF converter exists for your CAD application, you may be able to export your model in URDF format and import the URDF file into the Simscape Multibody environment. Note, however, that the URDF specification forbids closed-chain model topologies, such as those of four-bar linkages and gear assemblies. For more information, see URDF Import (Simscape Multibody).

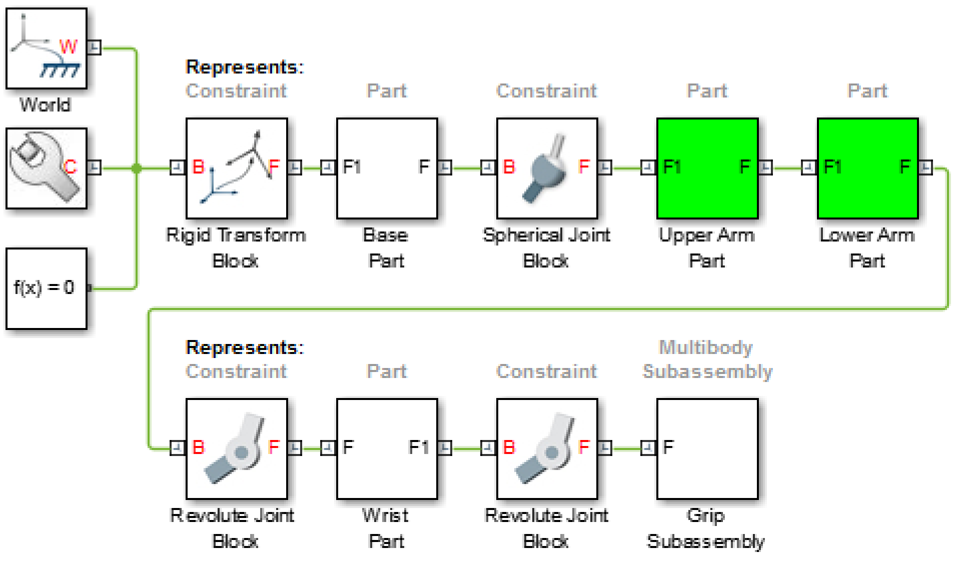
A Note About Export Errors

If the Simscape Multibody Link plug-in cannot export a part geometry file or translate a CAD constraint set, the software issues an error message. The error message identifies the bodies with missing geometry files and any unsupported constraints. You can import the generated XML multibody description file into Simscape Multibody software, but the resulting model may not accurately represent the original CAD assembly model.

CAD Import Errors

If a part geometry file is invalid or missing, the corresponding body cannot show in the Simscape Multibody visualization utility. If a CAD assembly model contains an unsupported constraint combination between bodies, Simscape Multibody software joins the bodies with a rigid connection. The rigid connection can take the form of a direct frame connection line, Rigid Transform block, or Weld Joint block.

Rigid Connection Due to Unsupported Constraints



If Simscape Multibody software cannot translate a CAD constraint combination, it issues a warning message on the MATLAB command window identifying the affected bodies and their connection frames. For example:

Warning: The set of constraints between upperarm_1_RIGID and forearm_1_RIGID

could not be mapped to a joint. A rigid connection has been added between port F
of upperarm_1_RIGID and port F1 or forearm_1_RIGID for these constraints.

Simplifying Model Topology

You can import a CAD model with a simplified topology. So that you can do this, the smimport function provides the ModelSimplification argument. You can set this argument to:

- bringJointsToTop to group each set of rigidly connected parts into a new subsystem and promote all joints to the top level of the model hierarchy.
- groupRigidBodies to group rigidly connected parts into subsystems (and leave joints in their original places in the model hierarchy).
- None to import the model as is, without simplification.

Use the bringJointsToTop or groupRigidBodies option if your CAD model has many rigidly connected components, such as nuts and bolts, that you prefer to group together—for example, to more intuitively grasp the key components of the model at a glance of the block diagram.

Use the bringJointsToTop option if your CAD model has joints inside subassemblies and you prefer to expose them at the top level—for example, to work with joint actuation and sensing signals without having to search for the joints inside different subsystems.

Note that model simplification is available for CAD import only. URDF models have flat topologies with little need for topology simplification.

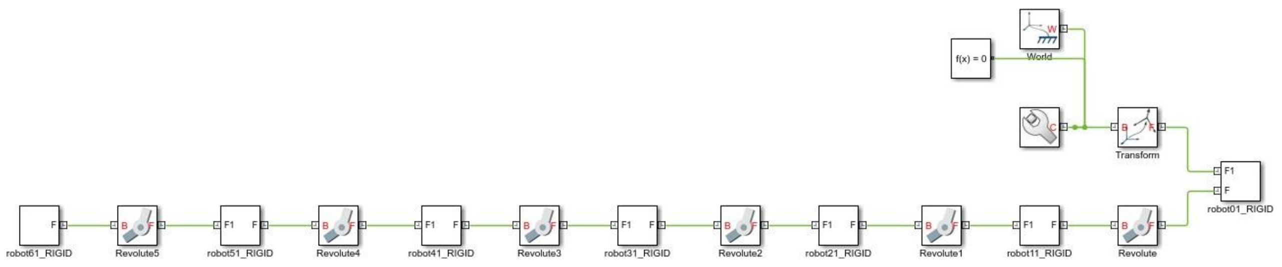
Exercise 5: Mechatronic system design and modeling using Inventor-Simscape.

In this example, we will investigate a practical workflow for designing and modelling a mechatronic system using Autodesk Inventor and Simscape Multibody.

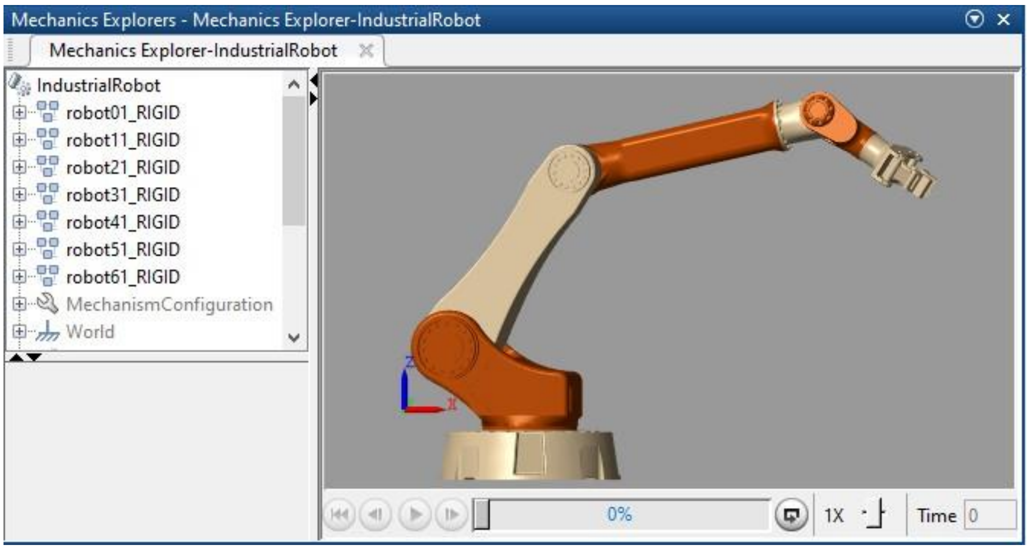
Design the mechanical System using Inventor and export it to Simscape Mutlibody

We first start by designing the mechanical system and drawing it using Autodesk Inventor 2017. Since this is a time-consuming process and the focus of this exercise is on Simscape Multibody, we will use an already assembled Autodesk Inventor model that we will download from the internet. Our target system for this exercise will be a 6-DOF robot.

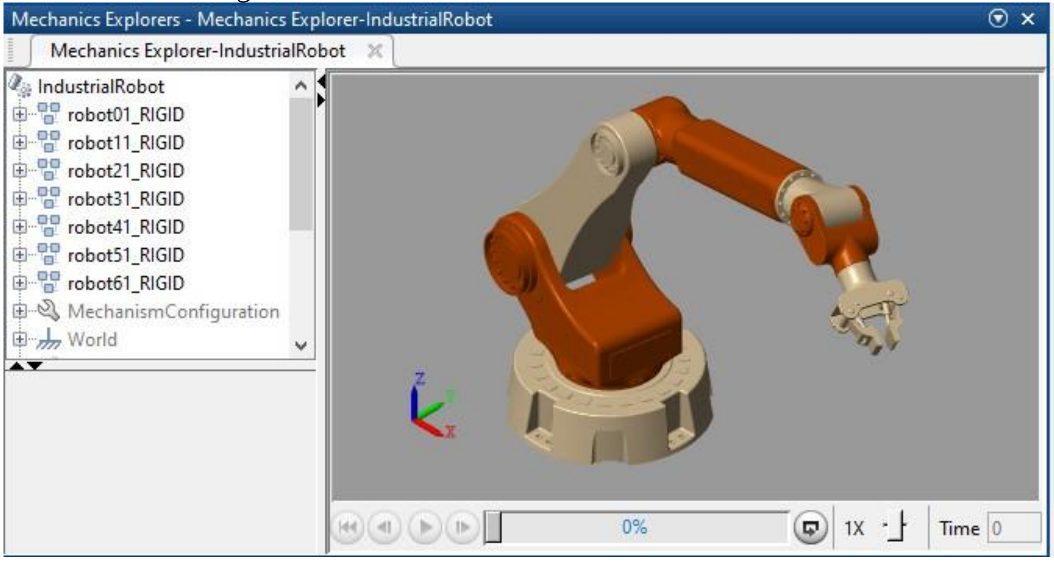
1. Download the mechanical assembly using this link: https://grabcad.com/library/industrial-robot-12
   Note: You need to have Autodesk Inventor 2017 to be able to open this assembly. Previous versions will not be able to open it.
2. The link contains an assembly for the mechanical structure of the robot. Follow the same steps as you did in **Exercises 2** and **3** to export the assembly from Inventor to an XML file and import it into Simscape Multibody.
3. After you have imported the model you should end up with a model like this:



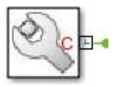
4. To view the imported model click on **Simulation** > **Update Diagram** you should see something like this:



5. As with the case in Exercise 3, the view of the robot is different from that in Inventor. Adjust the view convention to get an isometric view similar to this:

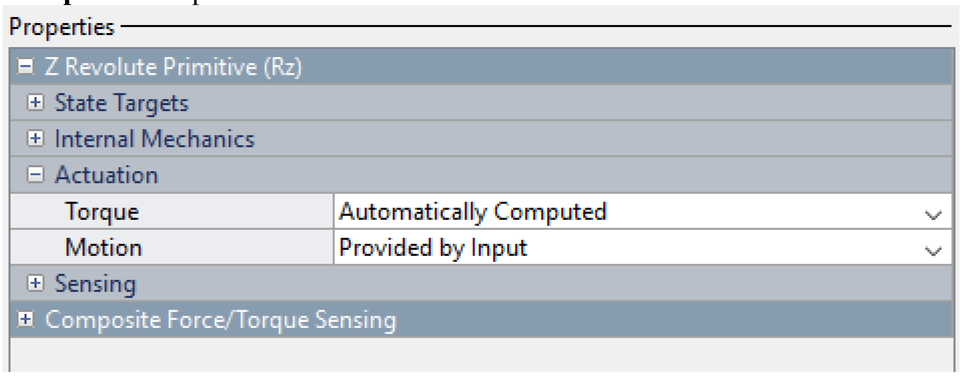


6. Adjust the gravity of the environment by double clicking the “Machine configuration block”



   Make sure that the gravity is acting in the negative Z-axis direction by setting the Gravity to [0 0 -9.80665]
7. Run the simulation and observe what happens to our robot.
   The robot falls under the effect of gravity because we haven’t defined any actuators to maintain its position or control its motion. Also, the robot does not come to a stop no matter how long you run the simulation because no damping is defined in the system. We will address this in the next steps.
8. First, we will try to fix the position of the robot and later on, we will try to control its motion. To fix the position of the robot we must apply an actuation input to all the joints of the robot. Double click the first revolute joint named **Revolute** and from the joint dialogue under **Actuation** change the **Motion** to **Provided by Input** and the **Torque** to **Automatically**

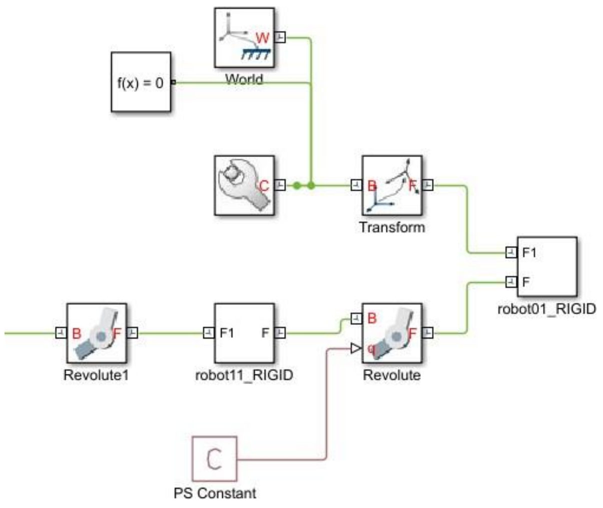
Computed and press OK.



Properties

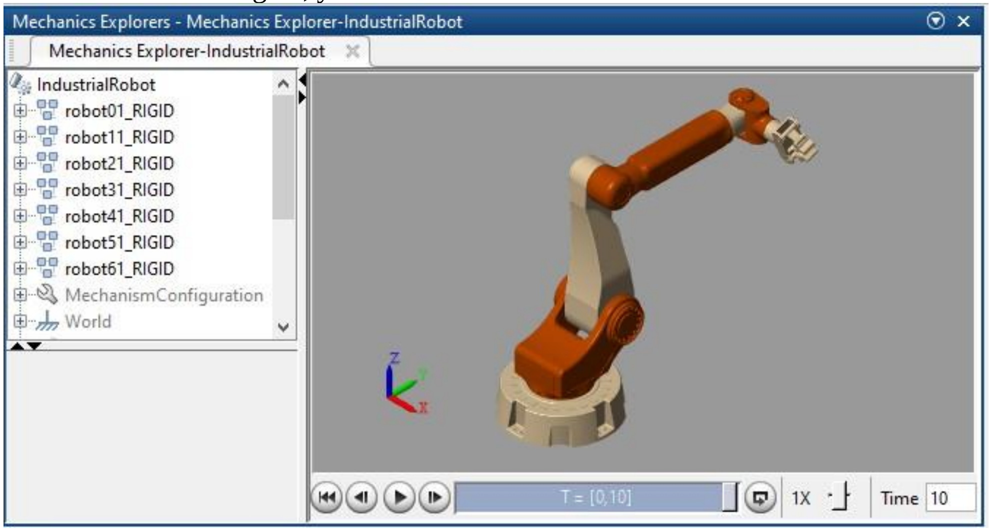
⊟ Z Revolute Primitive (Rz)	
⊞ State Targets	
⊞ Internal Mechanics	
⊟ Actuation	
Torque	Automatically Computed
Motion	Provided by Input
⊞ Sensing	
⊞ Composite Force/Torque Sensing	

9. A new input port will appear in the joint block. It needs to be connected to a physical signal to represent the angle that we want the joint to move to. We can either use a S-PS block and provide a normal Simulink signal or instead we will use the simpler **Constant PS** block which is found under **Simscape>Foundation Library>Physical signals>Sources>Constant PS.** We will then connect the source block to our newly created input port:



We will set the value of the constant to zero, to fix the robot at the zero position of the first joint.

10. Run the Simulation again and observe although the robot falls under gravity. It doesn't rotate around the Z-axis because we fixed the first joint.
11. We need now to fix the rest the of the joints of the robot to hold the robot in its position. Repeat Steps 7 and 8 for the other Revolute joints in the mechanism.
12. Run the Simulation again, you should see the robot fixed in this orientation:

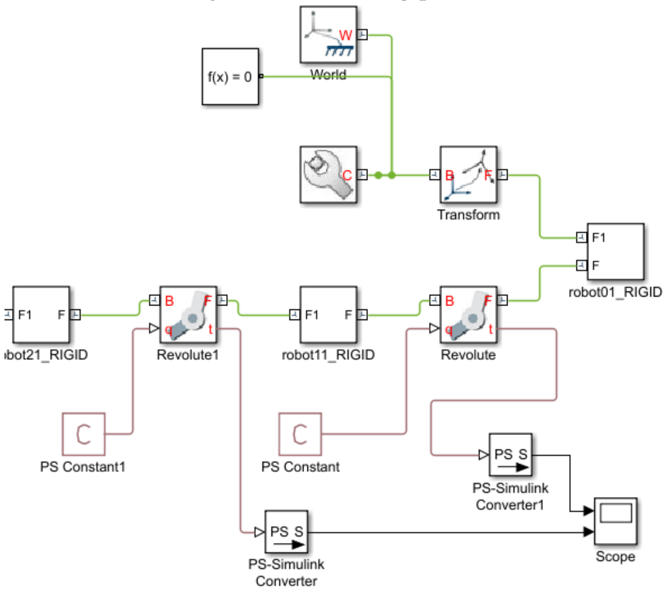


Why did the orientation of the robot change in this step from that of step 5?

…………………………………………………………………………………………………

…………………………………………………………………………………………………
…………………………………………………………………………………………………

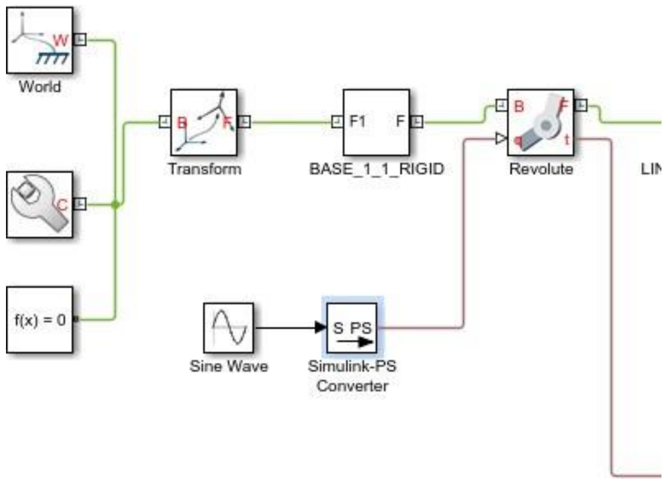
13. We have provided a constant input position input to the joints. Lets examine the torque that we need to apply on the joints to keep the robot fixed in this position. Double click the first revolute joint named **Revolute** and from the joint dialogue under **Sensing,** check the box next to **Actuation Torque.** Repeat the same step for the joint **Revolute 1**.
14. Connect the newly created sensing ports as shown here:



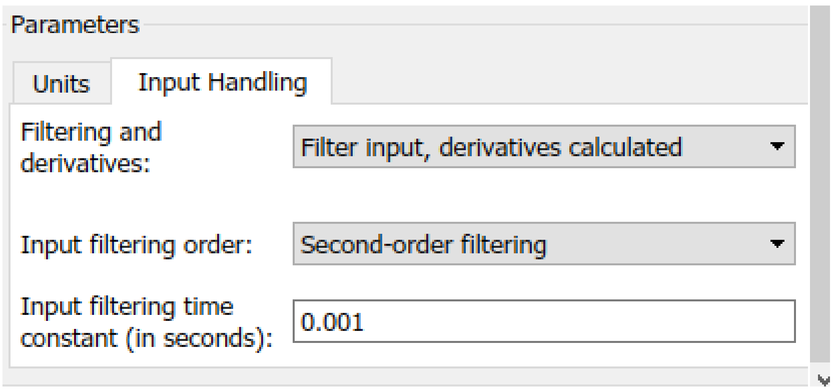
15. Run the simulation and observe the required actuation torques for both joints from the Scope. **Comment** on the values and what would affect them. Can you modify the orientation of the robot to minimize the actuator torque of the second joint?

…………………………………………………………………………………………………
…………………………………………………………………………………………………
…………………………………………………………………………………………………

16. Now that we have fixed the position of the robot we will try to move one of the joints. We can do this by changing the input of the first joint from **PS Constant** to an ordinary Sine wave block connected through a **Simulink-PS block**



Important note: In order to provide a varying motion input signal you need to provide the derivatives of the signal also. You can do this by double clicking on the **S PS** block and under input handling provide Second-Order filtering.


Parameters

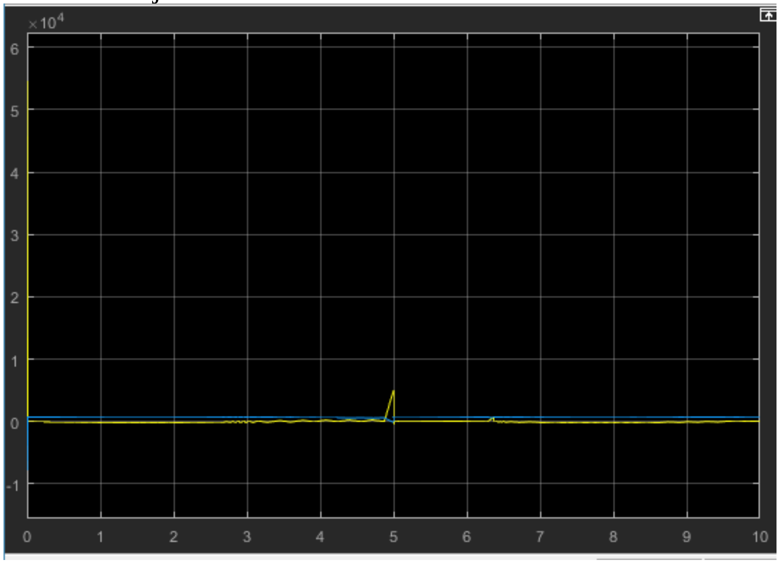
Units | Input Handling

Filtering and derivatives: Filter input, derivatives calculated

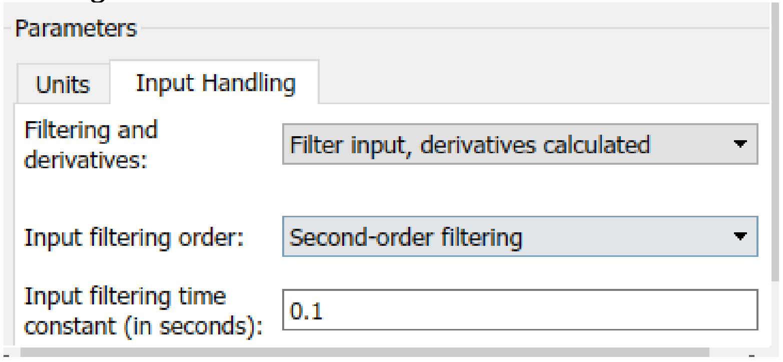
Input filtering order: Second-order filtering

Input filtering time constant (in seconds): 0.001

17. Run the Simulation and observe the motion of the robot.
18. The robot moves according to the input signal which is an oscillating sine wave with frequency 1 rad/sec. Double click on the scope to see the torque required to achieve this motion profile for the first two joints:



Note that the torque at time zero is very high (about 5 × 10⁴ Nm). This is because we are suddenly forcing the robot to follow the motion. To fix this problem we need to adjust the **Input filtering time constant** in the **S PS** block. Set it to 0.1 sec as shown below:


Parameters

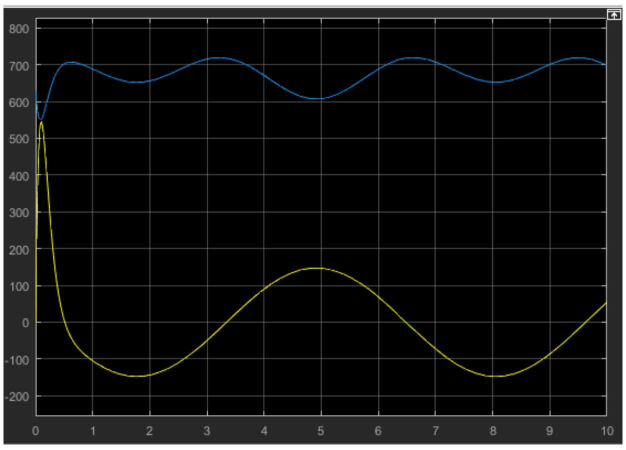
Units | Input Handling

Filtering and derivatives: Filter input, derivatives calculated

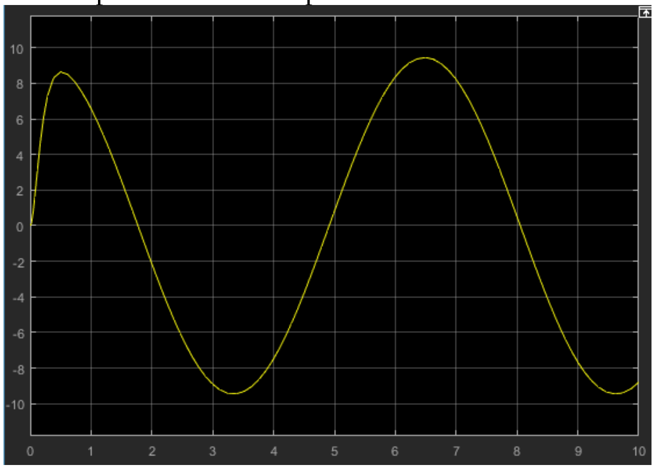
Input filtering order: Second-order filtering

Input filtering time constant (in seconds): 0.1

19. Run the simulation again and observe the torque values through the scope. It should look like the following (don't forget to Autoscale the scope using):

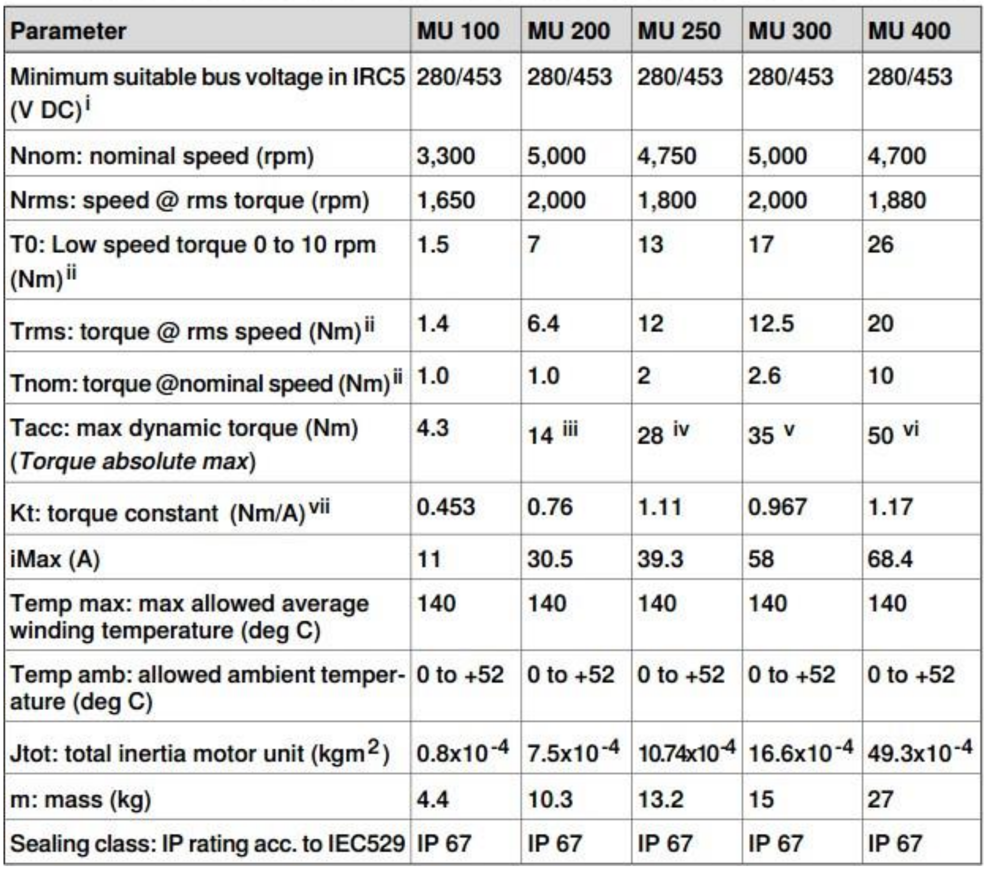


20. We can use the previous plot to estimate the minimum size of the motor that we need to move the joints. To achieve this motion profile, it seems that we need a motor that can provide a torque greater than 150 Nm for the first joint and 700 Nm for the second joint , but what about the angular velocity of the joint?
21. Add a scope to measure the velocity of the moving joint in rpm.
…………………………………………………………………………………………………
…………………………………………………………………………………………………
…………………………………………………………………………………………………
…………………………………………………………………………………………………
22. The scope should show a profile like this:



So it seems that we need to find a Servo motor that can supply a torque of at least 150Nm and with velocity more than 10 rpm. A Servo motor is just a DC motor placed in a closed position control loop with a controller. The following table is extracted from an ABB catalogue for DC motors, we need to select one of them which meets our specifications:

Table 1: Technical Specifications for ABB motor units, more details can be found in the mtors datasheet here: https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC040147-001&LanguageCode=en&DocumentPartId=&Action=Launch



Parameter	MU 100	MU 200	MU 250	MU 300	MU 400
Minimum suitable bus voltage in IRC5 (V DC)[i]	280/453	280/453	280/453	280/453	280/453
Nnom: nominal speed (rpm)	3,300	5,000	4,750	5,000	4,700
Nrms: speed @ rms torque (rpm)	1,650	2,000	1,800	2,000	1,880
T0: Low speed torque 0 to 10 rpm (Nm)[ii]	1.5	7	13	17	26
Trms: torque @ rms speed (Nm)[ii]	1.4	6.4	12	12.5	20
Tnom: torque @nominal speed (Nm)[ii]	1.0	1.0	2	2.6	10
Tacc: max dynamic torque (Nm) (*Torque absolute max*)	4.3	14 [iii]	28 [iv]	35 [v]	50 [vi]
Kt: torque constant (Nm/A)[vii]	0.453	0.76	1.11	0.967	1.17
iMax (A)	11	30.5	39.3	58	68.4
Temp max: max allowed average winding temperature (deg C)	140	140	140	140	140
Temp amb: allowed ambient temper-ature (deg C)	0 to +52	0 to +52	0 to +52	0 to +52	0 to +52
Jtot: total inertia motor unit (kgm^2)	$0.8x10^{-4}$	$7.5x10^{-4}$	$10.74x10^{-4}$	$16.6x10^{-4}$	$49.3x10^{-4}$
m: mass (kg)	4.4	10.3	13.2	15	27
Sealing class: IP rating acc. to IEC529	IP 67	IP 67	IP 67	IP 67	IP 67

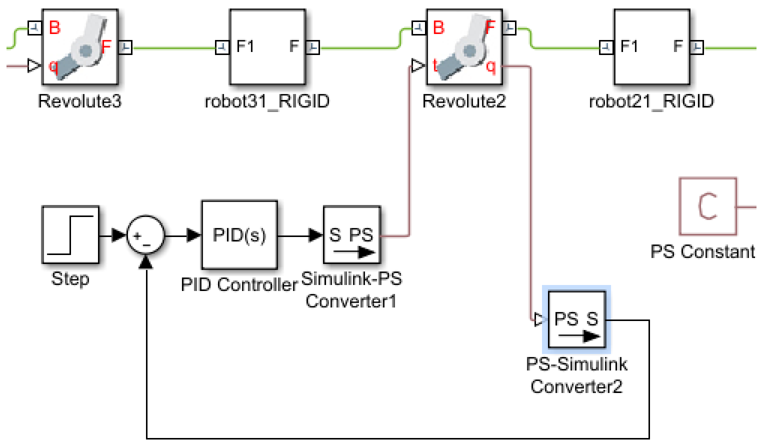
The nominal and speeds of all the motors are more than 100-250 times our target value, at the same time the maximum torque that they could provide is a lot less than our target value. A typical solution for this problem is to use a geared DC motor which has a speed reducing gearbox mounted on the motor. If we consider a gear ratio of 200, the motor (MU200) would be enough for our purpose (for the sinewave motion profile). It has stall torque (Low speed torque) of 7 Nm which if multiplied by our target gearbox ratio gives 1400 Nm.

23. The selected motor must also meet root mean square torque (Trms) of the target motion profile. To estimate the Trms, we need to export the torque profile of the first joint to MATLAB and use the function *rms* to estimate Trms. Use a **Sinks>To Workspace** block to export the torque of the first joint to a variable called T1, then use the *rms* function to estimate Trms. Does the MU200 meet the required rms Troque?:
…………………………………………………………………………………………………
…………………………………………………………………………………………………
…………………………………………………………………………………………………
24. As an initial trial we will assume that this motor is suitable for all the joints and we will use it to actuate the six joints of the robot.

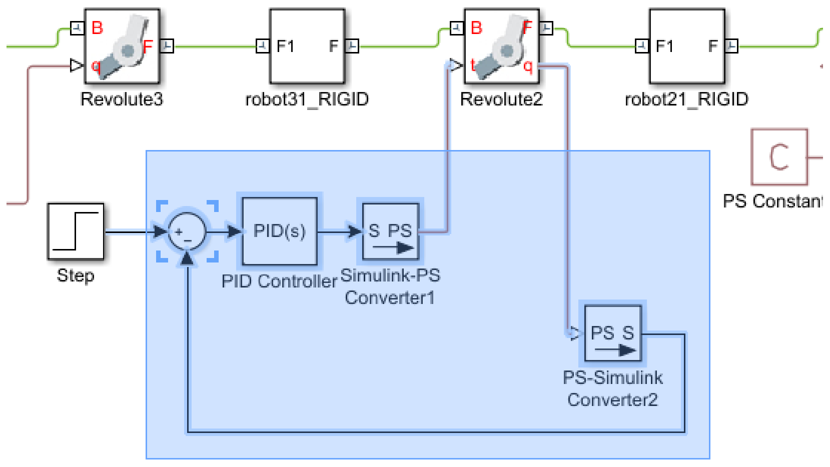
25. Don’t forget to save your model!

26. To include the selected geared DC motor in our model, we will start by constructing the closed loop control loop which is necessary to convert it to a Servo motor. For the first joint (named “**Revolute 2**”) under **Sensing** check the box next to **Position**. Under **Actuation** change the **Motion** to **Automatically Computed** and the **Torque** to **Provided by Input** and press OK. Note: This is the opposite to what we did in Step 7 because we are going to specify the torque input this time.

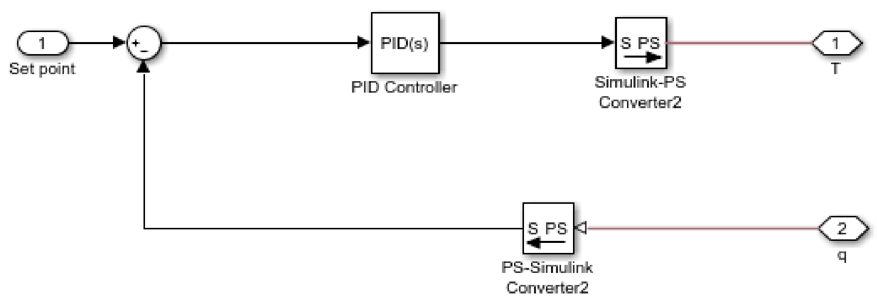
27. Construct the closed loop between the two newly created ports as shown below:



28. Leave the values of the newly added **Step** and **PID Controller** to their default values and run the simulation. Observe the third joint moves randomly; this is because we haven't tuned the controller. We will not tune the controller now, instead, note that we are applying the output of the controller directly as the torque input of the joints. In reality, this is done using our selected DC motor. In the next steps we will add the dynamics of the DC motor to our model.
29. First, we will create a subsystem for our Servo motor. Select the closed loop as shown below:



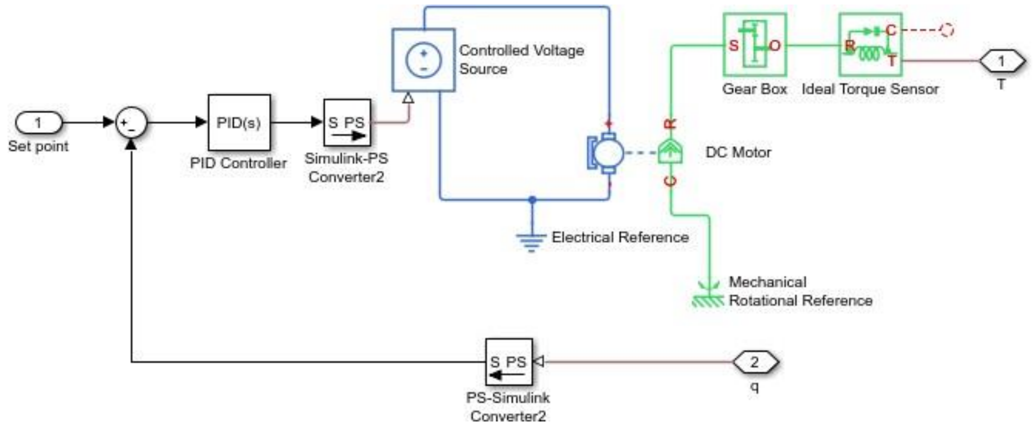
30. Right click on the selected components and select **Create Subsystem from selection.** Rename the created subsystem to **Servo** double click on it to open.
31. Reorganize the **Servo** subsystem and rename the input ports as shown in the figure below



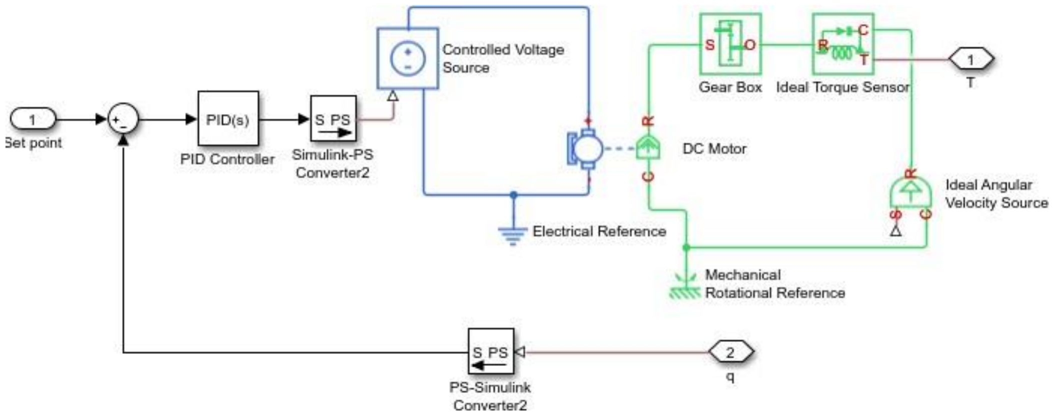
32. We now need to add the DC motor model between the controller and the torque output (**T**). Grab the following blocks into our subsystem:

Block	Location
DC Motor	Simscape>Electronics>Actuatos & Drives>Rotational Actuators
Controlled Voltage Source	Simscape>Foundation Library>Electrical>Electrical Sources
Electrical Reference	Simscape>Foundation Library>Electrical>Electrical Elements
Mechanical Rotational Reference	Simscape>Foundation Library>Mechanical> Rotational Elements
Ideal Torque Sensor	Simscape>Foundation Library>Mechanical>Mechanical Sensors
Gearbox	Simscape>Foundation Library>Mechanical>Mechanisms

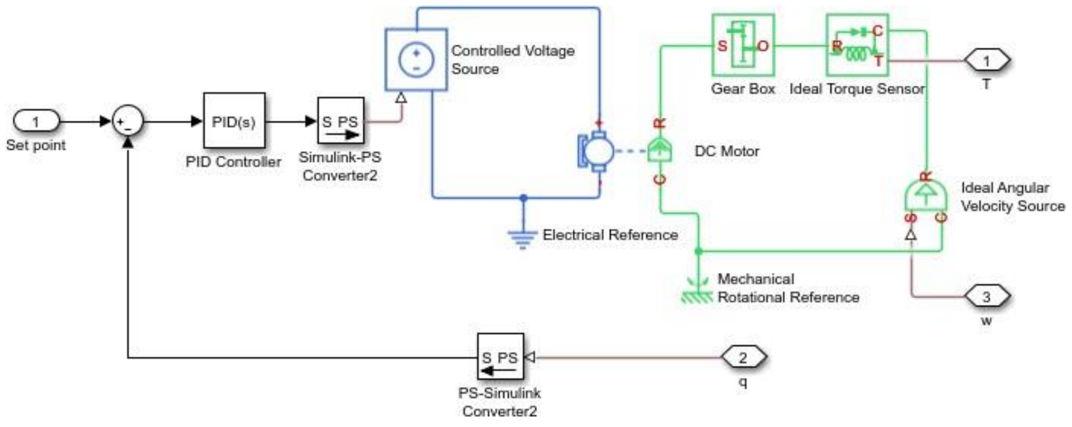
Try connecting these blocks to construct a model of the DC motor between the controller and the Torque port (T). The correct connections are shown in the next page.



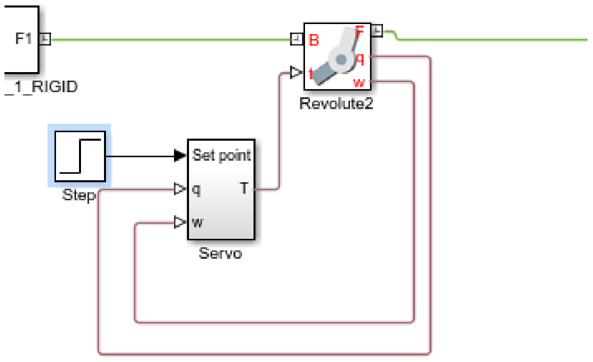
33. In the connection shown above the torque measured by the Ideal torque sensor (The Motor torque) is applied directly to the joint. This represents a one way coupling in which only the motor affects the robot mechanism, but the mechanism has no effect on the motor. To simulate the effect of the mechanism on the motor we will add an **Ideal Angular Velocity Source**. It is found under **Simscape>Foundation Library>Mechanical>Mechanical Sources** and connect it to our motor as shown below:



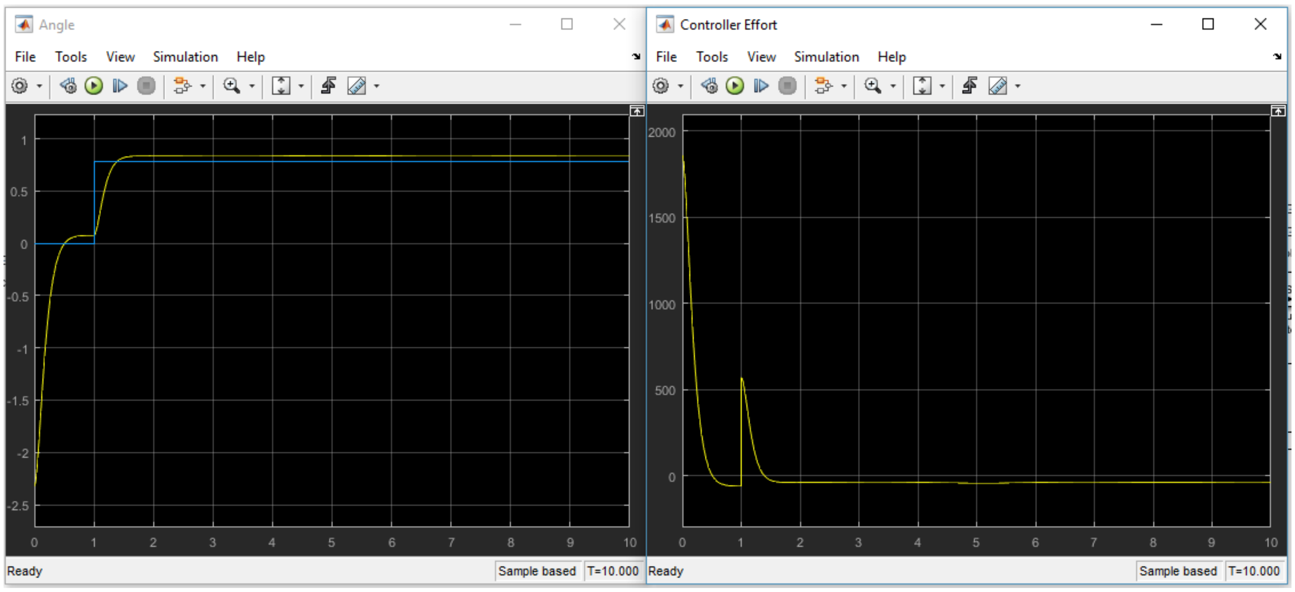
34. The velocity signal for the angular velocity source will come from the joint so we need first to create a **Connection port** for it in our subsystem you can find the physical **Connection Port** block under **Simscape>Utilities.** Rename it to w and connected to the Signal port of the velocity source:



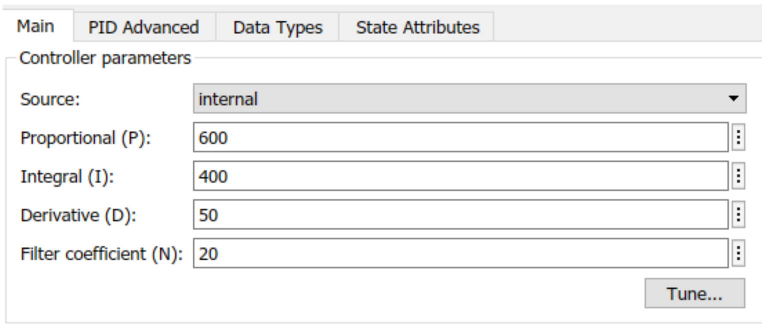
35. Exit the Servo Subsystem and double click on our target Revolute joint “**Revolute 2**” and under **Sensing** check the box next to **velocity.** Then connect the two “w” ports of the Servo and the revolute joint.



36. If you try to run the simulation now, you will notice that the third joint in the robot doesn’t go to the desired position. This is because:
    1. We didn’t tune the PID controller.
    2. We didn’t enter practical values for the DC motor properties or the gearbox.
37. Open the Servo subsystem and use the table in **step 22** to input the properties of the DC motor. We will use the MU 200 Model with a maximum voltage of 450 V. Also from the ABB datasheet, this model has a winding resistance of 15 ohm and inductance of 1.2 mH.
38. In the **PID Controller** block change the proportional gain (P) to 800 and the Integral gain (I) to zero.
39. Exit the Servo subsystem. Try running the model with a setpoint of pi/4 specified in Final value for the **Step** block.
    The system seems to stabilize to a certain point after oscillating for a while.
40. To have a closer look on the performance of our PID controller. Open the **Servo** block again and add two scopes named “Angle” and “Control Effort”. Connect them as shown below:



41. Three problems exist with our controller:
    1. From the left graph, the yellow curve representing the actual angle of the joint does not reach the blue curve representing the set point. The controller, thus has a large steady state error.
    2. The actual angle doesn't start from the set point of (0), instead it starts from -2.3 rad.
    3. The right graph represents the controller effort, which is the applied voltage on the DC motor in our system. At time 0, the voltage is 1700 volts which is more than the maximum voltage that our motor can handle (450 volts). In practice, that would burn the used motor.
42. You can address each problem by:
    1. Tuning the controller to at least include an integral action. Example tuning values are shown below:



Controller parameters

Source:	internal
Proportional (P):	600
Integral (I):	400
Derivative (D):	50
Filter coefficient (N):	20

    2. Limit the output of the controller so that it doesn't exceed ± 450 V. This can be done from the **PID Advanced** tab of the PID controller.
    3. Specify the initial position of the joint to be zero, this can be done by double clicking the joint "**Revolute 2**" and set **State Targets>Set Position Target>Value** to zero degrees. You may want to give some time for the controller output to settle before applying the step input. This can be done by changing the value of the **step time** in the Step block.
43. We have used the first joint torque as a method to select our motor; however, motors are usually selected for the required motor torque and not the joint torque as they might differ. Lets check that the required motor torque can be satisfied by our motor. Add a **Torque Sensor** to measure the rms and peak torque of the DC motor in the **Servo** block. Can the M200 motor supply the actual torque needed to move this joint with the selected profile?

    ………………………………………………………………………………………………

    ………………………………………………………………………………………………