Multibody dynamic simulation with Simscape: methods and examples

LI KENAN

255250

Tutor: Prof. TRIVELLA ANDREA

Contents

1.Introduction

With the increasing complicity of control systems and the rapid development of robotics technology, various complex mechanical systems have emerged, such as vehicles, spacecraft, robotic arms, robots, and human science. Due to the continuous improvement of the degree of freedom of the system, more and more complex system cannot be modeled by a single model. Therefore, it has a great significance to carry out research on modeling and simulation of multibody systems. On the other hand, due to the increasing complexity of modern precise control systems and more and more components inside the system, the probability of failure of the controlled system is increasing. In a large-scale complex system, once an accident occurs in a certain subsystem, it may cause significant property losses.

In 1977, the MultiBody Dynamics Symposium was first held in Munich, Germany. Since then, international conferences on multi-body dynamics have emerged in endlessly. In recent years, researchers have done deeply research on the modeling and simulation of multibody systems.

Multibody dynamics simulation (MBDS) software decomposes the actual system into rigid bodies, joints, constraints, coordinate systems, drives, sensors, inputs and other components. A complex system can be modelled with graphical components. Also, the entire simulation process contains two stages: modeling and solving. Modeling includes two processes:

First, physically modelling, form a physical model from a geometric model.
Second, mathematically modelling, form a mathematical model from a physical model.

The solver is a set of computation algorithms that solve equations of motion. The choose of solver depends on the type of problem, kinematics/dynamics, static balance, eigenvalue analysis, etc. The corresponding solver should be well selected for numerical calculation and solution.

It is suited to study the dynamic behavior of interconnected rigid and/or flexible bodies undergoing large translational or rotational displacements. The motion of those bodies is calculated based on applied loads and boundary conditions defined.

Multibody dynamics simulation (MBDS) can not only be used to predict mechanical system performance, trajectory of motion, collision detection, peak load and so on but also used in several aspects, such as safety, and comfort, can be evaluated and optimized. Users can not only use general modules to simulate simple mechanical systems, but also use specific modules to quickly and effectively model and simulate problems in related industrial applications.

The potential of MBDS in product development mainly according to following reasons:

1. Reduction of development lead time and life cycle cost.

   Manufacturers often struggle to understand real system performance until extremely late in the design process. Mechanical, electrical, and other subsystems are validated against their specific requirements within the systems engineering process, but full system test and validation comes late, leading to rework and design changes that are riskier and more costly than those made early on.

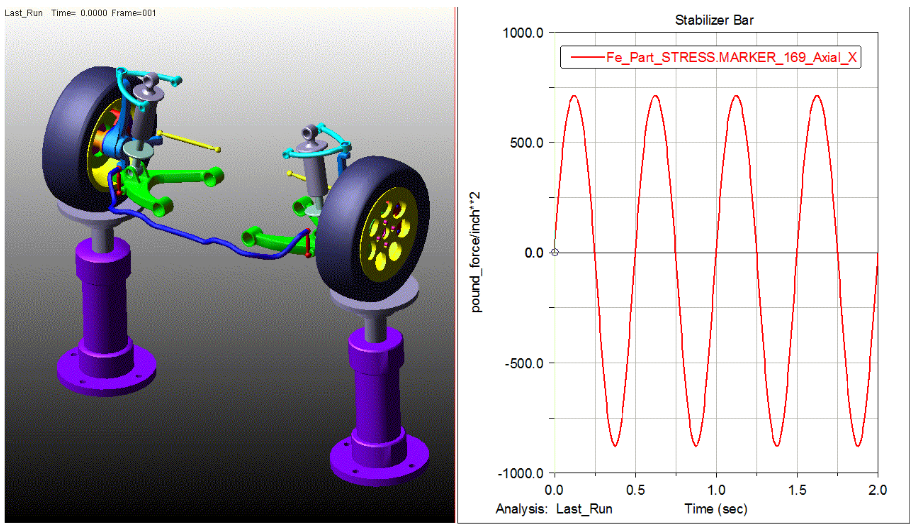


Figure 1 Simulation of anti-roll bar under roll motion

   MBS improves engineering efficiency and reduces product development costs by enabling early system-level design validation. Engineers can evaluate and manage the complex interactions between disciplines including motion, structures, actuation, and controls to better optimize product designs for performance, safety, and comfort. Along with extensive analysis capabilities, MBS is optimized for large-scale problems, taking advantage of high-performance computing environments.

   MBS offer comparably short calculation times. This makes them a very efficient simulation tool and an excellent choice for parameter studies and optimization of complex assemblies having many degrees of freedom.

   Due to their short calculation times, MBS enable early insights in the effect of design parameters on the overall performance of a system. This allows to make well-founded design choices in an early state of the development process, saving costs by lowering lead times and reducing the number of extended hardware tests significantly.

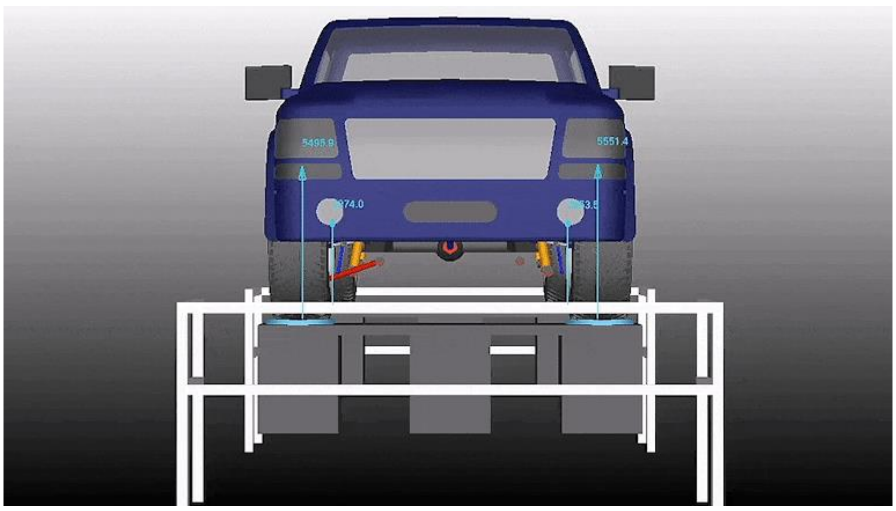


Figure 2 Force act at tire ground contact

To give an example, with Adams Car, engineers can quickly build and test functional virtual prototypes of complete vehicles and vehicle subsystems. Working in the Adams vehicle vertical environment, automotive engineering teams can exercise their vehicle designs under various road conditions, performing the same tests they normally run in a test lab or on a test track, but in a fraction of time. Following requests can be satisfied：

- Analysis of suspension, steering and full-vehicle maneuvers
- Easy integration of control systems into vehicle models
- Creation or import of component geometry in wireframe or 3D solids
- Extensive library of joints and constraints to define part connectivity
- Model refinement with part flexibility, automatic control systems, joint friction and slip, hydraulic and pneumatic actuators, and parametric design relationships
- Comprehensive linear and nonlinear results for complex, large-motion designs
- Comprehensive and easy to use contact capabilities supporting 2D and 3D contact between any combination of modal flexible bodies and rigid body geometry

2. Easy to use.

The complexity of the procedure for using the MBS is reduced compared with the past decade. As the MBS are well developed, it is now, not any more designed for specialist in simulation or expert in dynamic analysis, but also designed for normal design engineer.

In MATLAB® Simscape Multibody program a complex multibody system can be built by blocks representing bodies, joints, constraints, force elements, and sensors. It formulates and solves the equations of motion for the complete mechanical system and automatically generated 3D animation lets you visualize the system dynamics.

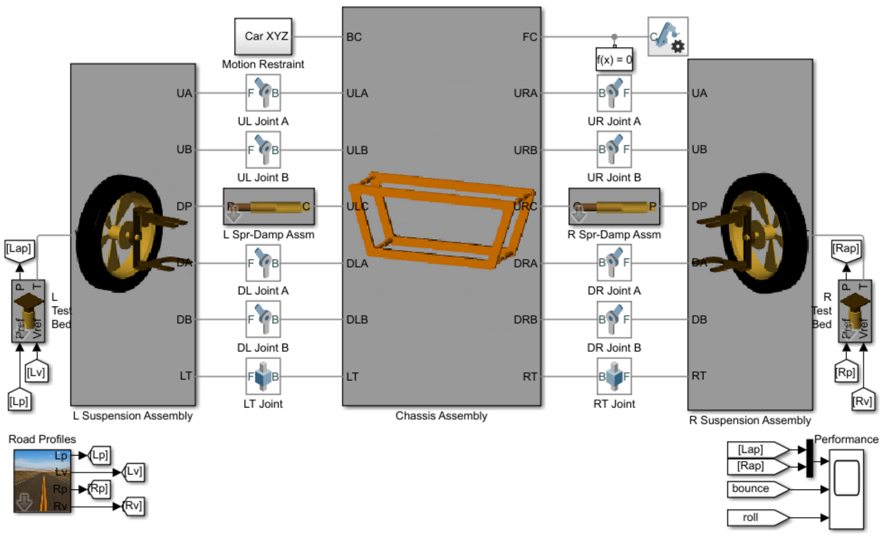


Figure 3 In Simscape Multibody, front suspension is modeled by blocks

Simscape Multibody contains a library of blocks and simulation and control interfaces to interconnect Simscape plans with the Simulink environment, also hydraulic, electrical, pneumatic, and other physical systems can be integrated into model using components from the Simscape family of products.

RecurDyn has an immensely powerful GUI with an intuitive design. It contains a completely integrated environment for model development, simulation, and analysis of results. Pre-processing for model creation and parameter definition and post-processing for analyzing the results is integrated directly into the GUI. A wide range of customization options also exist in the GUI to improve productivity by automating common tasks.

3. Accurate, or in other world, MBDS software can simulate the “real world” physics.

To simulate the dynamic behavior of a rigid multibody, it is necessary to solve a sequence of motion of equation which describe the motion of system. These equations are known as Differential Algebraic Equations (DAEs), a combination of differential equations that describe motion and algebraic equations that encapsulate joint constraints. For linear dynamic simulation, solver has high efficiency and accurate due to the multi-core processor nowadays.

Utilizing multibody dynamics solution technology, MBDS such as ADAMS runs nonlinear dynamics in a fraction of the time required by FEA solutions. Loads and forces computed by simulations improve the accuracy of FEA by providing better assessment of how they vary throughout a full range of motion and operating environments.

4. Repeatable. In real life, failure test costs time and money while using MBDS software, it is possible to simulate and validate repeatably under fault condition. We have certain benefits:

   Create Robust Designs. Specify failure criteria for components, including time, load, or temperature-based conditions. Model degraded component behavior, such as worn gear teeth or increased bearing friction. Automatically configure models to efficiently validate designs under fault conditions.

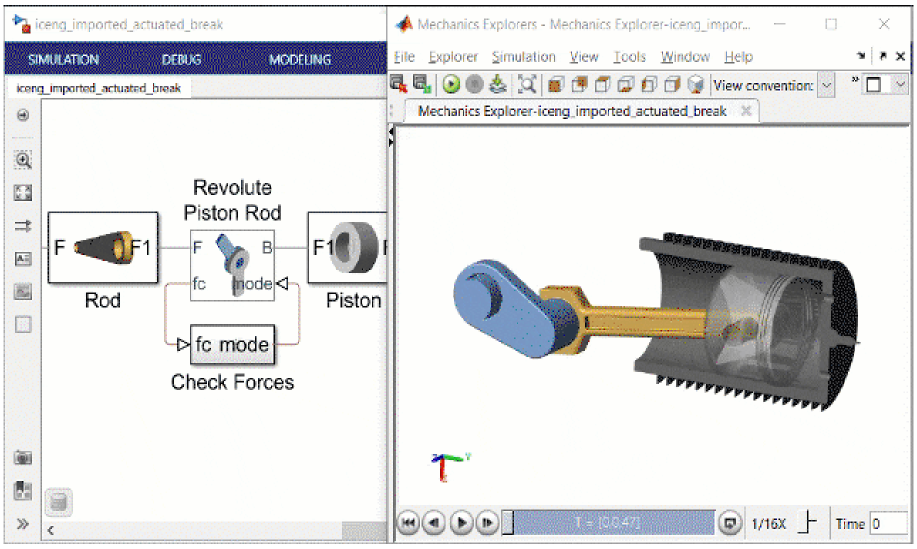

iceng_imported_actuated_break
SIMULATION
DEBUG
MODELING
Revolute
Piston Rod
F
F1
Rod
B
fc
mode
fc mode
Check Forces
Piston
Mechanics Explorers - Mechanics Explorer-iceng_impor...
File Explorer Simulation View Tools Window Help
View convention:
Mechanics Explorer-iceng_imported_actuated_break
1/16X
Time 0

Figure 4

A connection between two parts breaks as the force exceeds the upper limit for the joint.

Perform Predictive Maintenance. Generate data to train predictive maintenance algorithms. Validate algorithms using virtual testing under common and rare scenarios. Reduce downtime and equipment costs by ensuring maintenance is performed at just the right intervals.

Minimize Losses. Calculate the power dissipated by mechanical components. Verify components are operating within their safe operating area. Simulate specific events and sets of test scenarios, and then post-process results in MATLAB.

Multibody dynamic simulation software makes change in design stage much faster and at a lower cost than physical prototype testing would require. Provide more secure environment without the fear of losing data from instrument failure. Without physical testing, simulation or the analysis can be performed in all scenarios without the dangers.

2.Method and examples

In this chapter, serval examples that modeled and simulated based on Simscape Multibody are going to be shown. Examples are in the field of automotive engineering.

2.1 Mass-Spring-Damper model

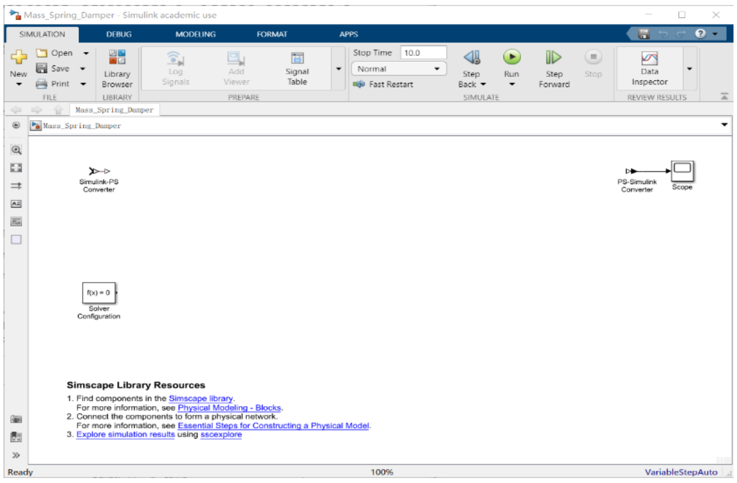
Fist, we start with a classic model constructed with Matlab Simscape.

Mass-Spring-Damper system is a quite common oscillatory model. By studying the model, we can have a better understanding of the object with complex mechanical properties such as nonlinearity. In this model, mass is considered as node which is connected with ideal weightless damper and spring representing the different mechanical properties. Also, by changing the characteristic of damper and spring to achieve desired effects. After building the model, applying the Newton's Law to the mass considering not only the force applied by damper and spring but also the external force gives differential equations for the motion of the mass. The differential equations are solved by standard numerical schemes for solving ODEs, in Simscape we have servals ODEs can be use, the ODE used in the examples are ode45.

The workflow of constructing a Mass-Spring-Damper is shown as below:

1. To build a Simscape model with default setting, use the command *ssc_new.*

   When command in used, following object will automatically be created: data logging, recommended solver and tolerance settings.



Simscape Library Resources

1. Find components in the Simscape library.
   For more information, see Physical Modeling - Blocks.
2. Connect the components to form a physical network.
   For more information, see Essential Steps for Constructing a Physical Model.
3. Explore simulation results using sscexplore

Figure 2.1.1 Create a Simscape model

2. Assemble Physical Network

   To model the system, blocks can be added from the Simscape Libraries, then connect the block through lines, into a physical network. The physical network built in the Simscape represent the real system, in the other words, Simscape mimic the physical system. When constructing the network, it is necessary to include the domain-specific reference blocks, such as Mechanical Translation Reference in this example.

The detailed instruction is described as following:

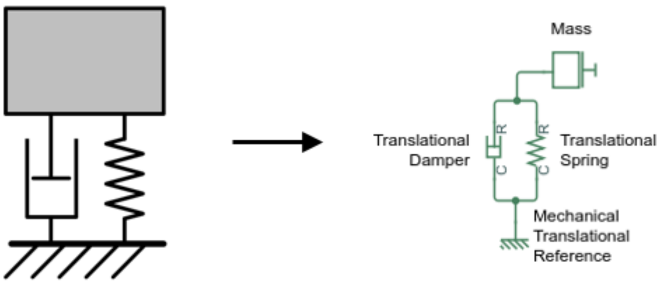


Figure 2.1.2Model represents the real system

2.1 Click Simscape - Foundation Library - Mechanical - Translational Elements library.

2.2 Drag Mass, Translational Spring, Translational Damper, and Mechanical Translational Reference blocks into the window. Then connect them

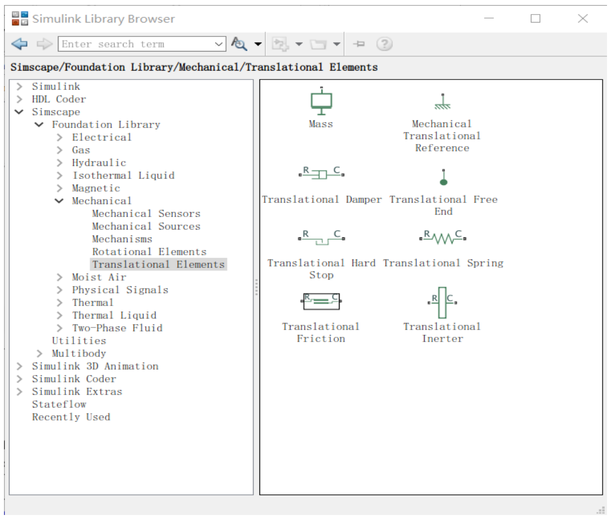


Figure 2.1.3 Add blocks

through lines.

To adjust or shorten the name of blocks, right click the block, select Format - Show Block Name - on

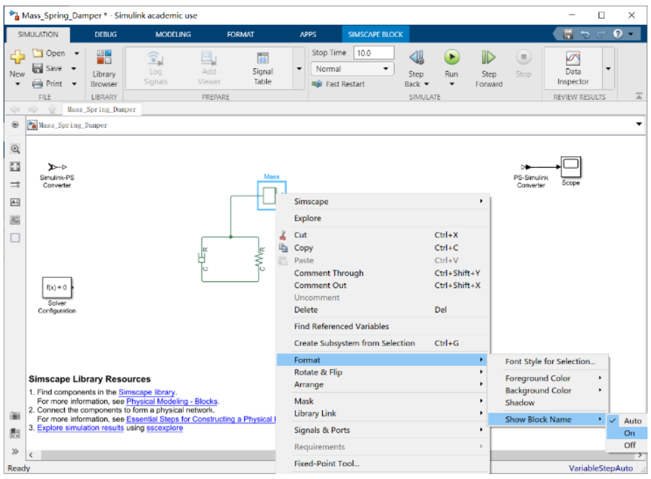
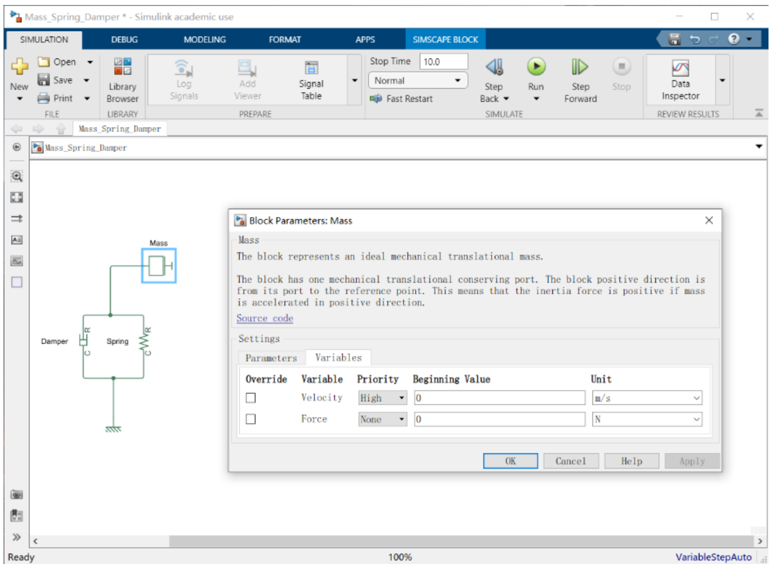


Figure 2.1.4 Show block name

3. Adjust Block Parameter

Each block represents a generic component that has initial values. In this model, mass, spring and damper block can be adjusted by double click the block. As an example, spring rate, damping coefficient and mass are adjust to 400N/m, 100N/(m/s) and 3.6kg. To specify the desired initial value for mass velocity, click **Variables.**



Block Parameters: Mass

Mass

The block represents an ideal mechanical translational mass.

The block has one mechanical translational conserving port. The block positive direction is from its port to the reference point. This means that the inertia force is positive if mass is accelerated in positive direction.

Source code

Settings

Parameters | Variables

Override	Variable	Priority	Beginning Value	Unit
☐	Velocity	High	0	m/s
☐	Force	None	0	N

OK Cancel Help Apply

Figure 2.1.5 Adjust Block Parameter

Notice that override means that the velocity has overridden the default variable initialization value. The variable with high priority, therefore, the solver will try to satisfy this initial value when it computes the initial conditions during the simulation.

4. Add Sources

Simscape source blocks represent physical effect, such as forces, voltages, or pressures, that act on the system. Also, quantities that flow through the system such as current, mass flow rate, and heat flux can be specified. To add representation of the force acting on the mass, use the Ideal Force Source block. The block represents an ideal source of force that generates force proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified force regardless of the velocity at source terminals.

The detailed instruction is described as following:

4.1 Simscape -- Foundation Library -- Mechanical -- Mechanical Sources library. Add the Ideal Force Source block to the diagram

4.2 To reflect the correct direction of the force shown in the original schematic, flip the block orientation. Select the Force Source block, on the Format tab, under Arrange, click Flip up-down

4.3 The system must have 2 constraints, so add another Mechanical Translational Reference block by right click it and drag it to a new location.

4.4 Connect port C of the Force Source block to this second Mechanical Translational Reference block and port R to the Mass block. The input signal stands for the force profile will be supplied through port S, after the system is connected to a Simulink source. A positive signal at port S will specify a force that acts from port C to R.

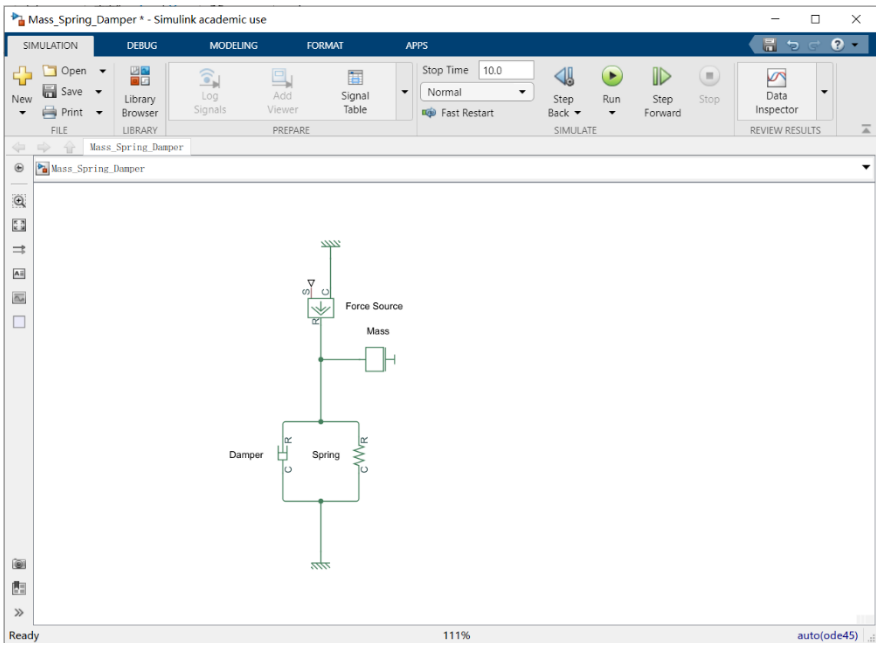


Figure 2.1.6 Add Source

5. Add sensors

Quantities from physical network can be measured by adding sensors block, it can be in series or parallel depends on the measured value. To measure a quantity defined by a Through variable (such as current, flow rate, force), connect the sensor in series. To measure a quantity defined by an Across variable (such as voltage, pressure, velocity), connect the sensor in parallel. In this example, spring deformation is an important parameter. To measure spring deformation, connect an Ideal Translational Motion Sensor block in parallel with the spring.

It is a device that converts an across variable measured between two mechanical translational nodes into a control signal proportional to velocity and position. The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

Also, some common uses of those quantities include feedback for a control algorithm, modeling physical components whose behavior depends on other physical quantities (such as temperature-dependent resistor), or simply viewing the results during simulation.

The detailed instruction is described as following:

5.1 Simscape -- Foundation Library -- Mechanical -- Mechanical Sensors library. Add the Ideal Translational Motion Sensor block to the diagram, for a better reading experience, shorten the block name as Motion Sensor.
5.2 For Ideal Motion Sensor, connect the R port and port as the figure shown below. Connections R and C are mechanical translational conserving ports and connections V and P are physical signal output ports for velocity and position, respectively. The block positive direction is from port R to port C.

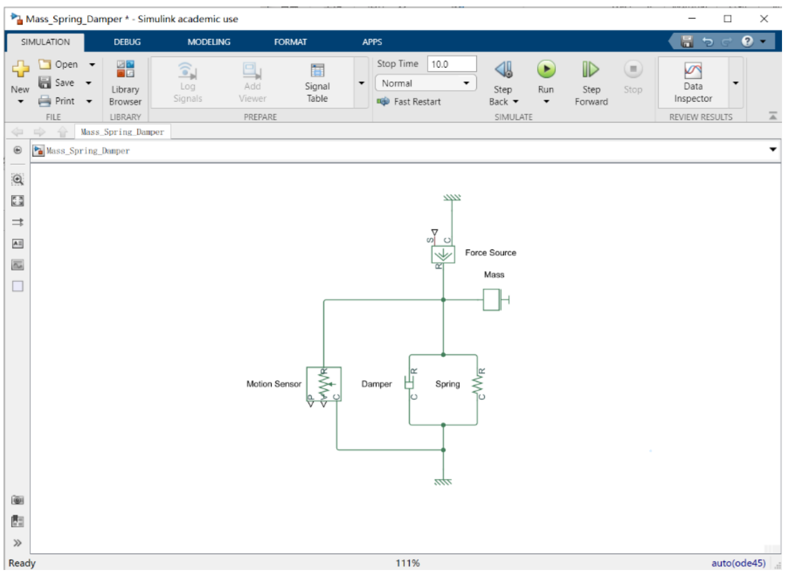


Figure 2.1.7 Add Sensors

6. Connect to Simulink with Interface Blocks

Interface blocks, such as Simulink-PS Converter and PS-Simulink Converter, takes in charge the signal conventions between these two modeling. Interface blocks are needed when Simulink signals specify quantities in a Simscape network, or when passing Simscape quantities to Simulink for control design. Every time when connect a Simulink block to a Simscape physical network, an appropriate converter block is needed.

Now, to connect the physical network to a controller built out of regular Simulink blocks. First, prepare the physical network to be connected to Simulink signals then, build and connect the controller:

The detailed instruction is described as following:

6.1 Connect the physical signal output port of the Simulink-PS Converter block to port S of the Force Source block. This signal is the input signal of the force source.

6.2 Connect the output port P of the Motion Sensor block to the physical signal input port of the PS-Simulink Converter block. As motioned before, the port P stands for the position.

6.3 Connect the Solver Configuration block to the circuit.

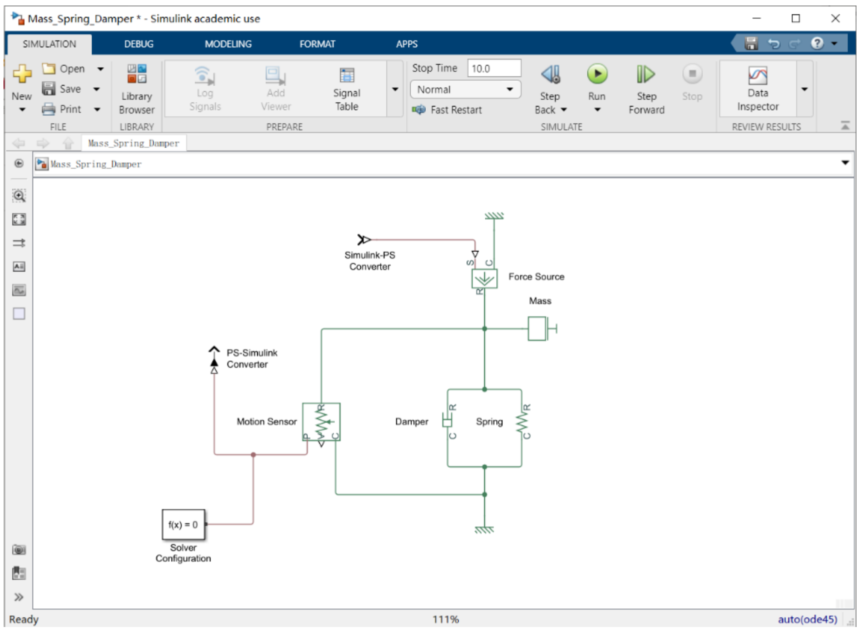
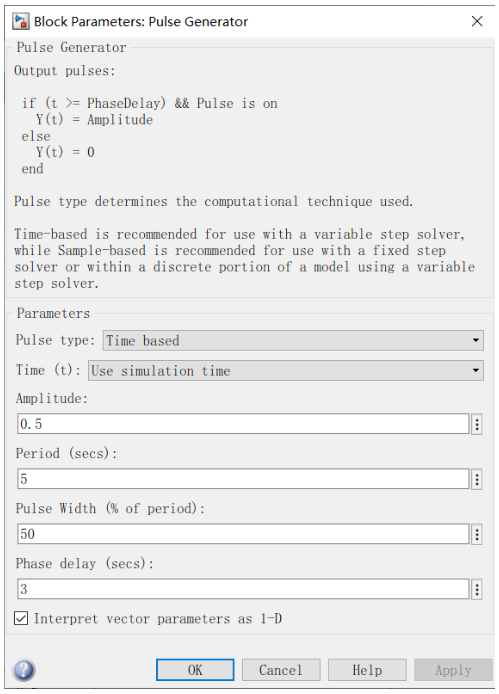


Figure 2.1.8 Physical Signal side

6.4 Open Simulink -- Sources library and drag the Pulse Generator block into the model. Set the parameter to the desired value.


Block Parameters: Pulse Generator

Pulse Generator

Output pulses:

```
 if (t >= PhaseDelay) && Pulse is on
   Y(t) = Amplitude
 else
   Y(t) = 0
 end
```

Pulse type determines the computational technique used.

Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.

Parameters

Pulse type: Time based

Time (t): Use simulation time

Amplitude: 0.5

Period (secs): 5

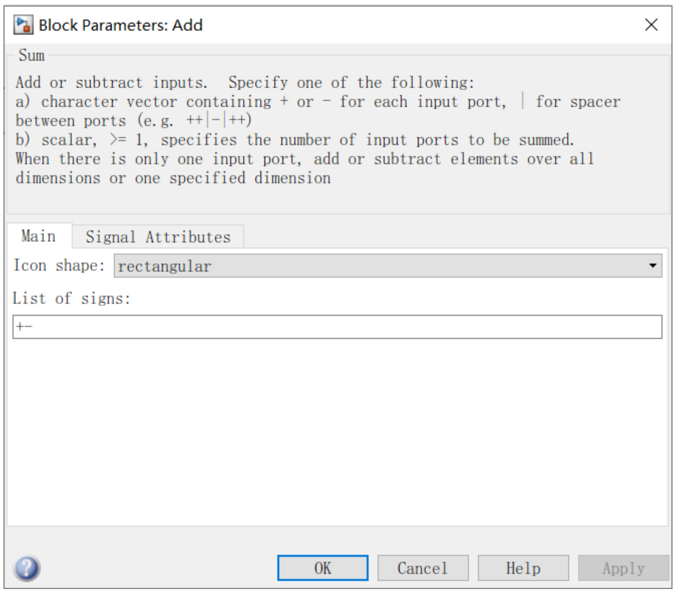
Pulse Width (% of period): 50

Phase delay (secs): 3

☑ Interpret vector parameters as 1-D

Figure 2.1.9 Position command

6.5 Open Simulink -- Math Operations library and drag the Add block as feedback into the model. Set the correct signs



Block Parameters: Add

Sum

Add or subtract inputs. Specify one of the following:
a) character vector containing + or − for each input port, | for spacer between ports (e.g. ++|−|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

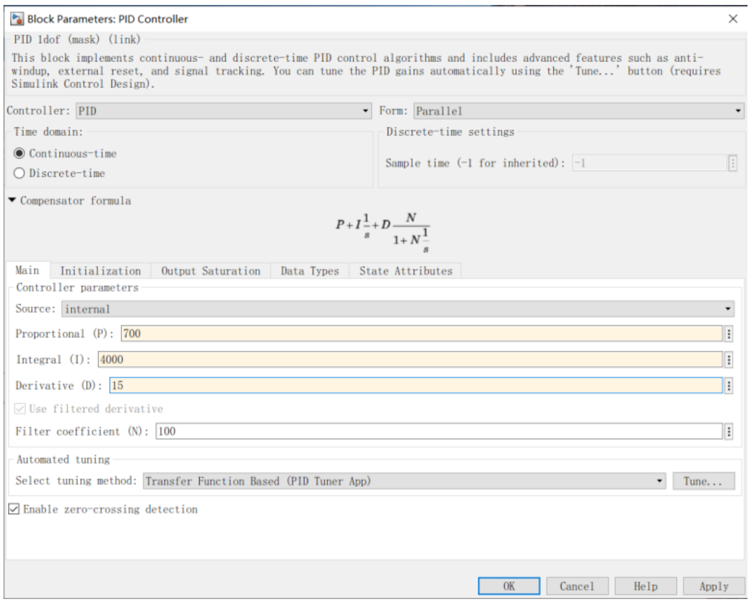
Main | Signal Attributes

Icon shape: rectangular

List of signs: +−

Figure 2.1.10 Feedback

6.6 Open the Simulink -- Continuous library and drag the PID Controller block into the model. Set the Proportional (P), Integral (I), and Derivative (D) parameter with desired value



Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID   Form: Parallel

Time domain: [✓ Continuous-time] Discrete-time

Discrete-time settings: Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Main | Initialization | Output Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 700

Integral (I): 4000

Derivative (D): 15

☑ Use filtered derivative

Filter coefficient (N): 100

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

☑ Enable zero-crossing detection

Figure 2.1.11 PID controller

6.7 Connect the block as following figure.

The control Simulink signal goes to the input port of the Simulink-PS Converter block, where it is converted into a physical signal driving the force profile of the Ideal Force Source block. The output port P of the Ideal Translational Motion Sensor block, which measures the spring deformation, connects to the PS-Simulink Converter block. This block converts the physical signal into a feedback Simulink signal for the controller.

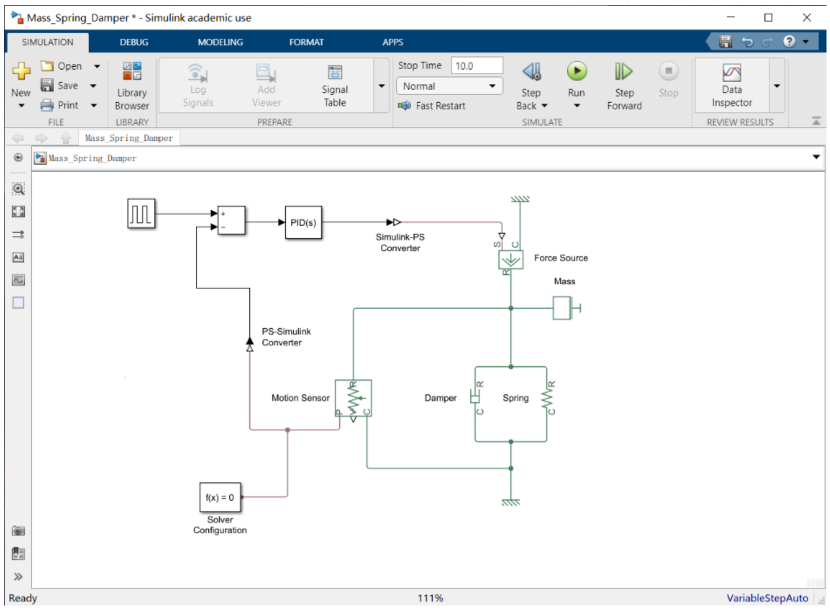


Figure 2.1.12 Connect to Simulink with Interface Blocks

7. Simulate Model

To compare the input and feedback signals, connect them to a signal viewer:

7. 1 Right-click the Simulink signal that goes from the Pulse Generator block to the Sum block. From the menu, select Create & Connect Viewer – Simulink -- Scope.

7.2 Then, right-click the Simulink signal that goes from the PS-Simulink Converter block to the Sum block. From the context menu, select Connect to Viewer -- Scope.

7.3 Click the run button in Simulink Toolstrip or in Scope. The Simscape solver evaluates the model and runs the simulation. Figure 2.1.13 shows the comparison of input and feedback signals, the yellow square wave is the input signal while the blue line with a little overshot is the feedback signal.

8. View Simulation Results

The other way to view the results is to use Simscape Results Explorer. It analyzes simulation data by using data logging functionality. Right click the block (component) select Simscape -- View simulation – simlog.

The Simscape Results Explorer window opens, with the node corresponding to the Spring block highlighted in the left pane. The right pane displays the simulation data plots for the variables associated with the block shown as Figure 2.1.14.

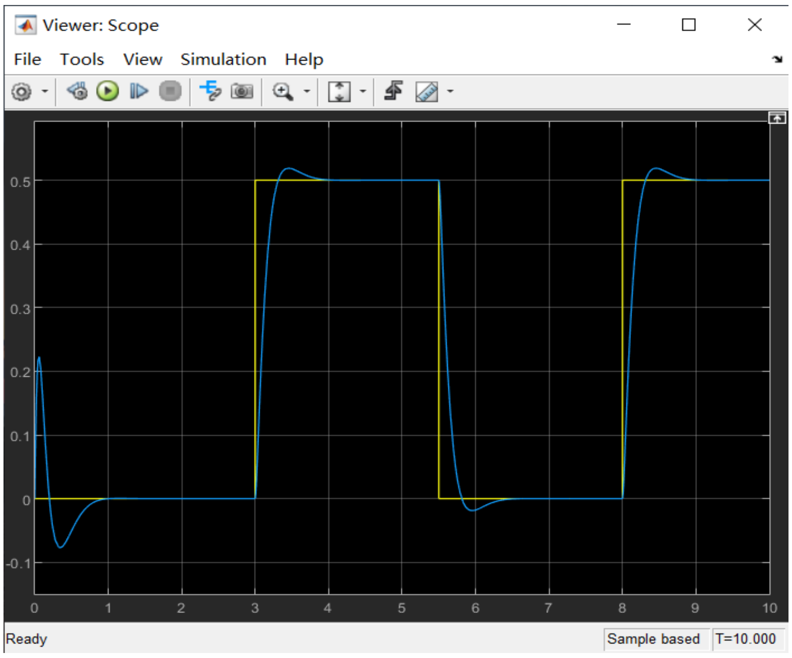
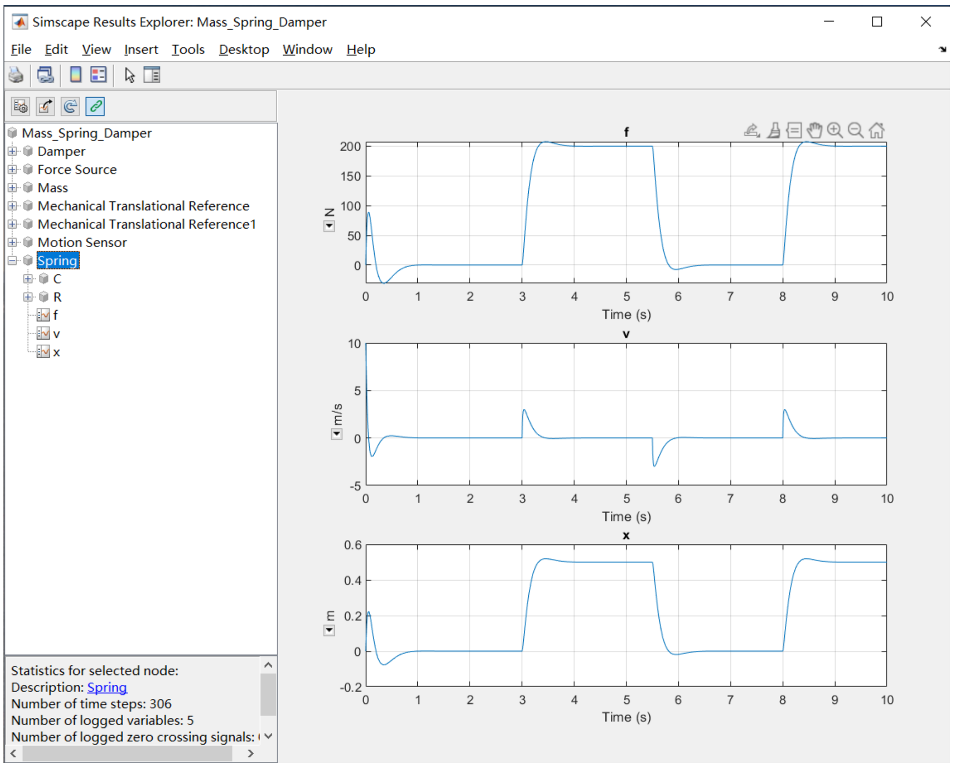


Figure 2.1.13 Compare the input and the feedback signals


Simscape Results Explorer: Mass_Spring_Damper
File Edit View Insert Tools Desktop Window Help
Mass_Spring_Damper
Damper
Force Source
Mass
Mechanical Translational Reference
Mechanical Translational Reference1
Motion Sensor
Spring
C
R
f
v
x
f
N
v
m/s
x
m
Time (s)

Statistics for selected node:
Description: Spring
Number of time steps: 306
Number of logged variables: 5
Number of logged zero crossing signals:

Figure 2.1.14 Simulation Results of Spring

2.2 Pendulum model

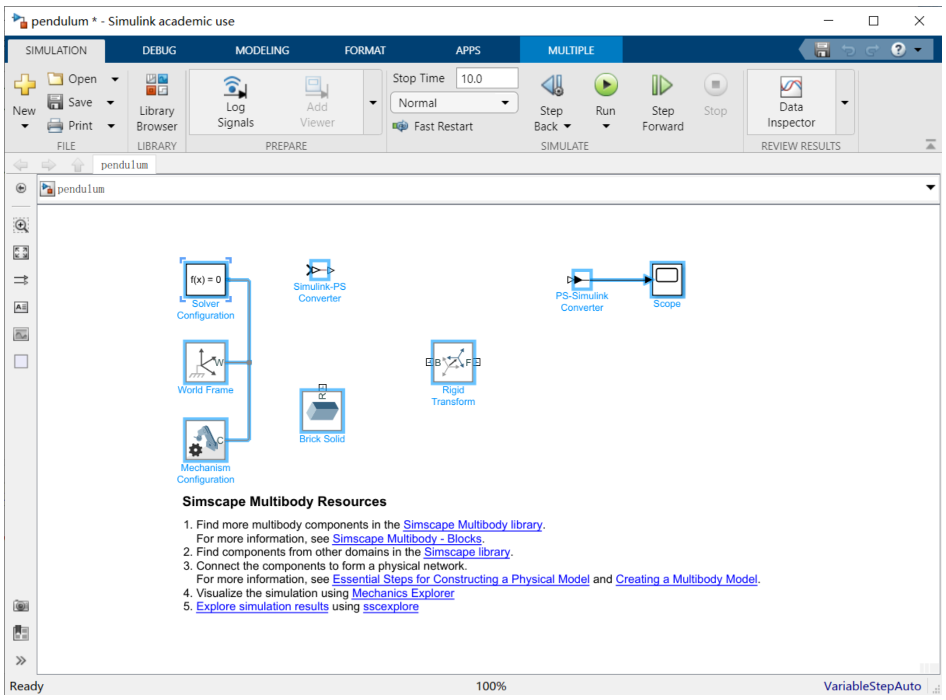
In this section, a simple pendulum is modeled and analyzed using Matlab Simscape Multibody. The pendulum is a simple mechanical system. For this example, the system contains two bodies, a link and a fixed pivot, connected by a revolute joint.

1. Model a simple link

Mechanical links are common building blocks in linkages, mechanisms, and machines. To model a pendulum, a simple link should be modeled first. For simplicity, the model assumes the link has a brick shape and two end frames.

1.1 Create a model template

To open a Simscape Multibody model template with default setting use command *smnew*



Simscape Multibody Resources

1. Find more multibody components in the Simscape Multibody library.
   For more information, see Simscape Multibody - Blocks.
2. Find components from other domains in the Simscape library.
3. Connect the components to form a physical network.
   For more information, see Essential Steps for Constructing a Physical Model and Creating a Multibody Model.
4. Visualize the simulation using Mechanics Explorer
5. Explore simulation results using sscexplore

Figure 2.2.1 Model template

1.2 Duplicate the Rigid Transform block to add a second end frame to the link.

Rigid Transform block defines a fixed 3-D rigid transformation between two frames. Two components independently specify the translational and rotational parts of the transformation. Different translations and rotations can be freely combined.

In the expandable nodes under Properties, choose the type and parameters of the two transformation components. We can use the Cartesian, standard axis and cylindrical as translation axis, each parameter can be modified under properties tab. We can specify the following methods of translation:

Standard axis and elementary translation along one base axis

Cartesian, three-dimensional translation, as a vector XYZ.

Cylindrical, three-dimensional translation, as a vector r theta and Z

For rotation, we have different choice such as standard axis, aligned axes, arbitrary axis, rotation sequence and rotation matrix. Rotate around standard axis is defined with the right-hand rule, the axis of rotation is defined positive in the direction of the thumb.

Standard axis: where the elementary rotation is along one base axis.

Arbitrary axis: where the elementary rotation is along one custom axis.

Aligned Axe: where we have 2 axis mappings between base and follower coordinate frames.

Rotation matrix: where we define a 3x3 rotation matrix from a base to follower frame.

Rotation sequence: where we specify frame rotation as a sequence of three elementary rotation angles and axis.

Ports B and F are frame ports that represent the base and follower frames, respectively. The transformation represents the follower frame origin and axis orientation in the base frame.

These parameters specify the locations of two end frames on the simple link.

Parameter	Rigid Transform1	Rigid Transform	Units
Translation -- Method	Standard Axis	Standard Axis	Not applicable
Translation -- Axis	-X	+X	Not applicable
Translation -- Offset	L/2	L/2	cm

Table 2.2.1 Parameters in Rigid Transform blocks

1.3 Flip the Rigid Transform1 block so that you can connect the B ports of two Rigid Transform blocks to each other and the Brick Solid block. And connect the remaining blocks.

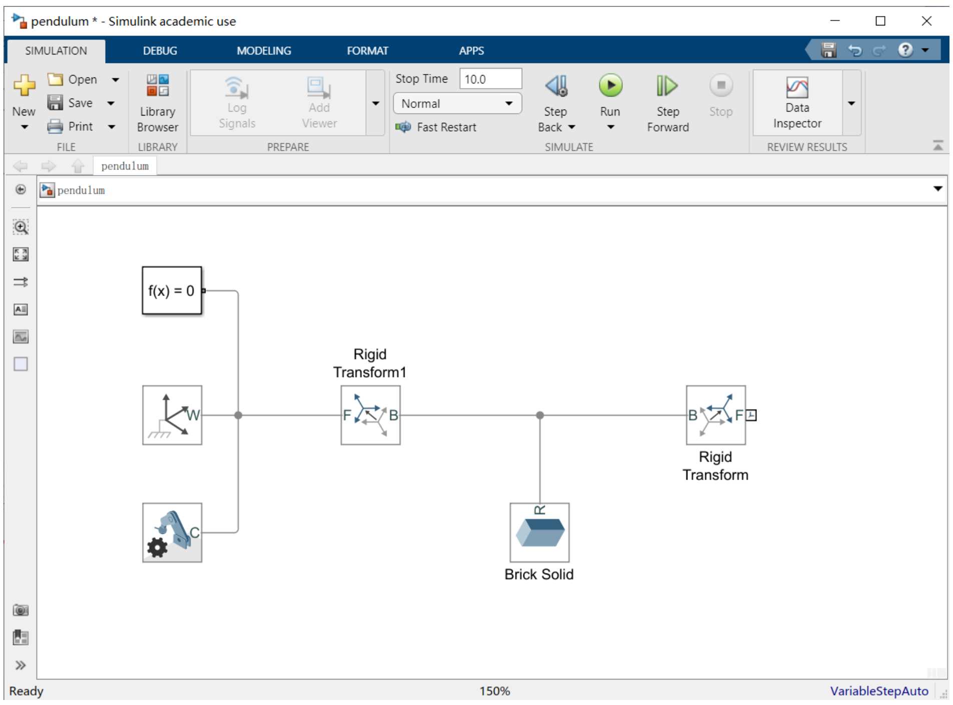


Figure 2.2.2 Connect blocks

1.4 Define link's physical properties, such as shape, mass and appearance in Brick Solid block.

Brick Solid box represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

After changing the parameters, the Brick Solid and Rigid Transform blocks will be highlighted in red because variables have not defined yet. This issue will be solved after imputing all the numerical values for the parameters, which will be described in the following section. Instead inserting the numerical numbers, we insert variables which is much more convenient if we have to modify some parameters.

Parameter	Value	Units
Geometry > Dimensions	[L W H]	cm
Inertia > Density	rho	kg/cm^3
Graphic--Visual Properties-- Color	rgb	Not applicable

Table 2.2.2 Parameters in Brick Solid Box

1.5 Generate Subsystem

A complex multibody system can be build using several simple models, such as a simple link model. The physical parameters of these simple models usually need to be adjusted to fit different design requirements. To simplify the parameter adjusting process, Subsystem blocks is used for these simple models. A Subsystem block enables the updates for many parameters in a single place — the Subsystem block dialog box. In this section, it is exampled how to create a Subsystem block for the simple link model.

Select the Brick Solid block and the two Rigid Transform blocks by

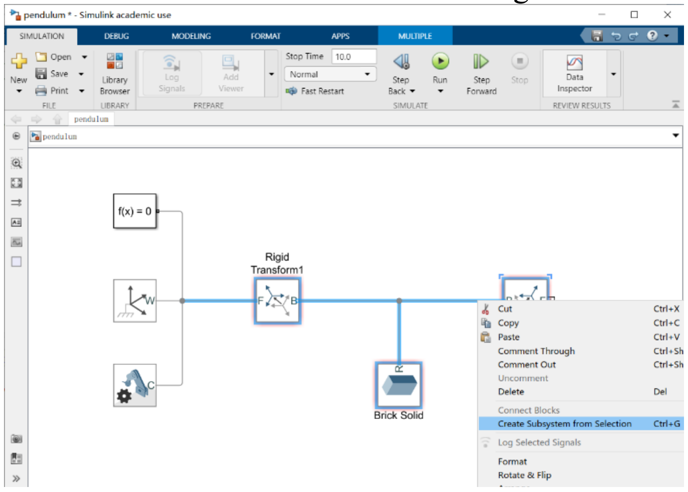


Figure 2.2.3 Create Subsystem

holding shift and clicking the blocks.

Right-click one of the selected blocks and select Create Subsystem from Selection. Simulink adds a new Subsystem block that contains the Brick Solid and Rigid Transform blocks.

Double-click the Subsystem box. A new tab displays the children blocks of the Subsystem block. Double-click the PMC_port on the left named Rigid Transform2, which is the Physical Modelling Connection Port block for subsystem, set Port location on parent subsystem to left. Click OK to apply the change and navigate back to the parent model by clicking the up arrow button next to the Subsystem tab.

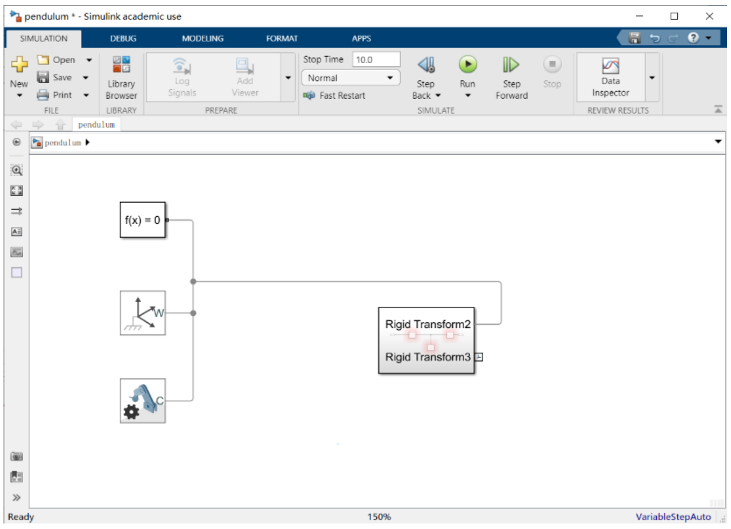
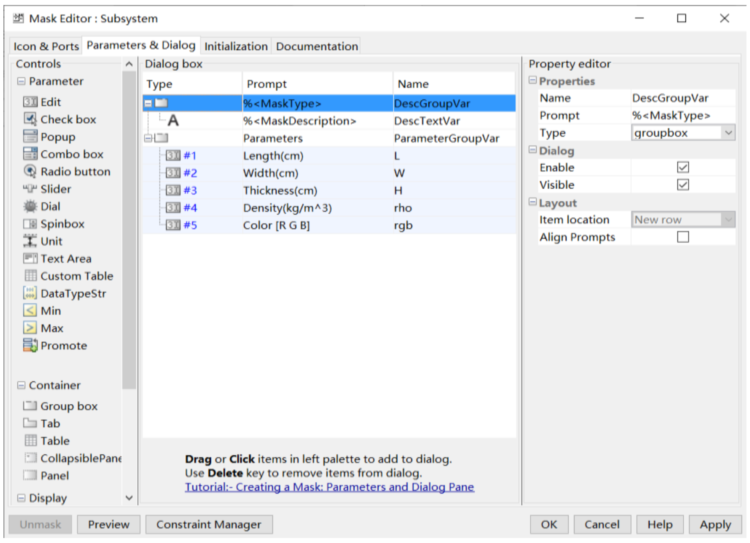


Figure 2.2.4 Subsystem

1.6 Specify the parameters


Mask Editor : Subsystem

Icon & Ports | Parameters & Dialog | Initialization | Documentation

Type	Prompt	Name
	%<MaskType>	DescGroupVar
A	%<MaskDescription>	DescTextVar
	Parameters	ParameterGroupVar
#1	Length(cm)	L
#2	Width(cm)	W
#3	Thickness(cm)	H
#4	Density(kg/m^3)	rho
#5	Color [R G B]	rgb

Property editor

Properties	
Name	DescGroupVar
Prompt	%<MaskType>
Type	groupbox
Dialog	
Enable	✓
Visible	✓
Layout	
Item location	New row
Align Prompts	☐

Drag or **Click** items in left palette to add to dialog.
Use **Delete** key to remove items from dialog.
Tutorial:- Creating a Mask: Parameters and Dialog Pane

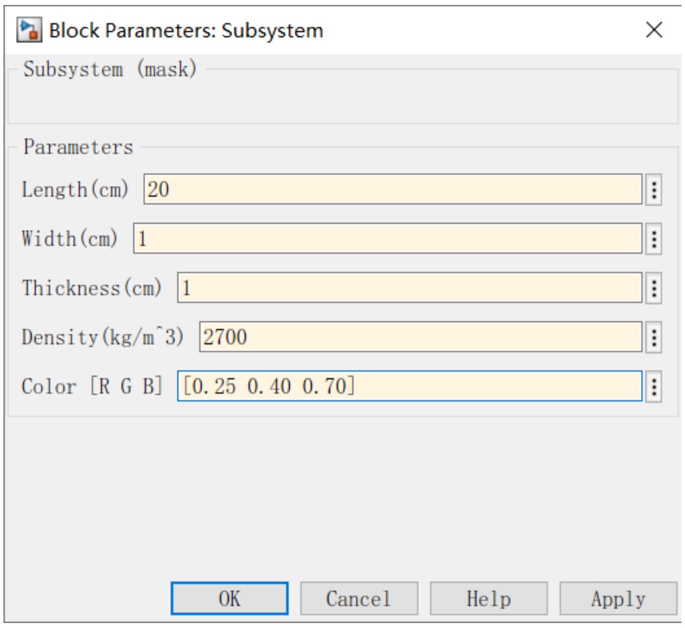
Figure 2.2.5 Specify the parameter

Right-click the Subsystem block and select Mask -- Create Mask. The Mask Editor window opens, where we can define the variables that entered in the Brick Solid and Rigid Transform block dialog boxes.

Click the Parameters & Dialog tab and click Edit. The Prompt property specifies the names of the parameters that you can enter in the Subsystem block parameter window. The Name property specifies their corresponding

MATLAB variables, shown as Figure 2.2.5.

Double-click the Subsystem block. The mask of subsystem opens. Enter the numerical values shown in Figure 2.2.6 the Subsystem Block Parameters. These values specify the shape of Brick Solid and the location of Rigid Transform blocks.


Block Parameters: Subsystem

Subsystem (mask)

Parameters

Length(cm)	20
Width(cm)	1
Thickness(cm)	1
Density(kg/m^3)	2700
Color [R G B]	[0.25 0.40 0.70]

Figure 2.2.6 Insert the numerical values

1.7 Visualize Model

Run the model. The Mechanics Explorer opens with a front view of the simple link model. So far, the link has been fully constructed shown as Figure 2.2.7

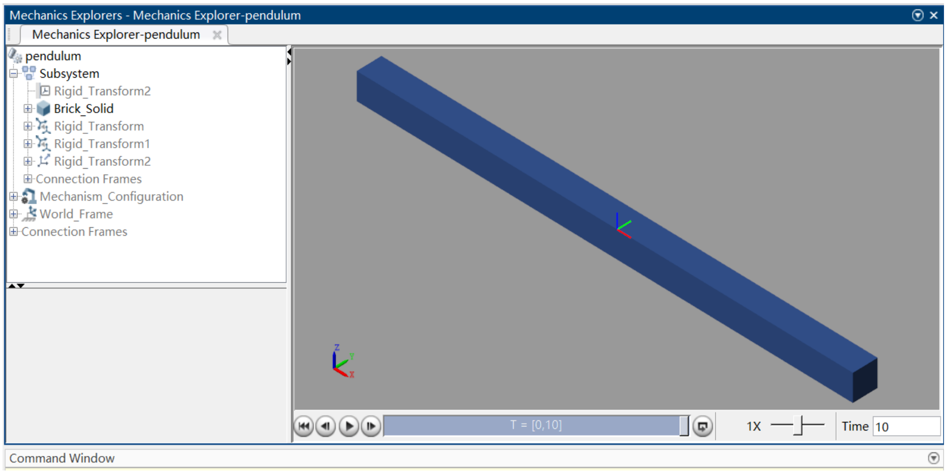


Figure 2.2.7 Visualize model

2. Model pendulum

As a simple link is modeled, we can now add joint and define initial condition.

2.1 Add joint

In Simscape -- Multibody -- Joints library, add a revolute joint. This block provides one rotational degree of freedom between its port frames. Revolute joint block represents a revolute joint acting between two frames. This joint has one rotational degree of freedom represented by one revolute primitive. The joint constrains the origins of the two frames to be coincident and the z-axes of the base and follower frames to be coincident, while the follower x-axis and y-axis can rotate around the z-axis.

For properties, we can specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint.

Ports B and F are frame ports that represent the base and follower frames, respectively. The joint direction is defined by motion of the follower frame relative to the base frame.

Add a Solid Block, the block connects rigidly to the World frame and therefore has no effect on model dynamics, specify the following parameters as Table

Parameter	Value	Units
Geometry -- Dimensions	[4 4 4]	Change to cm
Graphic – Visual Properties -- Color	[0.80 0.45 0]	Not applicable

Table 2.2.3 Parameter of solid box

Connect the blocks as shown in the Figure 2.2.8. The port orientation of the Revolute Joint block becomes important when you specify joint state targets, prescribe joint actuation inputs, or sense joint dynamic variables. The Revolute Joint block interprets each quantity as that applied to the follower frame with respect to the base frame, so switching the port connections can affect model assembly and simulation.

The Revolute Joint block uses the common Z axis of the base and follower frames as the joint rotation axis. To ensure the pendulum oscillates under the effect of gravity, change the gravity vector so it no longer aligns with the Z axis. To do this, in the Mechanism Configuration block dialog box, set the Uniform Gravity -- Gravity parameter to [0 -9.81 0].

Mechanism Configuration block Sets mechanical and simulation parameters that apply to an entire machine, the target machine to which the block is connected. In the Properties section below, you can specify uniform gravity for the entire mechanism also set the linearization delta. The linearization delta specifies the perturbation value that is used to compute numerical partial derivatives for linearization. Port C is frame node that you connect to the target machine by a connection line at any frame node of the machine.

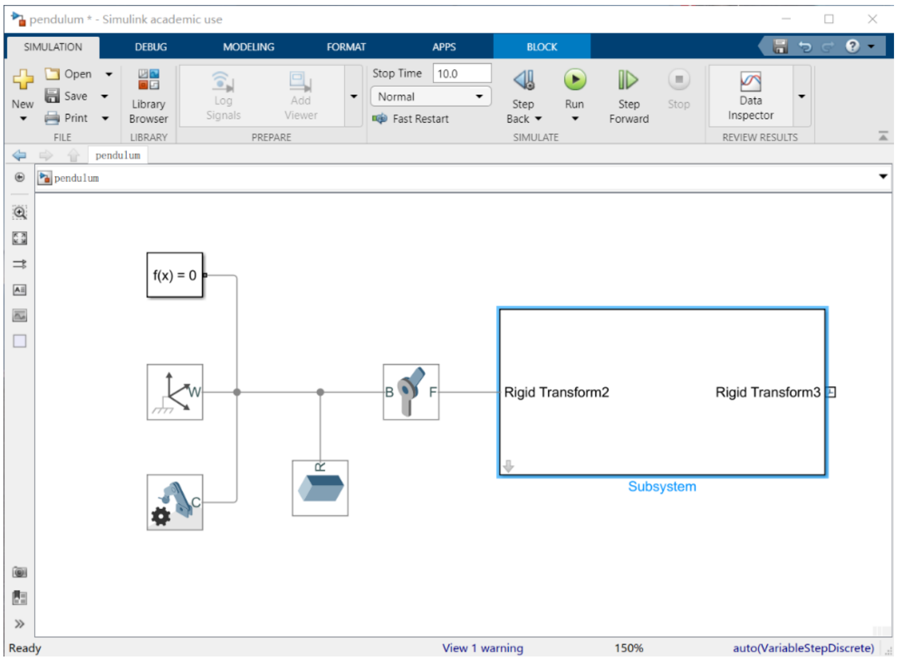


Figure 2.2.8 Assembly Model

2.2 Set initial position and Configure Solver

In the Revolute Joint block dialog box, using the State Targets menu, we can specify the desired joint angle. In this example, we set the angle in default which corresponds to a horizontal pendulum starting position.

To Configure Solver open the Configuration Parameters. In the Modeling tab, click Model Settings. In the Solver tab, set the Solver parameter to ode15s (stiff/NDF). This solver is the recommended choice for physical models. Set Max step size to 0.01, the small step size increases the simulation accuracy and produces a smoother animation in Mechanics Explorer. Small step sizes can have a detrimental effect on simulation speed but, in such a simple model, a value of 0.01 provides a good balance between simulation speed and accuracy.

Update the block diagram. In the Modeling tab, click Update Model. In the Mechanics Explorer toolstrip, check that the View convention parameter is set to Y up (XY Front). This view convention ensures that gravity is vertically aligned on your screen. Select a standard view button to refresh the Mechanics Explorer display.

3 Simulation

The Figure 2.2.9 shows the simulation of the model.

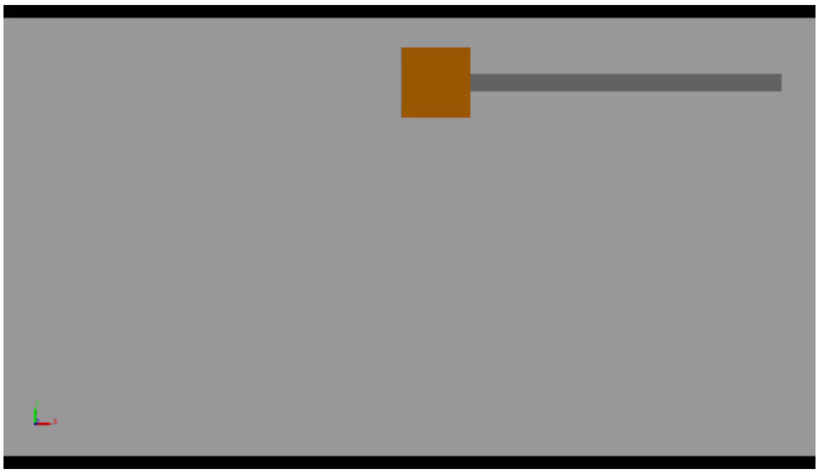


Figure 2.2.9 Simulation

4 Analyze Pendulum

In this section, various forces and torques is added to a model, using blocks with motion sensing capability, you analyze the resulting dynamic response of the model. The result is a set of time-domain and phase plots, one for each combination of forces and torques.

By adding forces and torques to this model, you incrementally change the pendulum from undamped and free to damped and driven. The forces and torques that you apply include:

Gravitational force (F_g) — Global force, acting on every component in direct proportion to its mass, that you specify in terms of the acceleration vector g. You specify this vector using the Mechanism Configuration block.

Joint damping (F_b) — Internal torque, between the pendulum and the joint fixture, that you parameterize in terms of a linear damping coefficient. You specify this parameter using the Revolute Joint block that connects the pendulum to the joint fixture.

Actuation torque (F_A) — Driving torque, between the pendulum and the joint fixture, that you prescribe directly as a Simscape physical signal. You prescribe this signal using the Revolute Joint block that connects the pendulum to the joint fixture.

To sense the position and velocity, in the Sensing menu of the Revolute Joint block dialog box, select the 2 variables. The block exposes two additional physical signal ports, labeled q and w, that output the angular position and velocity of the pendulum with respect to the world frame.

Add the following blocks to the model, shown as Table2.2.4. You use them here to output the joint position and velocity to the MATLAB base workspace.

Library	Block	Quantity
Simscape > Utilities	PS-SimulinkConverter	2
Simulink > Sinks	To Workspace	2

Table 2.2.4 Output variable to workspace

Connect the blocks as shown in the Figure 2.2.10. Ensure that the To Workspace block with variable name out.q connects, through the PS-Simulink Converter block, to the Revolute Joint block port q, and that the To Workspace block with variable name out.w connects to the Revolute Joint block port w.

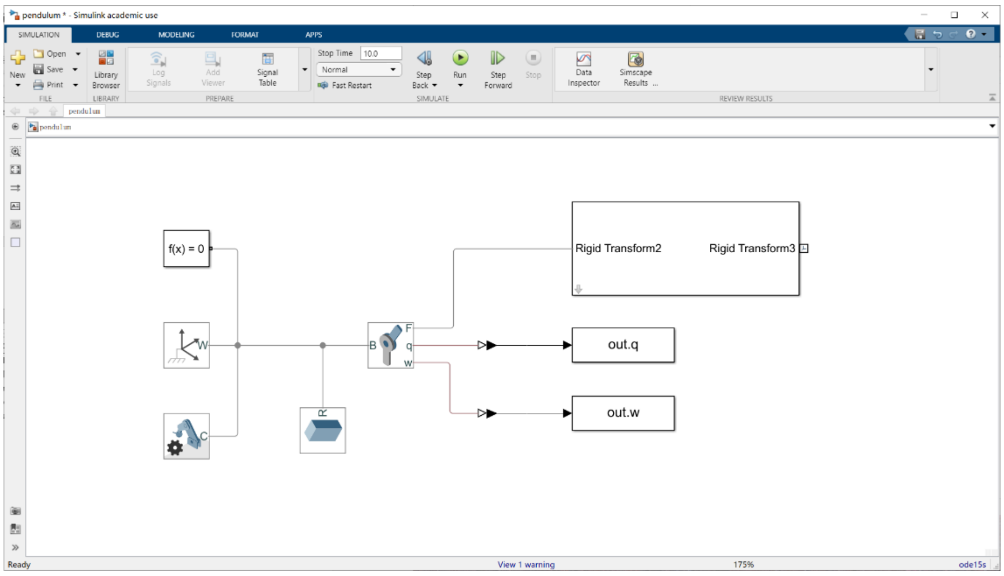


Figure 2.2.10

Run the simulation, plot pendulum angular velocity and pendulum angle by using code at MATLAB command prompt, the result shows as Figure2.2.11

```
figure; % Open a new figure
hold on;
plot(q); % Plot the pendulum angle
plot(w); % Plot the pendulum angular velocity
```

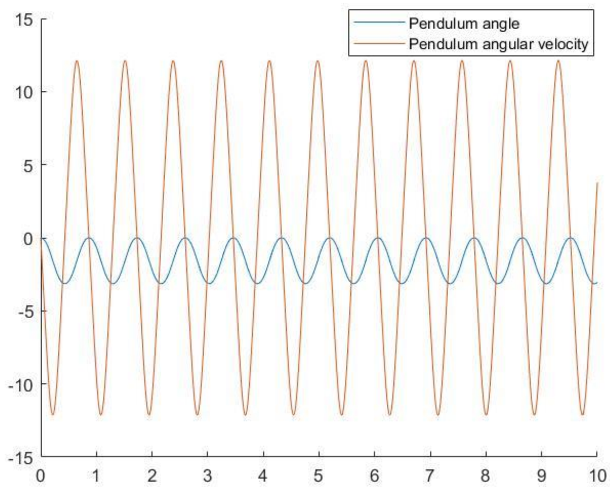


Figure 2.2.11 Pendulum angle and velocity as a function of time

5 Analyze Damped Pendulum

The damping coefficient causes energy dissipation during motion, resulting in a gradual decay of the pendulum oscillation amplitude. To have a better understanding, we set Internal Mechanics -- Damping to $8e^{-5}(N*m)/(deg/s)$ in the Revolute Joint block dialog box.

Then the joint position and velocity with respect to time can be plotted as Figure 2.2.12

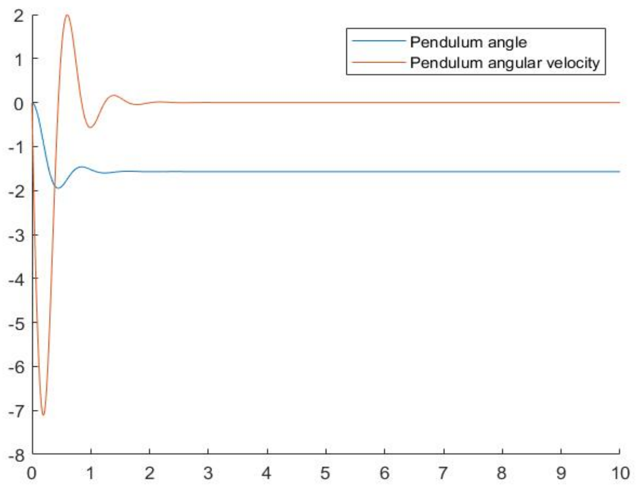
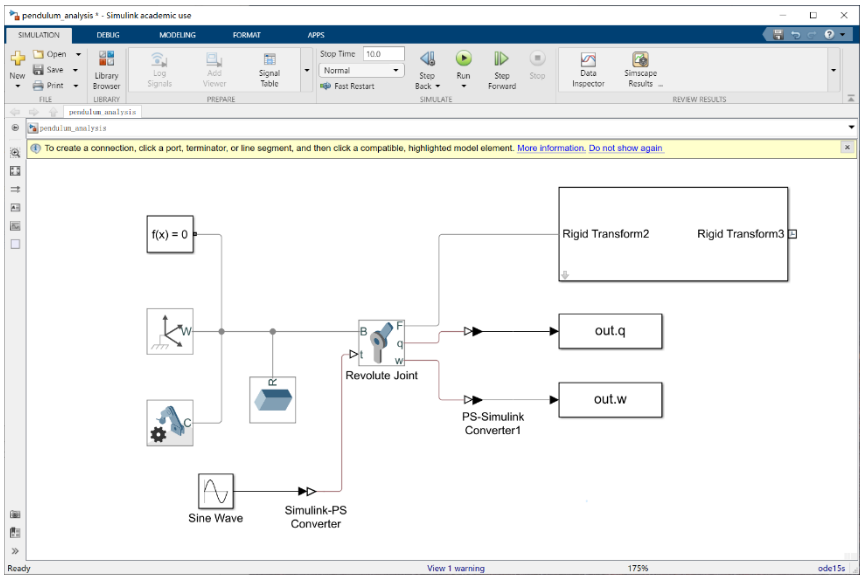


Figure 2.2.12 Damped pendulum

6 Analyze Damped and Driven Pendulum

In the Revolute Joint block dialog box, set Actuation -- Torque to Provided by Input. The block exposes a physical signal input port that you can use to prescribe the joint actuation torque. In order to input a Sine wave physical signal, we need a Simulink-PS Converter block to convert Simulink signal to a Simscape physical signal.



To create a connection, click a port, terminator, or line segment, and then click a compatible, highlighted model element. More information. Do not show again

Figure 2.2.13 Damped and driven pendulum

Figure2.2.14 is the result of joint position and velocity with respect to time

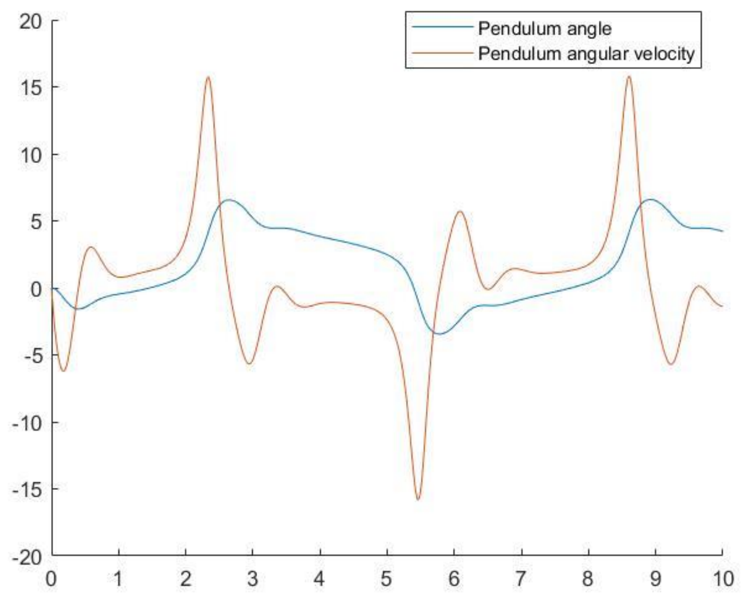


Figure 2.2.14 Pendulum angle and velocity of a damped and driven pendulum

2.3 Simplified suspension

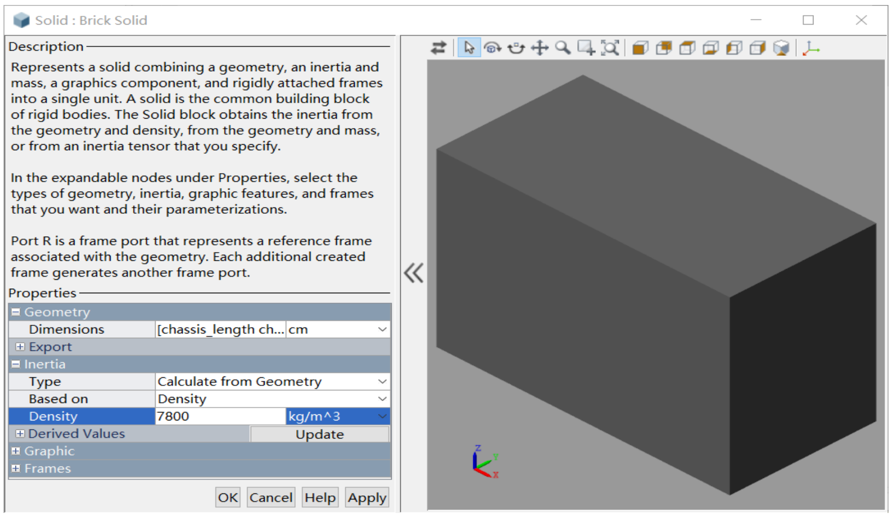
In this chapter, a simplified suspension model is shown and simulated. The system will undergo different types of force tests, we can observe the left and right wheel, the suspension arms, shock absorber. Also, we have some visualization for the instantaneous and the roll centers. Some part of the suspension is simplified, the advantage is that we can easily parameterize the model and optimize design iterations automatically using optimization tools

1. Build the components

   The components of a suspension system that consists of a simplified chassis, suspension arms and attire with trim to form the wheel system.

   1.1 Simplified chassis

   The simplified chassis is modeled by a solid block. Notice that, the reference frame is by default at the center of the geometry. For convenience, the dimension is not specified with numerical numbers, but with variable names. All the variables should be well defined in advance using a Matlab script file. Shown as Figure 2.3.1.


Solid : Brick Solid

Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

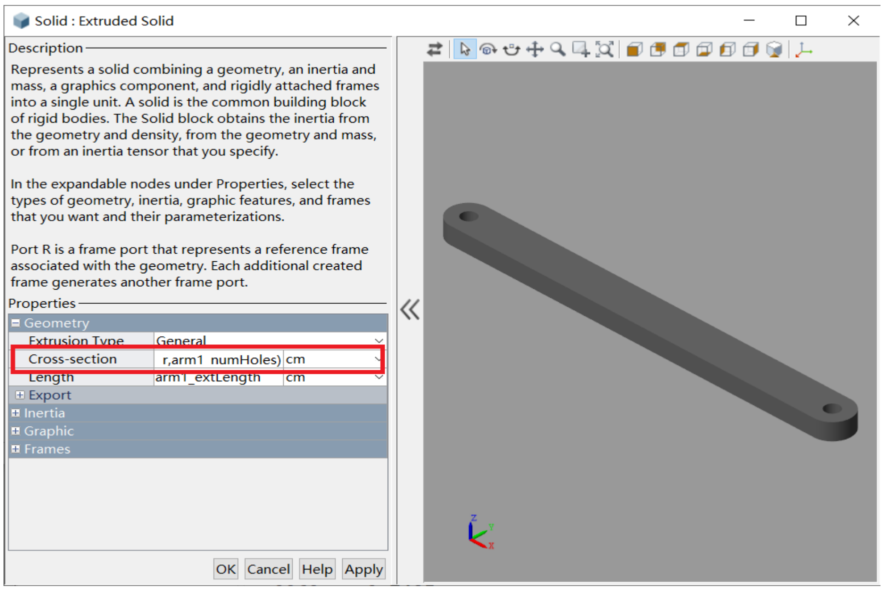
Properties

⊟ Geometry		
Dimensions	[chassis_length ch...	cm
⊞ Export		
⊟ Inertia		
Type	Calculate from Geometry	
Based on	Density	
Density	7800	kg/m^3
⊞ Derived Values		Update
⊞ Graphic		
⊞ Frames		

Figure 2.3.1 Simplified Chassis

1.2 Suspension arms

The suspension arm is a more complex part. To specify the complex geometry, set the shape parameter of solid body to a general extrusion. By this we can specify the cross section for the extrusion as an array of coordinate pairs. The cross section can be considered as a sweep along an axis (Z-axis at default) of the reference frame. We can extrude shapes that represent the suspension arm. Shown as the Figure 2.3.2.


Solid : Extruded Solid

Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

⊟ Geometry		
Extrusion Type	General	
Cross-section	r,arm1 numHoles)	cm
Length	arm1_extLength	cm
⊞ Export		
⊞ Inertia		
⊞ Graphic		
⊞ Frames		

Figure 2.3.2 Suspension arm

To specify the cross section, modify the cross-section. This will extend the custom cross-section through the z axis and centroid located at the block reference frame. Each row of array represents a different coordinates pair, adjacent coordinates pair are connected with a straight line to form a closed polygon. The dimension and the shape of our suspension arm is specified by a function. We could write our own function similarly for a shape of our choice.

To visualize the cross-section, shown as the Figure 2.3.3, in the figure, we can see the structure of arm.

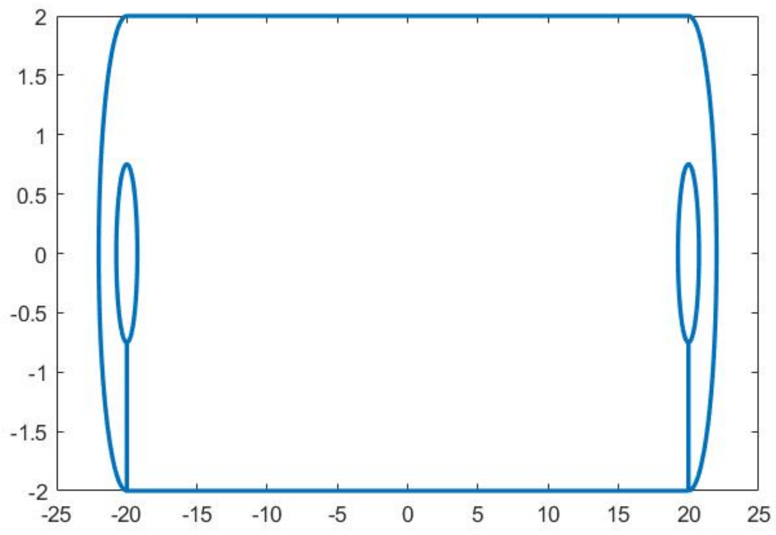


Figure 2.3.3 Visualization of the cross section

Note that for the cross-section parameter to be valid, the array of coordinates must follow the rule:

1. Coordinates must define a closed polygon that does not self-intersect.
2. Counterclockwise loop encloses the solid section, clockwise loop encloses hollow section.

1.3 Rim

To model a rim, insert a Revolved Solid block, specify an array of coordinate pairs as the cross-section parameter in the dialog box similar to extrusion example. The visualization of the cross section is shown as Figure 2.3.4.

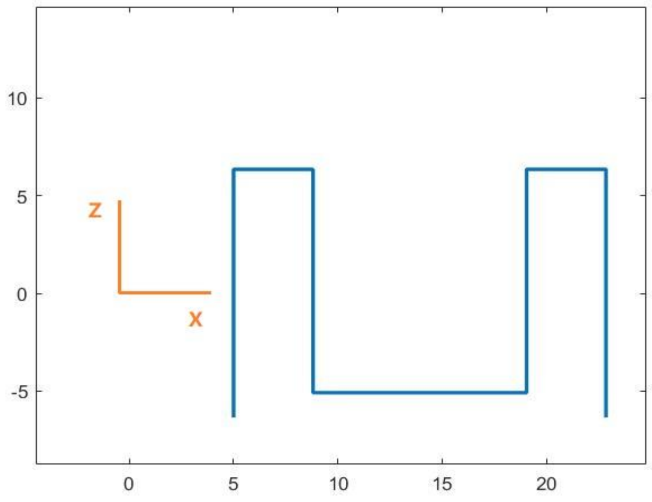
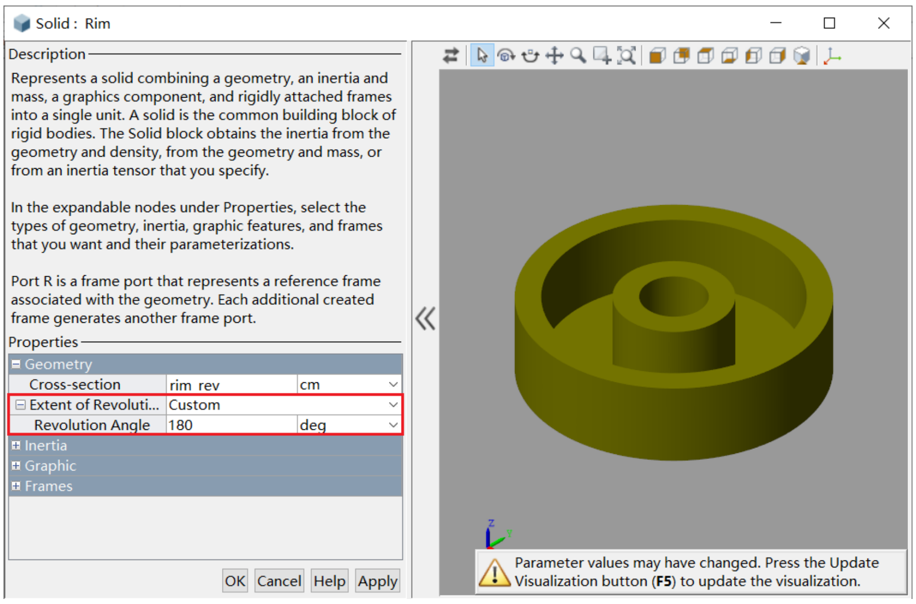


Figure 2.3.4 Cross section of revolved solid body

Here, in Figure 2.3.4 the cross section is shown in the XZ plane, the cross section is rotated around the Z axis to create the revolved solid body. Notice that, in our example, we select the extent of revolution is full, but we can create partially revolved solids by setting this parameter to custom and setting a revolution angle. Shown as Figure 2.3.5.



Solid : Rim

Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

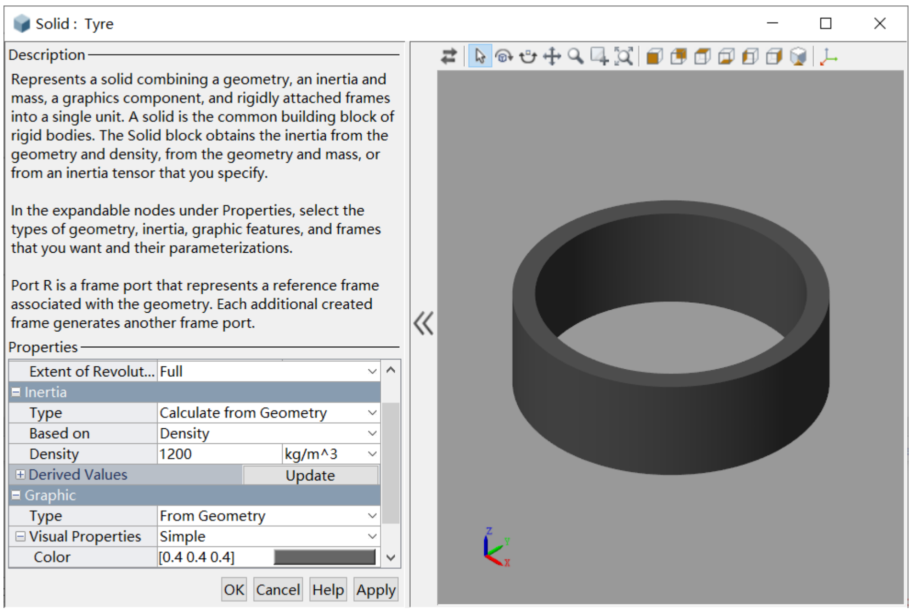
Geometry		
Cross-section	rim_rev	cm
Extent of Revoluti...	Custom	
Revolution Angle	180	deg
Inertia		
Graphic		
Frames		

Parameter values may have changed. Press the Update Visualization button (**F5**) to update the visualization.

Figure 2.3.5 Rim

1.4 Tire

For modeling the tire, the approach is the same with the rim, just modify the cross-section parameter.



Solid : Tyre

Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

Extent of Revolut...	Full	
Inertia		
Type	Calculate from Geometry	
Based on	Density	
Density	1200	kg/m^3
Derived Values	Update	
Graphic		
Type	From Geometry	
Visual Properties	Simple	
Color	[0.4 0.4 0.4]	

Figure 2.3.5 Tire

2. Proper connection

In Mechanics Explorer shown as Figure 2.3.6, we can notice that all the components are aligned along the center of mass, which means we have to position these components appropriately. We use rigid transform blocks to do this job.

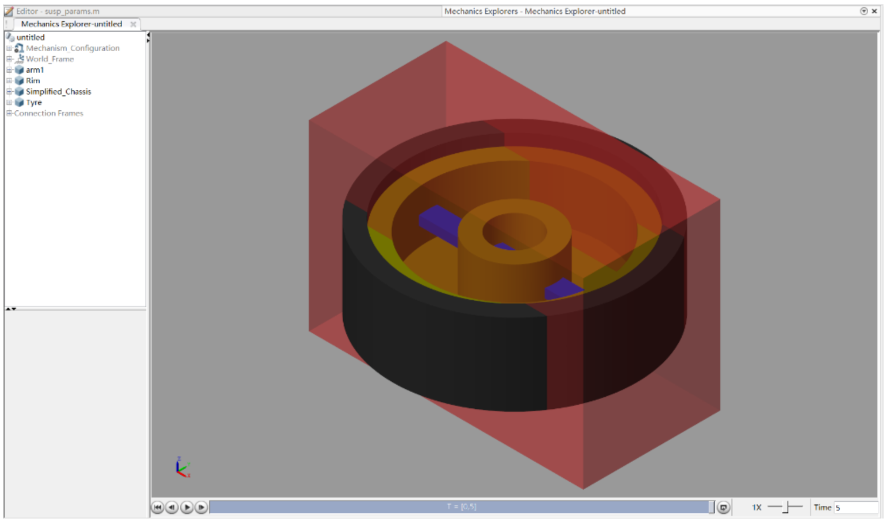


Figure 2.3.6 Need proper connection

2.1 Position the suspension arm

To position the arm to the right place we need to implement coordinate transforms represent degrees of freedom and specify body interfaces.

As mentioned before, the transform block defines fixed relationship between Base B and Follower F coordinate frames. This relationship is defined by translational and rotation transformation.

First, specify the translation with respect to chassis. We need to move the arm along the Y axis by half of the chassis width, we also move it to the appropriate position in the vertical Z axis, the distance is the variable arm1_chassis_dist which is the specified in the Matlab script. We use the Cartesian method to specify the translation in two axes. Then connect the rigid transform between the chassis and the arm. This transform block is

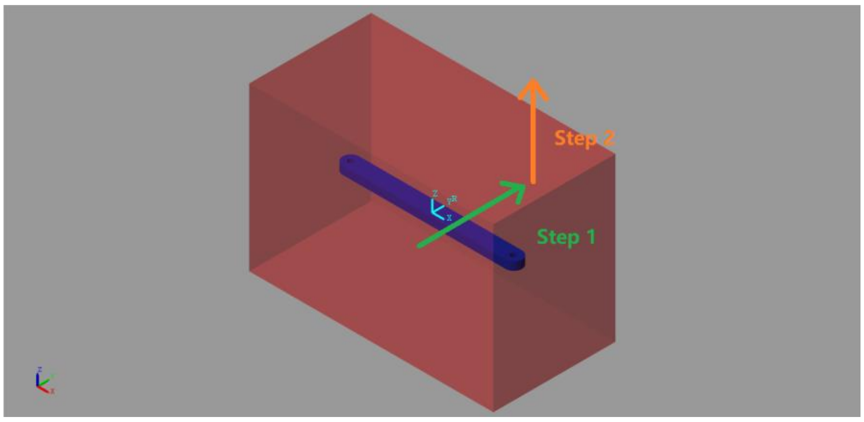


Figure 5 2.3.7 Translation of suspension arm with respect to chassis named as chassis to suspension arm.

Then we need to rotate the arm about its z axis by 90 degrees and the x axis by 90 degrees as well as with respect to its own axis. After rotation, Base Z axis becomes the Follower X axis, and the Base X axis becomes the Follower Y axis. In transform block, select method as Aligned Axes, then modify the coordinate pairs. After that we still need to do another translation by half the length of the arm to position the link on the chassis. This transform block is named suspension arm to chassis.

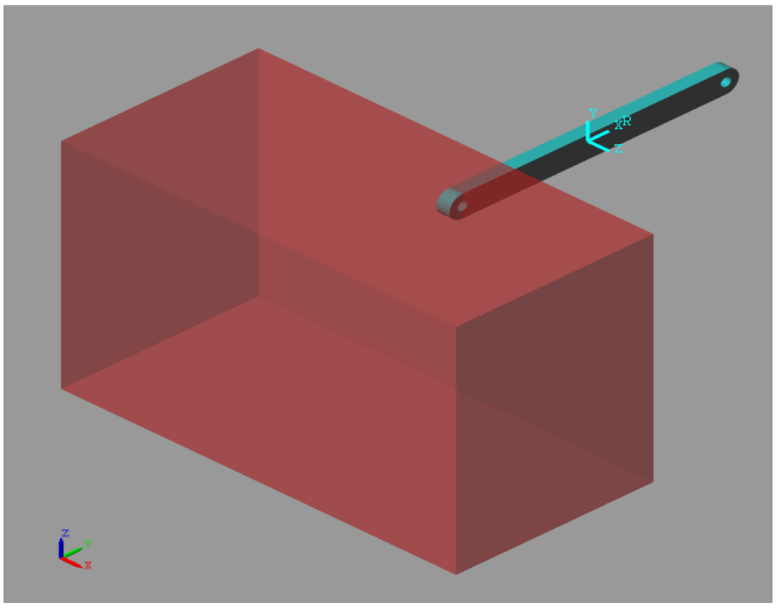


Figure 2.3.8 Position the suspension arm

2.2 Connection between arm and rim

First, we need to translate the rim to the end of the arm. Next, we need to rotate the rim to position it perpendicular to the arm this rotation is going to be with respect to the rim about its Y axis 90 degrees. This is going to be the translation with respect to the arm. As the rotation is around its own local Y axis, the transformation is going to be from the rim to the arm. So, we have to flip this block. The rim connection is at the center of the rim, instead it needs to be translated about the vertical axis by the parameter wheel_bias_1.

2.3 The connection between tire and rim

As the tire is rigidly connected with rim. Any transformation that the rim undergoes the tire also undergoes.

So now we have positioned the arm with respect to the chassis and the rim with respect to the arm. Shown as Figure 2.3.9.

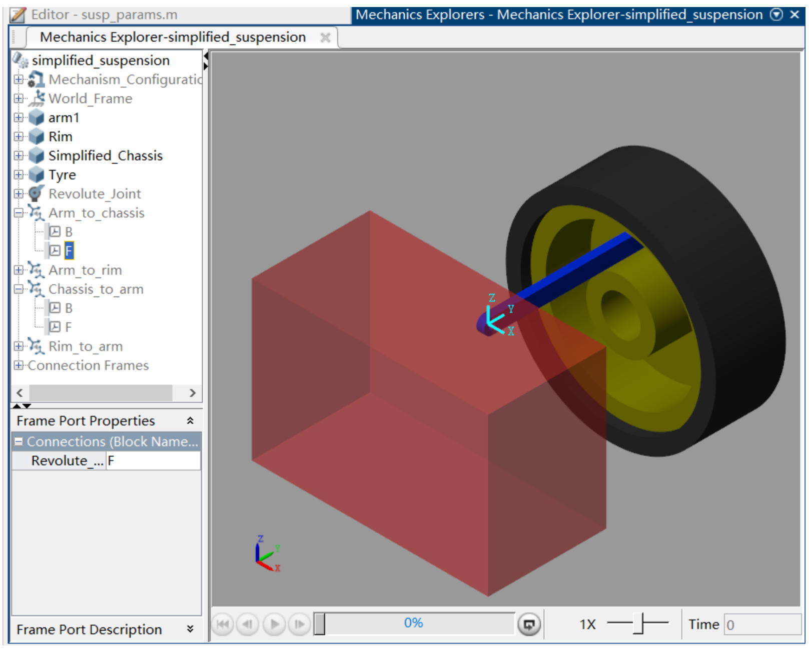


Figure 2.3.9 Proper connection between chassis, suspension arm and rim

2.4 Joints

For now, our model does not have any dynamics behavior even all the components are positioned correctly. That is because all the components are rigidly connected, we need to define the degrees of freedom for the components with respect to each other by adding joints. In our model, the links between the chassis and the arm, arm and the rim, represented by the revolute joints.

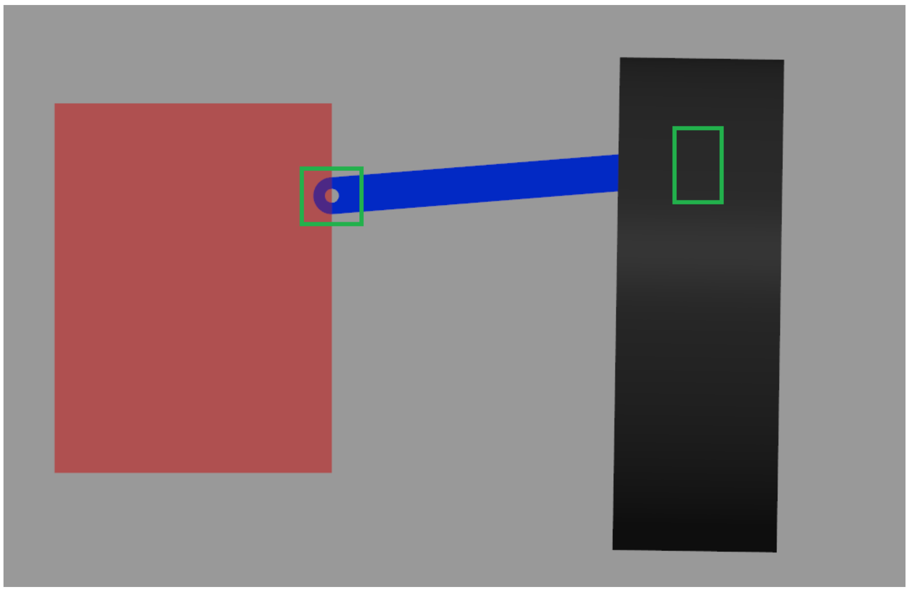


Figure 2.3.10 Link between components

As mentioned in the pendulum model in previous, a revolute joint has a single degree of freedom which lets the bodies rotate about the Z-axis.

So, it is important that the base coordinate and the follower coordinate Z-axis are aligned. In this example we would like to connect the joint between two Rigid Transforms blocks. In this case the Base of the joint will be the follower of the Chassis to Arm transform block. We can notice that the Z axis is pointing upwards, however for the revolute joint to make sense it needs to be pointing out of the screen (along the X-axis). Also, the Follower in Arm to Chassis block, the Z axis is pointing upwards. The situation is shown as Figure2.3.11.

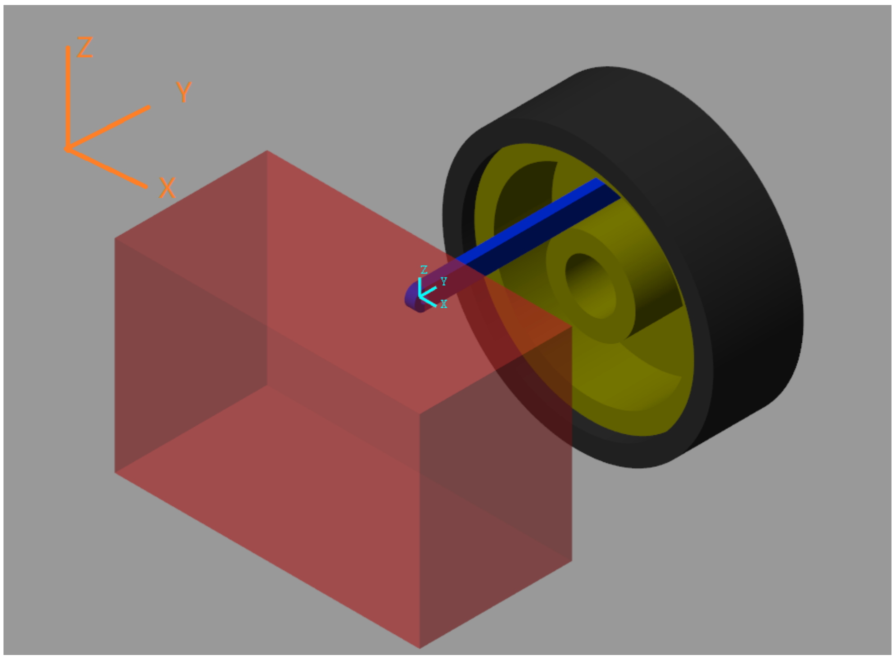


Figure 2.3.11 Wrong orientation of Z-Axis

We have to rotate the 2 Z-axis. In this case even if our base and the follower frames are co-located in other words the origins are at the same position they are not oriented correctly. We need an addition rotation transform. For both Rigid Transform block.

First, in Chassis to Arm block, a rotation about Y-axis with 90 degrees is added. Then inside the Arm to Chassis block, the Base coordinate is shown as Figure 2.3.12. Mention that the Follower in Arm to Chassis block has to be aligned with the Follower inside Chassis to Arm block, so another rotation is added shown as Figure 2.3.13.

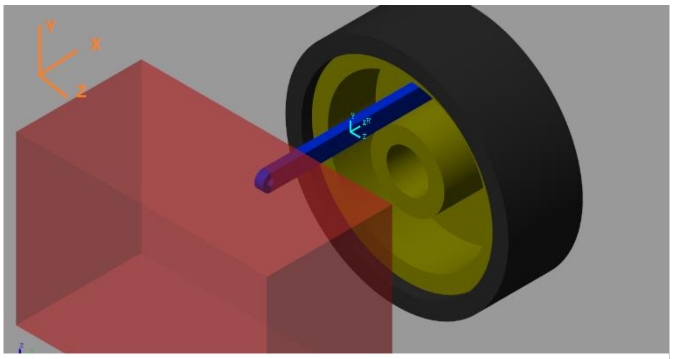
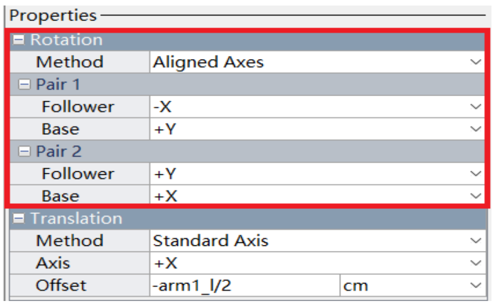


Figure 2.3.12 Base coordinate



Properties

Rotation	
Method	Aligned Axes
Pair 1	
Follower	-X
Base	+Y
Pair 2	
Follower	+Y
Base	+X
Translation	
Method	Standard Axis
Axis	+X
Offset	-arm1_l/2 cm

Figure 2.3.13

Now 2 frames are aligned for the revolution, Z-axis is also in the correct orientation for the joints.

While for the revolute joint add between rim and wheel, after observing Rim to Arm transform block and Arm to Rim transform block , the Followers are located at the same position which means 2 frames are aligned. Also, the Z-axis is in the right orientation, no additional operation is needed.

2.5 Create the second arm

To create and assembly the second arm, is just the same with the first suspension arm, the only different is the specification and variable names which is already defined in the Matlab script. After grouping related components to make subsystems, shown as Figure 2.3.14 can be obtained.

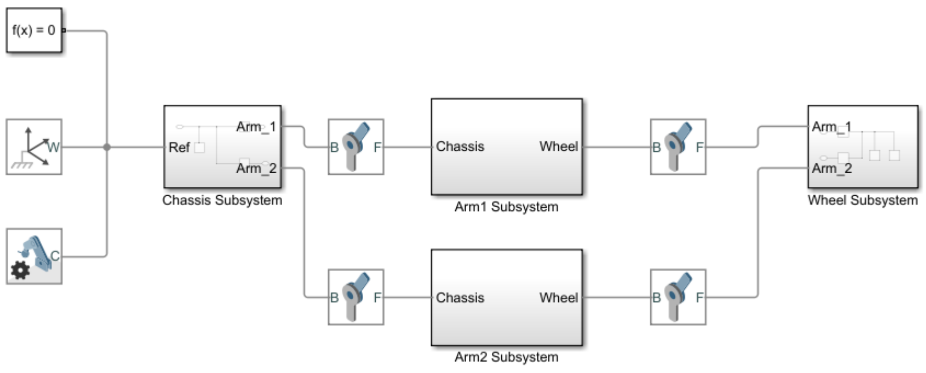
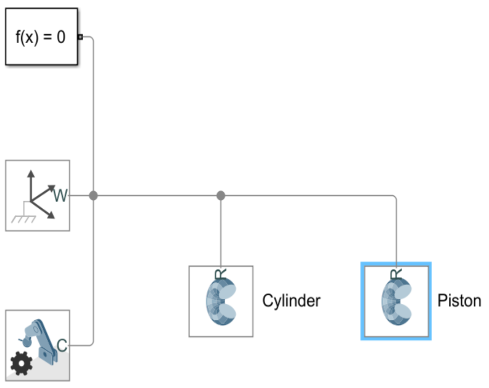


Figure 2.3.14 Mechanical Assemblies

2.6 Create Shock Absorber

The system still behaves like a pendulum now, with only forces under gravity, because the system is not fully constrained since the two arms are not bounded by a shock absorber component. The shock absorber can be simplified as the combination of a cylinder and a piston. We use Revolved Solid block to create piston and cylinder.



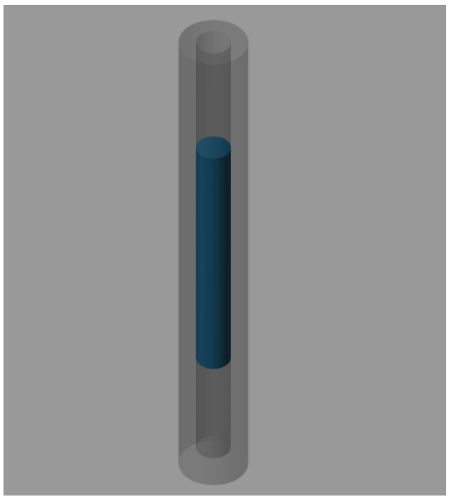


Figure 2.3.15 Shock Absorber

Create the other port and connect the cylinder to the system. Once, the closed loop is created, which means the 2 arms are forced connect to the same point, center of mass of the cylinder and the piston, which is not physically possible in mechanical configuration. In order to avoid this error, we better not to close the loop at first. Instead, it is better to build the assemblies increment from one end to the other. Shown as Figure 2.3.16

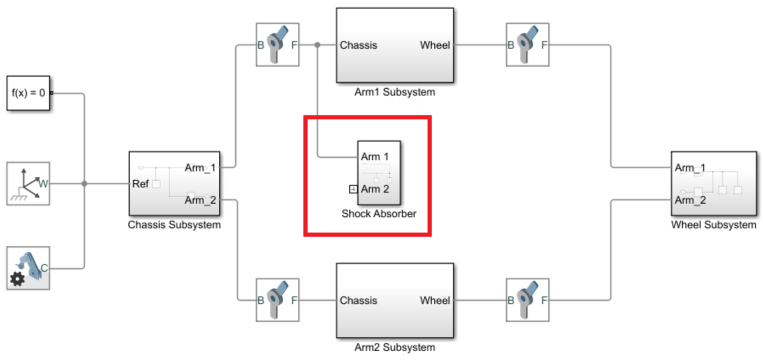


Figure 2.3.16 Open loop

To move the shock absorber to the right position and orientation.

First, we have to rotate the cylinder about its Y-axis by negative 90 degrees. Notice that the rotation is going to be form cylinder to the arm, which means the Base of the Rigid Transform block is connect to the cylinder.

Then to move the link between the chassis and shock shocker to the end of the cylinder instead of at the center of mass, we have to translate this component along its own Z-axis.

After that, add a revolute joint between the cylinder and the chassis connection point and check the rotation axis Z-axis is in the right orientation or not.

For our example, the piston needs to be able to move up and down, but the cylinder and the piston are rigidly fixed to each other, we need to add a single translation degree of freedom. To do this, we can use a Prismatic Joint added between cylinder and piston. Prismatic joints represent a prismatic joint between two frames. This joint has one translational degree of freedom represented by one prismatic primitive. The joint constrains the follower origin to translate along the base z-axis, while the base and follower axes remain aligned. In the expandable nodes under Properties, specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint. After you apply these settings, the block displays the corresponding physical signal ports.

Next connect the piston to the lower arm at a given distance. Again, add a revolute joint. In this case we can see that the Z-axis is pointing down for the Base frame. While is the Follower of the Chassis to Arm the Z-axis is pointing out, which means we need to align these 2 axes by rotating the frame about Y-axis by 90 degrees. After that using the translation dialog block to move the piston to proper position shown as Figure 2.3.17

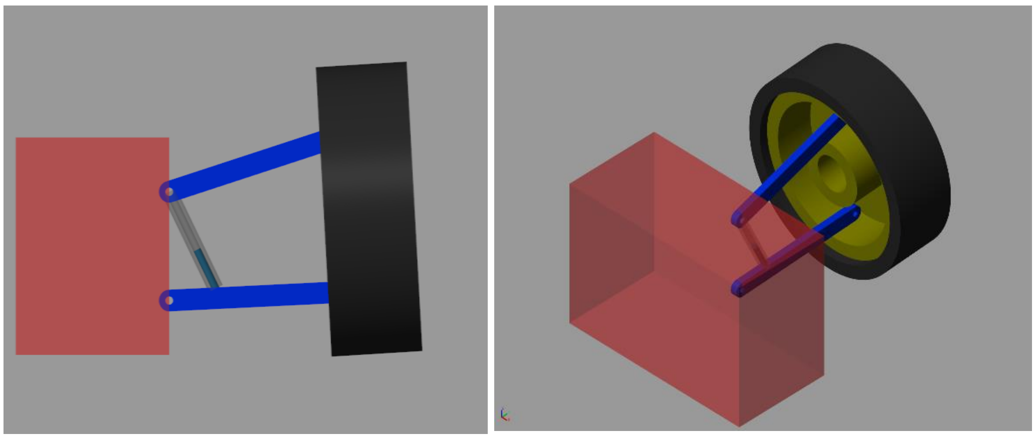


Figure 2.3.17

From Figure 2.3.17 we can notice that the wheel position seems misguided suppose we would like the wheel to be positioned lower for an initial state initial configuration. For mechanical assemblies can be guided by providing initial conditions to necessary joints in this case it could be useful to specify an initial condition for the shock absorber piston’s displacement. By doing that we can control the wheel’s vertical position. To do this, adjust the State Targets in Prismatic Joint, specify the position target. This operation is the same with the Mass-Spring-Damper model, in that model we also set the state initial conditions.

Finally, connect the piston to the lower arm to close the loop.

For now the system is still act like a pendulum, the reason is that the shock absorber is not limiting the motion of the suspension arms. In the dialog box of Prismatic Joint, inside the Sensing section, we can config the joint to output various quantity, such as position shown as Figure 2.3.18.

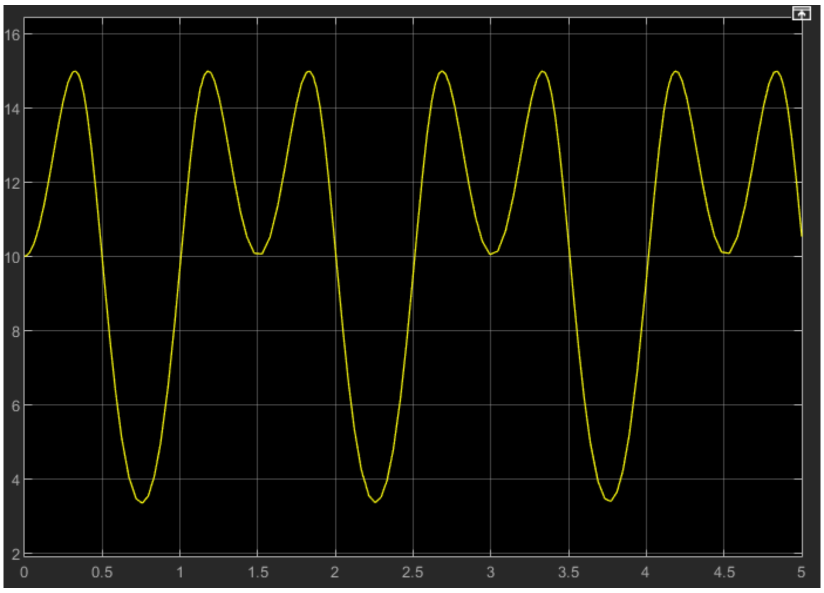


Figure 2.3.18 Undamped displacement

The system seems to be undamped because the joint does not have any stiffness or damping effect. To add real world mechanic effect, we can specify the equilibrium position, spring stiffness and damping to model through Internal Mechanics dialog.

2.7 Add external force

We will use a test platform and exert force on our tire.

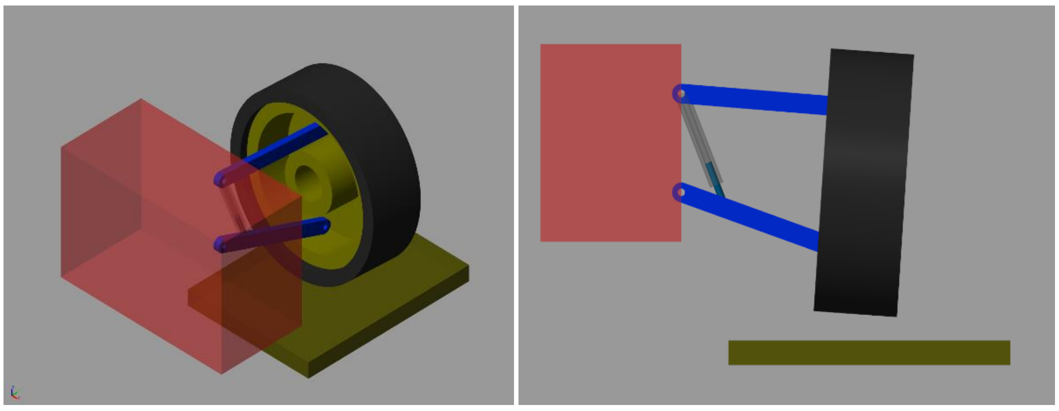
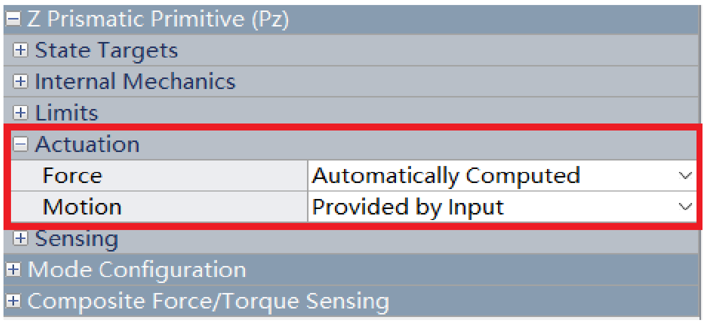


Figure 2.3.19 Test platform

We can move the test platform by using joint blocks. In our example, suppose platform is restricted to only a translational motion in one axis that is the vertical z axis of the world frame we can implement this using a prismatic joint. The joint is added between the world and the platform.

Under the Joint block, inside the actuation section, we specify the

Automatically Computed for motion and set force to be Provided by Input. This actuation mode is called forward dynamics. Alternatively, we can select the motion profile to be Provided by Input and the force to be Automatically Computed, this is called inverse dynamics. In our scenario we are not interested in the forces required to move the plate rather we would just like to move the platform with a provided motion profile. So, we set the motion to be Provided by Input and automatically compute the force required to get that motion profile. Shown as Figure 2.3.20



Z Prismatic Primitive (Pz)
- State Targets
- Internal Mechanics
- Limits
- Actuation

Force	Automatically Computed
Motion	Provided by Input

- Sensing
- Mode Configuration
- Composite Force/Torque Sensing

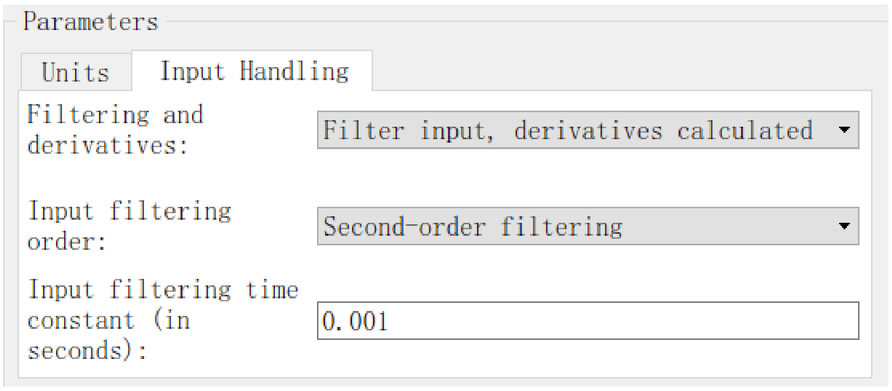
Figure 2.3.20 Inverse dynamics

After that we can input a physical signal to the prismatic joint. We can input any user-defined signal through the Input Port. This port is a root level input port and can accept data from the MATLAB scripts. Additional information for Simulink-PS Converter is there is a Input Handling tab, to avoid solver errors it is recommended that we provide the first and second derivatives that is velocity and acceleration of a motion profile. Input filtering makes the input signal smoother and generally improves model performance.

The first way is to use the Filter Input method to take filter derivatives of the position signal.

The second way is to provide our own input derivatives these could be by generated using custom transfer function blocks.

In our example, we use the Filter Input method and use second-order filtering with the default Input Filtering time. Shown as Figure 2.3.21



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

Figure 2.3.21

Then we have to model the force interaction between the platform and the tire. We can use the prismatic joint to establish the degree of freedom between the tire and the platform and restrict the motion to only the vertical axis. We can also use a planar joint to give it three degrees of freedom which is displacement across the y and z of the world frame and the rotation about the x axis. To exert forces on the wheel from the platform we can activate this planar joint using force in the forward dynamic mode. By obverse, we can notice that the force from the platform is always applied vertically regardless of the rotation of the tire which means the force relationship between the platform and the tire is not easy to model.

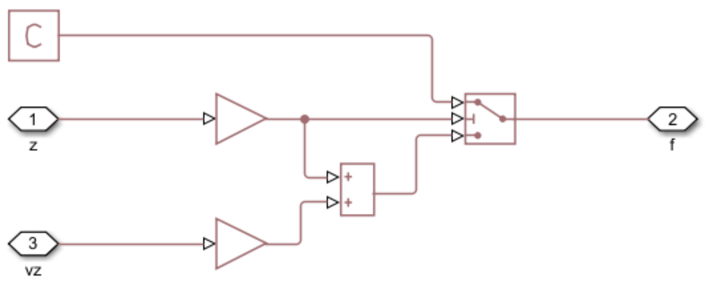
To exert force on any frame regardless of joint blocks we use the External Force and Torque block. This block applies an external force and torque at the attached frame. The force and torque are specified by the physical signal inputs. This block can be configured to apply force to any coordinate frame whether a joint is attached or not. In our case, we need a force that along the Z-axis of the world frame. Then we can see an input force Fz which needs to be computed based on the motion of the platform and the port F is going to be the attached frame which the force will be applied. Show as Figure 2.3.22. To compute the force FZ based on the motion of the platform we need to measure the displacement and velocity between the platform and the wheel. We can do this using a Transform Sensor Block.

fz
F
External Force
and Torque

Figure 2.3.22 External Force and Torque block

A Transform Sensor Block measures time-dependent relationship between two frames. A Transform Sensor passively senses this 3-D time-varying transformation, and its derivatives, between the two frames. In the expandable nodes under Properties, select which rotational and translational relationships, including velocities and accelerations, you want to measure. After you apply these settings, the block displays the corresponding output physical signal ports. Ports B and F are frame ports that represent the base and follower frames, respectively. The sensor measures the transformation and its derivatives as follower frame relative to base frame. The transformation components can be projected into one of several frames.

In our case we want to measure the translation and the velocity along the world z axis. After that we have to compute the force by designing a subsystem Shown as 2.3.23. Given the Z and Vz it computes the force. In the subsystem if the measured distance is greater than zero, we implement the equation to calculate the force otherwise the force is zero. Kw and Dw



```
if (z>0)
  f=Kw*Z+Dw*Vz
else
  f=0
```

Figure 2.3.23 Calculate the force

here are the proportional gain (spring) and derivative gain (damping).

Then we connect the Base and the Follower of the Transform Sensor to the Base and the Follower of the planar joint block which describes the relationship between the platform and the tire.

Last but not least, we connect our subsystem to the output of the Transform Sensor block and connect the External Force block to the output of the calculation block. shown as Figure 2.3.24

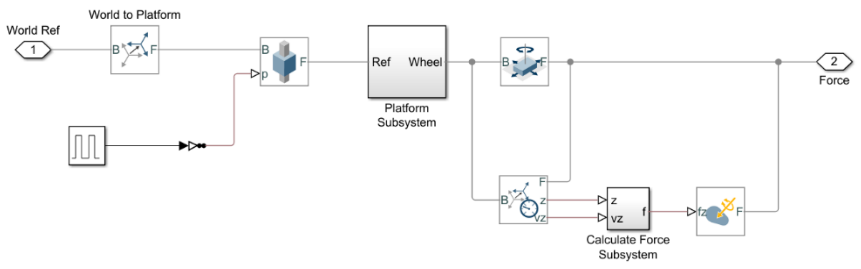


Figure 2.3.24 Test plate

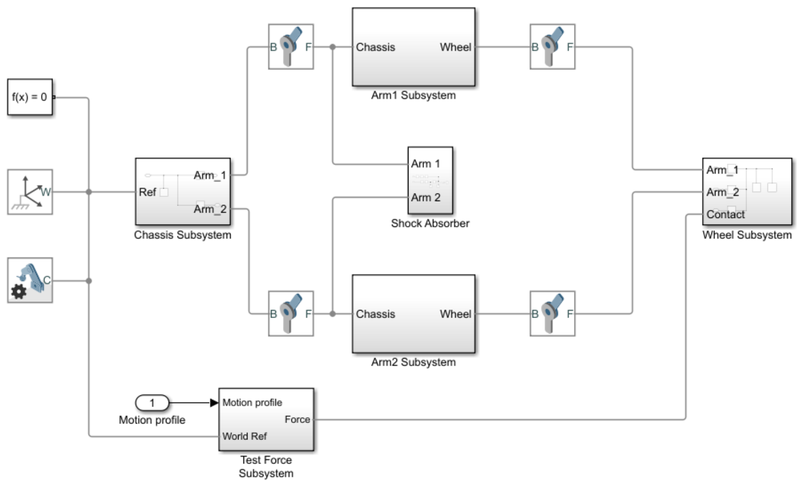


Figure 2.3.25

Back to the upper level of the model, after defining the contact point of the wheel subsystem using the Rigid Transform block. Show as Figure 2.3.25.

2.8 Simulation

Now we can input the signal through the Input Port. The input signal is defined through the Matlab script.

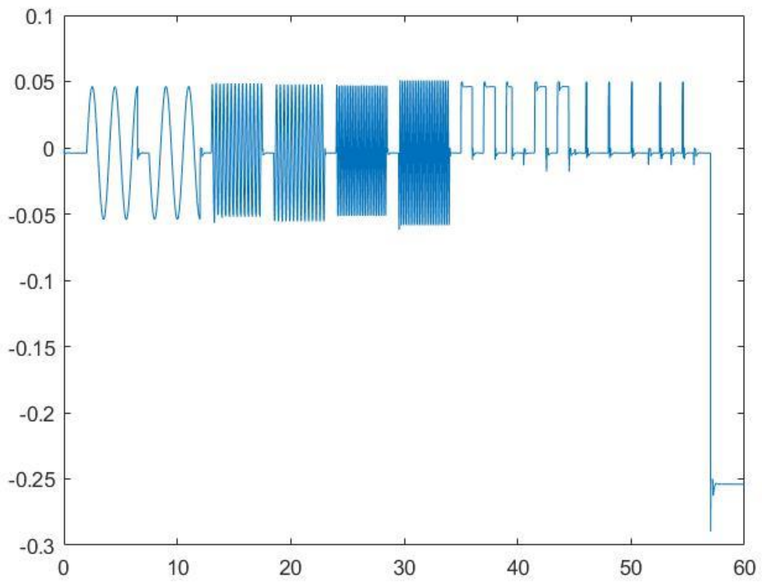


Figure 2.3.26 Input signal

The suspension system going through different motion profiles. We can observe the displacement of the shock absorber for the various force tests. We can measure all sorts of parameters like constraint forces to further analyze the dynamics of our system.

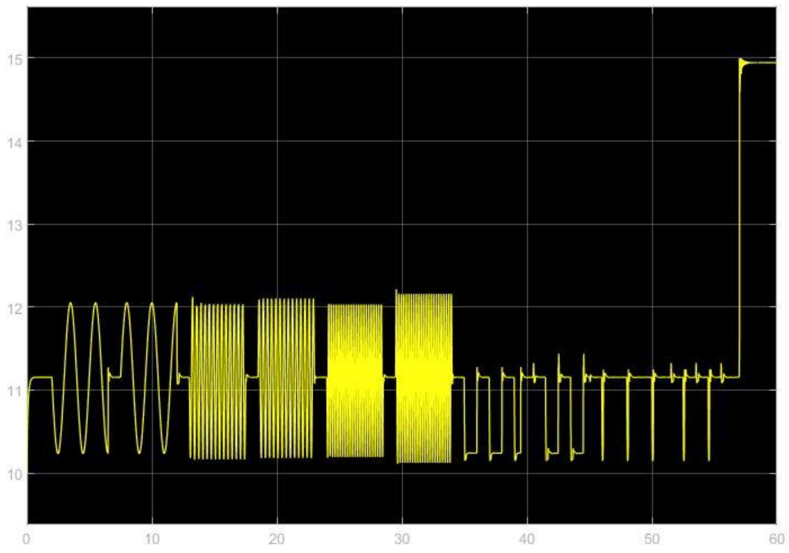


Figure 2.3.27 Displacement of shock absorber

2.9 Designing tune and optimization

Now we have finished building the right part of the suspension, the right part can be built with the same approach. In order to tune the model, it is better to define the Instantaneous center and Roll center. These 2 points define the property and behavior of the suspension. The definition of these 2 centers is shown as Figure 2.3.28

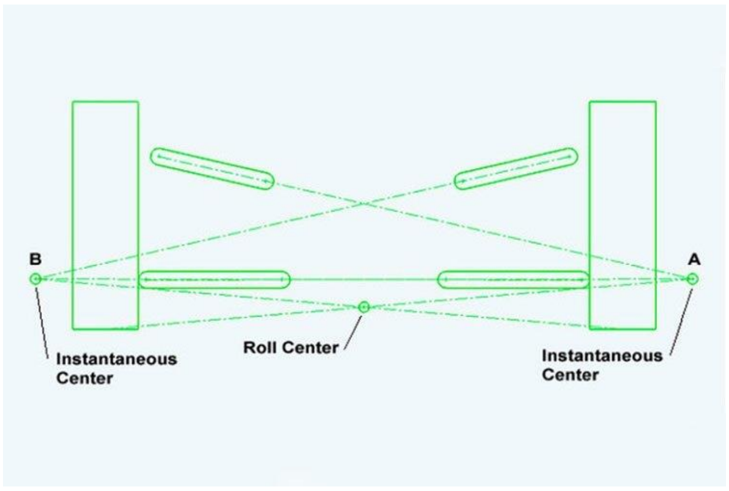
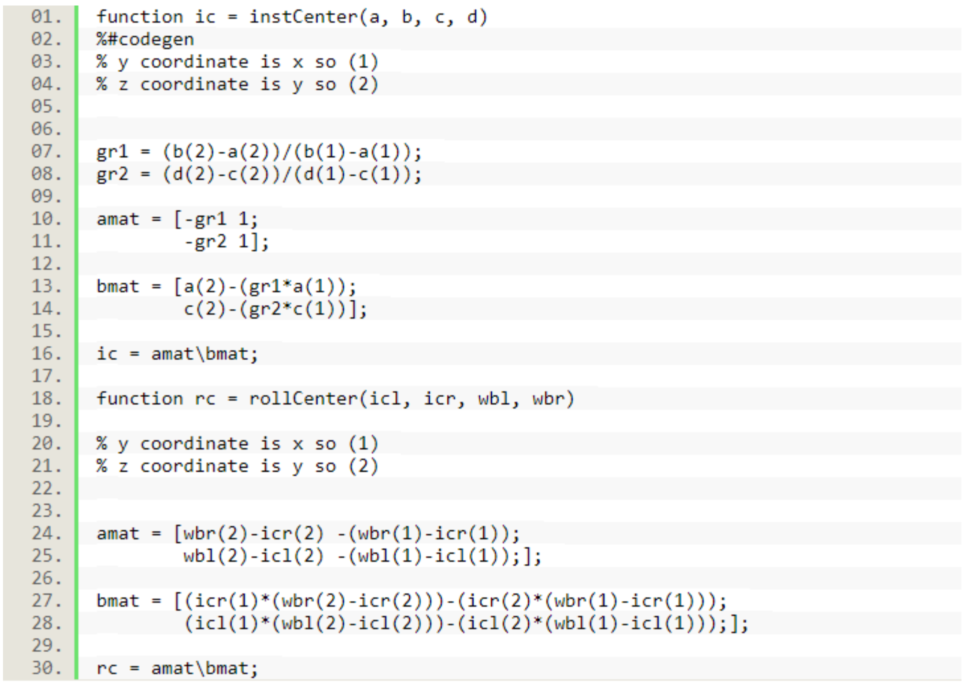


Figure 2.3.28 Roll center and instantaneous center

To calculate the roll center and instantaneous center, we can use Matlab function block. We solve the trigonometric equations based on the parameters measured in the model. The function of calculating instantaneous center and roll center is shown as Figure 2.3.29. Parameters a,b,c,d, wb are the component positions of the suspension.



```
function ic = instCenter(a, b, c, d)
%#codegen
% y coordinate is x so (1)
% z coordinate is y so (2)


gr1 = (b(2)-a(2))/(b(1)-a(1));
gr2 = (d(2)-c(2))/(d(1)-c(1));

amat = [-gr1 1;
        -gr2 1];

bmat = [a(2)-(gr1*a(1));
        c(2)-(gr2*c(1))];

ic = amat\bmat;

function rc = rollCenter(icl, icr, wbl, wbr)

% y coordinate is x so (1)
% z coordinate is y so (2)


amat = [wbr(2)-icr(2) -(wbr(1)-icr(1));
        wbl(2)-icl(2) -(wbl(1)-icl(1));];

bmat = [(icr(1)*(wbr(2)-icr(2)))-(icr(2)*(wbr(1)-icr(1)));
        (icl(1)*(wbl(2)-icl(2)))-(icl(2)*(wbl(1)-icl(1)));];

rc = amat\bmat;
```

Figure 2.3.29 Formular

After defining the position of roll center and instantaneous center, using a visualization subsystem to visualize the points. We simply feed in the center position and actuate a solid sphere through a planar joint to move it in the Y Z plane. The visualization subsystems are individual mechanical systems by themselves that are used only for the visualization and do not have any effect n the rest of the system. Then our model become as Figure 2.3.30

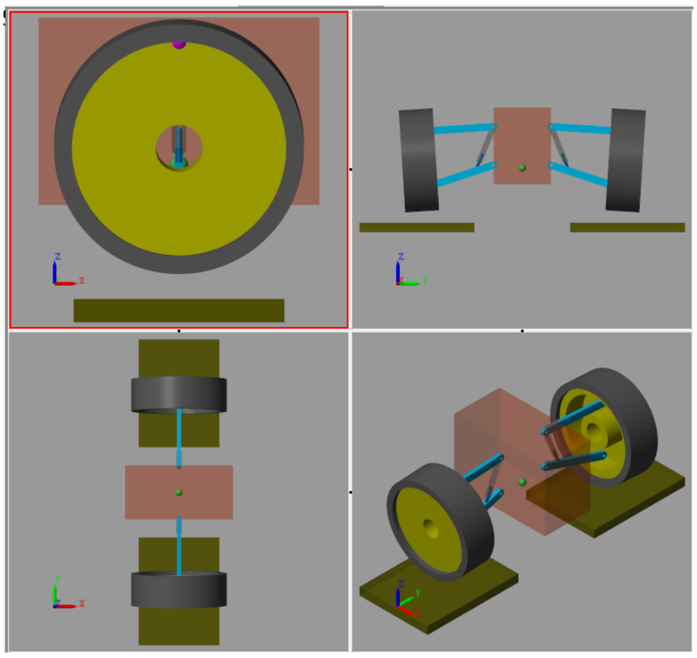


Figure 2.3.30

Then we have to optimize our model. First, we have to constrain the camber angle of the wheels between $-1° \; and - 7°$. Camber angle of a wheel is measured with respect to the vertical. The orientation of camber angle is shown as Figure 2.3.31. a negative camber gives the better grip when cornering.

So we have to sense the signal of camber angle using the Transform Sensor ad calculate the camber angle. Shown as Figure 2.3.32

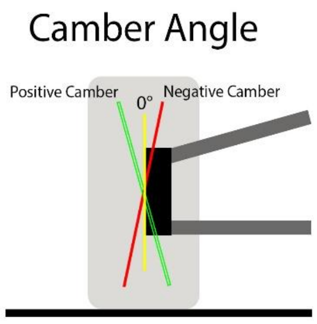


Figure 2.3.31 Camber angle

Second, constrain the Role Center height.

To do these, we have to know the position of the connection point between the upper arm and the chassis which is controlled by a distance variable that

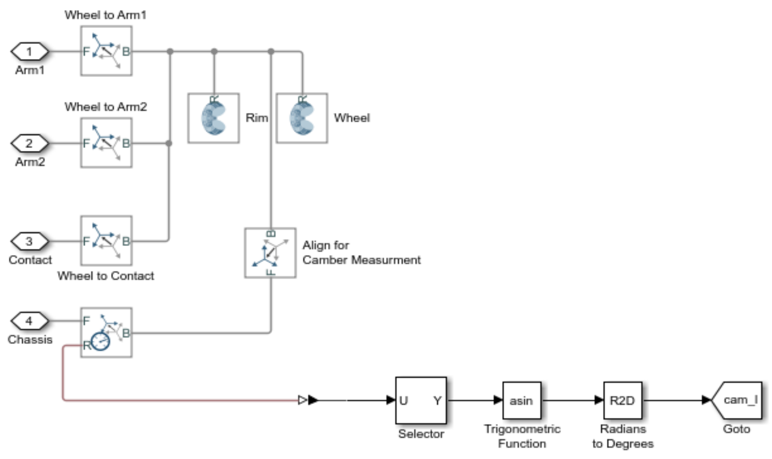


Figure 2.3.32 Monitor camber angle

specifies how far the upper arm is positioned from the chassis center of gravity. Also, we have to know the length of upper arm itself, changing this value we change the geometry of the suspension system and have an effect on both camber angle and the roll center height.

There are two types of optimization techniques:

1. Parameter Estimation
   We use experimental data to fit model parameters. The goal is to improve our design such that the simulation is more likely to match the data provided. Used to tune plant model parameters to match data.
2. Response Optimization
   We use performance matrix such as cost functions to tune algorithm parameters include constraining signals, reducing settling time, reducing overshoot. This method can be used to tune both plant and algorithm models. For our example, we use this method to constrain the roll center and camber angle.

The optimization flow shown as Figure 2.3.24 begins with simulating the model with a set of initial values for our design parameters such as upper arm length. The model outputs are evaluated with user specified requirements such as camber angle and roll center height. This produces a cost function which have to be minimized. As long as the requirements have not been achieved, the model parameters are modified based on the optimization flow and simulation runs with new values and the cost function is evaluated again

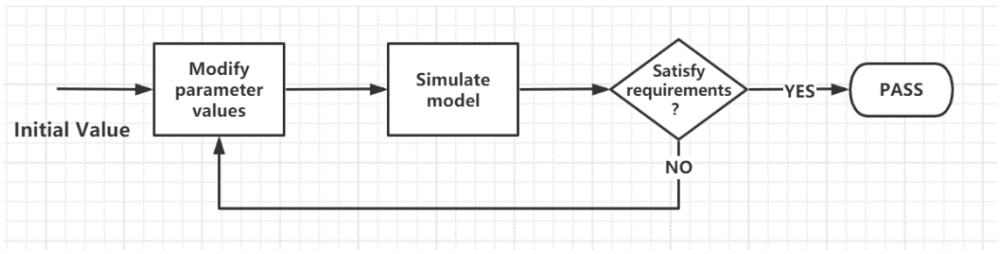


Figure 2.3.24 Optimization flow

To use the Response optimization approach, we have to install Optimization Toolbox addons in Matlab. Then we have to define which variables will be used for the optimizations and what requirements to satisfy. To define which parameters are tunable by the optimization algorithm select New from the Design Variable Set and select the variable which is the distance between suspension arm and chassis and the length of suspension arm. Due to the symmetry of the model, 1 variable can control both sides.

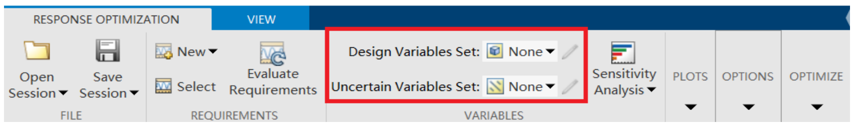


Figure 2.3.25 Select tunable variables

It is recommended that we provide meaningful minimum and maximum values for the parameters to restrict the search space. For example, the connection point between arm and chassis does not have the full freedom to placed on the chassis because it is constrained by the chassis design. Similarly, the arm length itself needs to be constrained by track length or the width of the vehicle.

Then we need to define the requirements for the optimization to achieve. First, set the constraint for the camber angle. Inside the Requirement section, for our application, select Signal Bound since we are constraining a time domain signal under a particular value. Then we have to define the upper and lower bound and choose the signal that can represents the camber angle. Since our model is symmetric, we can choose the signal from either right wheel or the left wheel. For setting up requirements for the roll center height, just do the same process, then we simulate the model and obverse the result as Figure.2.3.26.

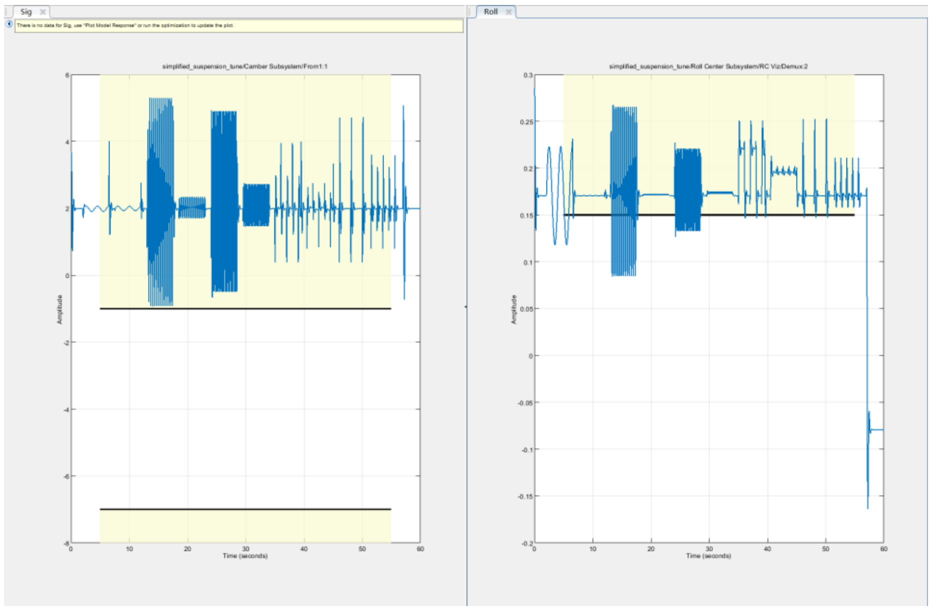


Figure 2.3.26 Before optimization

Left: Camber angle Right: Roll center height

From the plot, we can notice that, both signals exceed the threshold. Which means the current parameter does not satisfy the requirement. So, the next step is to optimize these parameters. The optimization progress report will give us information about the iteration. The optimization tool runs the model evaluates cost function based on the design requirements, changes our selected design parameters based on the default optimization algorithm and then runs the model again until the requirements are met. Once we see from the progress report that the optimization has converged and successful, we can observe the plot, shown as Figure 2.3.27.

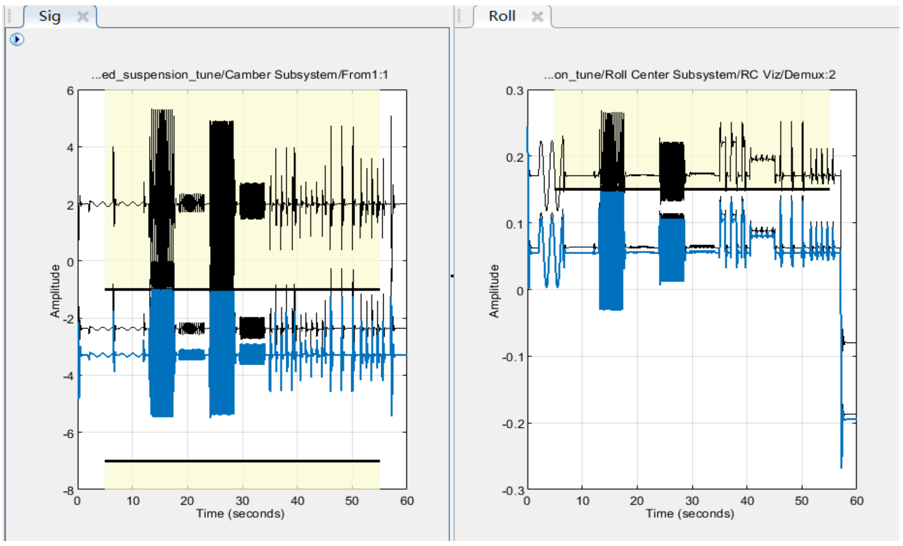


Figure 2.3.27 After optimization

Left: Camber angle Right: Roll center height

In the plot, we can see the signal values for different iterations, the blue plot is the final optimized response we can notice that it is within the signal threshold, so the optimization was successful. The optimized value is stored and modified in the Matlab workspace.

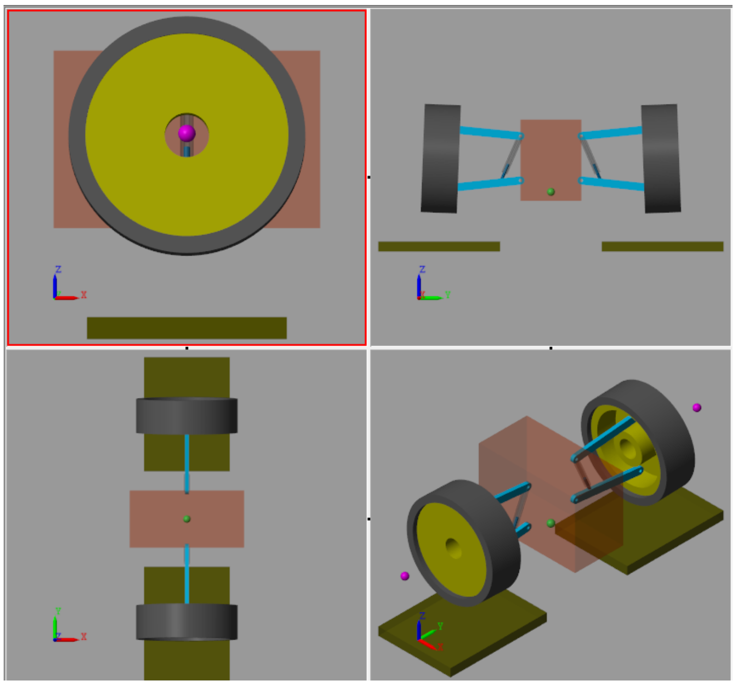


Figure 2.3.28 Final configuration

Compared to initial configuration, the final configuration has a shorter upper arm resulting in the wheels pointing towards the suspension which means a negative camber angle. Also, the upper arm connection point is moved upwards, the roll center is moved downwards.

2.10 Import CAD model

We have built a suspension using Simscape Multibody, it allows us to create our own custom components. However, mechanical systems are commonly designed using CAD software. We can use CAD models to automatically generate Simscape Multibody block diagrams. This allows us to do various operation with a CAD model such as data logging, analysis, dynamic simulation, control design, parameter optimization, automatic C/C++ code generation etc.

There are 2 import workflows:

1. Importing individual parts
   Here we already have a Multibody model and would like to import an individual CAD part to enhance the existing model.
2. Importing assemblies
   In this case we import the entire assembly from the CAD software including degrees of freedom, inertia properties etc.

Let us import the individual parts at first. Most CAD platforms allow us to save each part of the model as a STEP or a STL file. This allows us to use these files to define solid geometries. As the model is export from CAD, we can use File Solid block to import the file. File Solid is a block that Represents a solid whose geometry, material and visual properties are read from a file which could be of CATIA, NX, SolidEdge and other formats. The File Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that we specify. Then we can this solid as any other components, create frames to integrate with the existing mechanics system. Shown as Figure 2.3.29

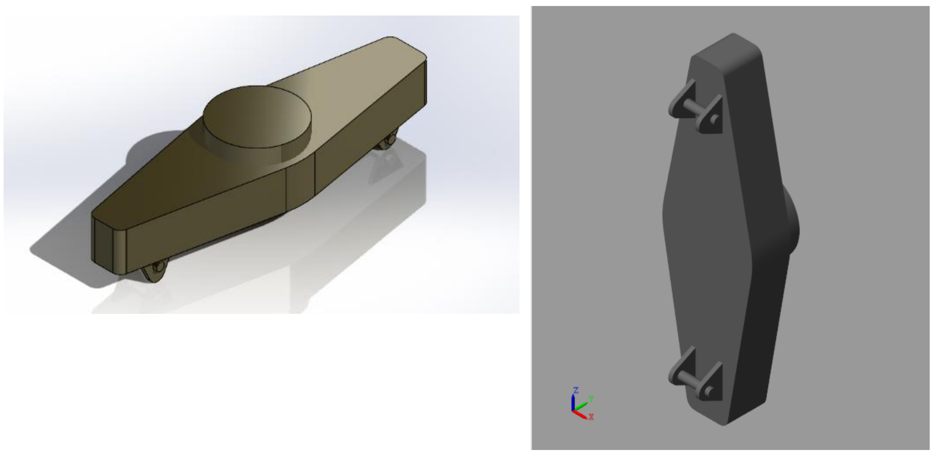
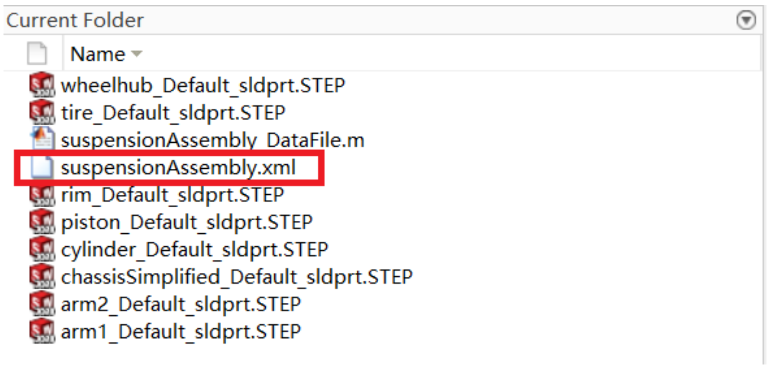


Figure 2.3.29 Import individual part

Left: Wheel hub in SolidWorks   Right: Wheel hub imported to Simscape

To export the assemblies, we have to use the Simscape Multibody link add-on. Simscape Multibody Link can automatically convert a CAD assembly into a sim mechanics block diagram. The CAD platforms supported in Simscape Multibody link are Autodesk Inventor®, Creo™ Parametric and SolidWorks®. After installed the add-on, software can export a CAD model into XML and graphics files. Shown as Figure 2.3.30.


Current Folder

Name

wheelhub_Default_sldprt.STEP
tire_Default_sldprt.STEP
suspensionAssembly_DataFile.m
suspensionAssembly.xml
rim_Default_sldprt.STEP
piston_Default_sldprt.STEP
cylinder_Default_sldprt.STEP
chassisSimplified_Default_sldprt.STEP
arm2_Default_sldprt.STEP
arm1_Default_sldprt.STEP

Figure 2.3.30 Exported assemblies

One STEP file per rigid part, XML file which contains information such as units, coordinate transforms, constraints, inertial properties, colors etc.

In order to input the assemblies, we need to use command *smimport.* Notice that the only a subset of CAD constraints can be translated. If the command is unable to translate a CAD constraint, it will assume a rigid connection. When this occurs, we will receive a warning.

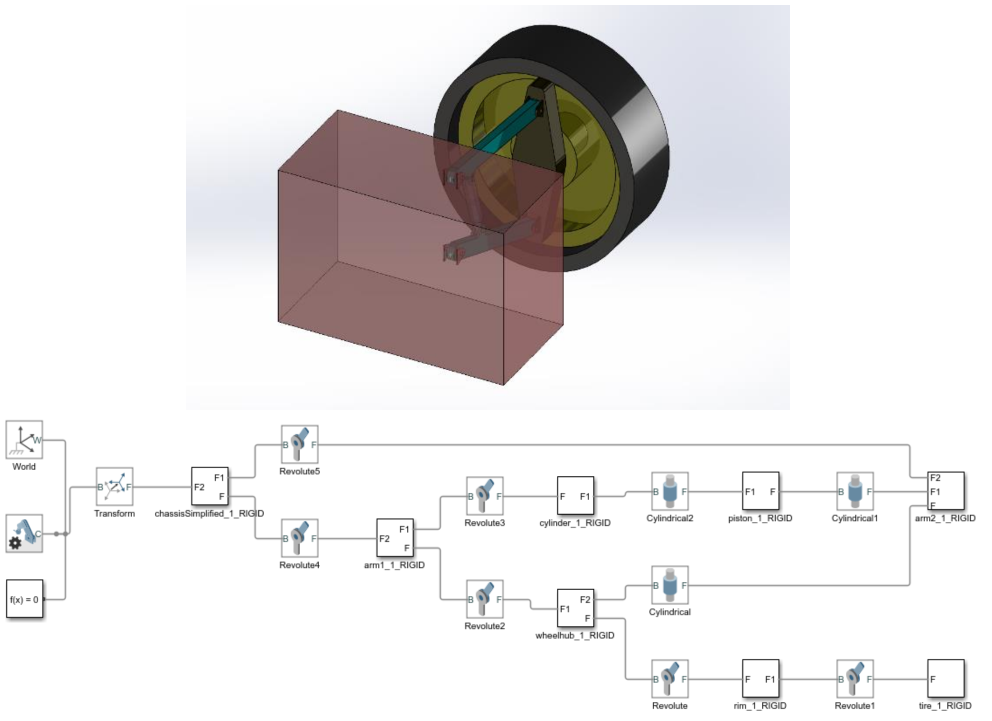


Figure 2.3.31 Import assemblies

Top: Assembly model in SolidWorks

Bottom: Converted into Simscape Multibody model

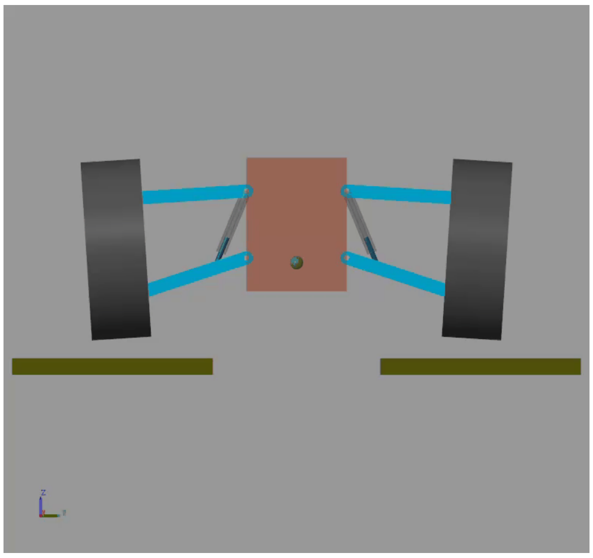
From Figure 2.3.31we can see that 3 essential blocks used in every sim model. Also, we can see the different subsystems, each component is well defined. The constraints in our assembly have been translated into mechanical joint block. There are some cylindrical joints in the model do not have any effect on the original constraints of our system in the CAD model. These joints are connected for all practical purposes, they are just fundamental revolute. And in the case of the connection between the cylinder and piston it is prismatic.

In order to let the model, work properly, we have to modify the gravity vector orientation to negative Z-axis direction. Notice that, the internal mechanical properties are not translated by Simscape Multibody Link, which means we have to specify the stiffness or damping of the shock absorber joints. The joints were modeled in an ideal way. The following Table 2.3.1 shows the feature that can be captured by Simscape Multibody Link

Mechanical Element	Translatable by Simscape Multibody Link?
Geometric properties	Yes
Inertial properties	Yes
Coordinate transforms	Yes
Joints	Yes
Distance/Angle constraints	Yes
Graphic properties	Yes
Initial conditions	Can translate position cannot translate velocity
External mechanics	No
Internal mechanics	No
Sensing and actuation	No

Table 2.3.1 Features of Simscape Multibody Link

After refining the model, we can add sensor, actuator, connect to other models or the external force test subsystem.



3. Conclusion

Simscape extends Simulink with libraries for modelling and simulating multi-domain physical systems. It is built on the top of Matlab and Simulink, contains models of foundation elements for various physical domains. also, contains other capabilities which is useful when modeling physical system (such as unit manager, data logging, etc). MathWorks offers a suite of physical modeling products built on top of Simscape for specialized modeling of multi-domain physical systems. For example, the Simscape Multibody is used to model 3D systems.

With Simscape we can create the model using the physical network method where deriving the system level equation is not required. This makes the model easier to create, understand and maintain.

4. Reference

1. *MathWorks Simscape Multibody Introduction* from
   https://www.mathworks.com/products/simmechanics.html?s_tid=srchtitle

2. MathWorks Student Competitions Team. (2017). *Physical Modeling for Formula Student: Simscape Introduction* from
   https://www.mathworks.com/matlabcentral

3. *MathWorks Simscape Multibody Help Center* from
   https://www.mathworks.com/help/physmod/sm/

4. Curtis Dietzsch. (2014). *Finding Your Center – Finding Your Front and Rear Roll Center* from
   https://www.onedirt.com/tech/chassis-suspension/finding-your-center-finding-your-front-and-rear-roll-center/

5. *What is camber?* from
   https://help.summitracing.com/app/answers/detail/a_id/5256/~/what-is-camber%3F

6. *Multibody simulation* from
   https://en.wikipedia.org/wiki/Multibody_simulation

7. *RecurDyn Overview* from
   https://support.functionbay.com/en/page/single/2/recurdyn-overview

8. *Adams Introduction* from
   https://www.mscsoftware.com/product/adams

9. Dion Besselink. *Multibody dynamics* from
   https://www.code-ps.com/services/multibody-dynamics.html