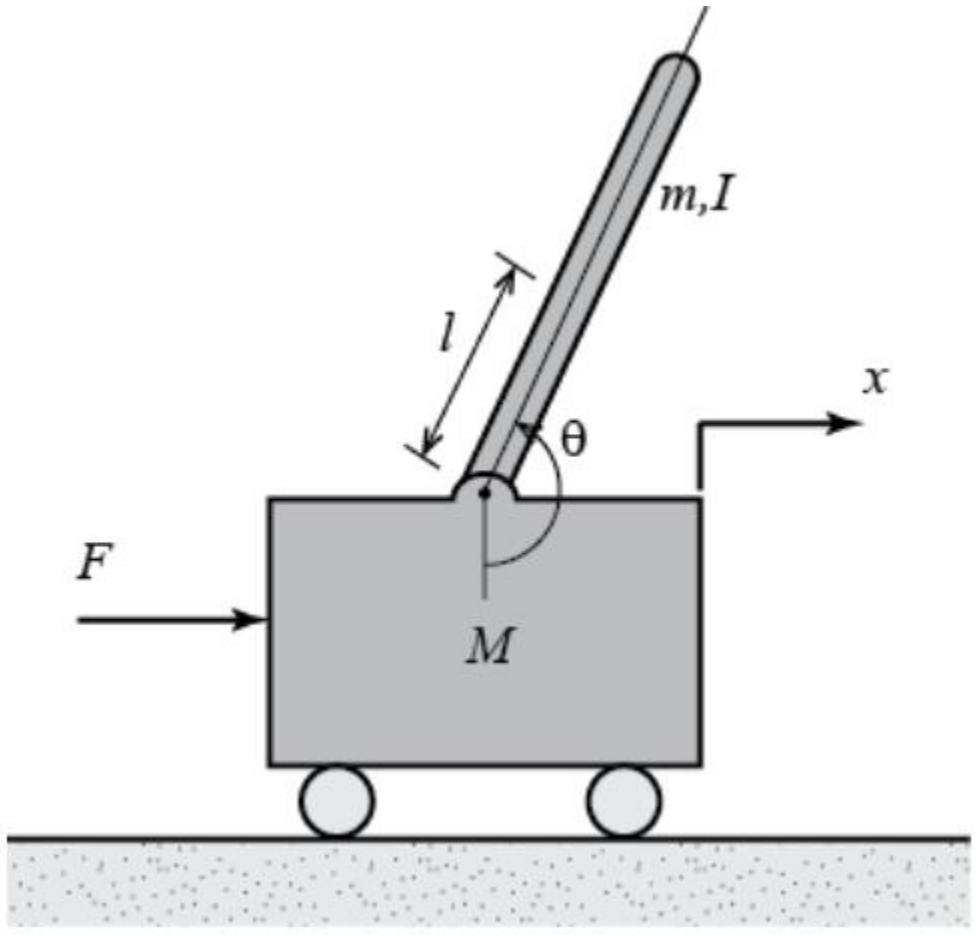
Table of Contents

1. Physical Setup

In this section we show how to build the inverted pendulum model using the physical modeling blocks of Simscape Multibody.



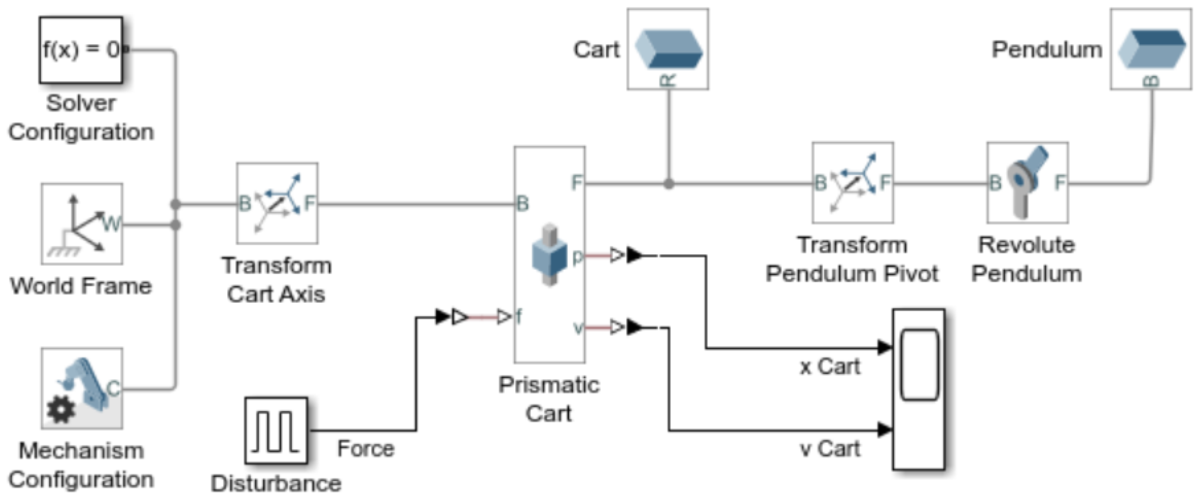
The system parameters are defined as follows:

(M)	mass of the cart	0.5 kg
(m)	mass of the pendulum	0.2 kg

(b)	coefficient of friction for cart	0.1 N/m/sec
(l)	length to pendulum center of mass	0.3 m
(I)	mass moment of inertia of the pendulum	0.006 kg.m^2

2. Simscape Multibody

First I set gravity to "[0 0 -9.81]" acting along the global -Z direction. Open the Solver Configuration block and ensure that the Use local solver checkbox is not selected. Then I assemble base plane and cart. I will model the cart as a point mass moving along an axis. We use the World Frame to define the axis along which the cart will travel. Then we add prismatic and revolute joints and submit along with their reference frames. Pendulum subsystem and connecting the cart to the pendulum are the next step of the simulation. The model at this point should now appear as follows.



3. Selecting outputs for controller and angle conversion

I now need to measure the angle and angular velocity of the pendulum:

- Double-click on "Revolute Pendulum"

- In group **Z Revolute Primitive (Rz)** under **Sensing**, select checkboxes for **Position** and **Velocity**
- Make two copies of the PS-Simulink converter block
- Double-click on one PS-Simulink block and set **Output signal units** to "rad"
- Connect that PS-Simulink block to the q port on Revolute Pendulum
- Double-click on the other PS-Simulink block and set **Output signal units** to "rad/s"
- Connect that PS-Simulink block to the w port on Revolute Pendulum

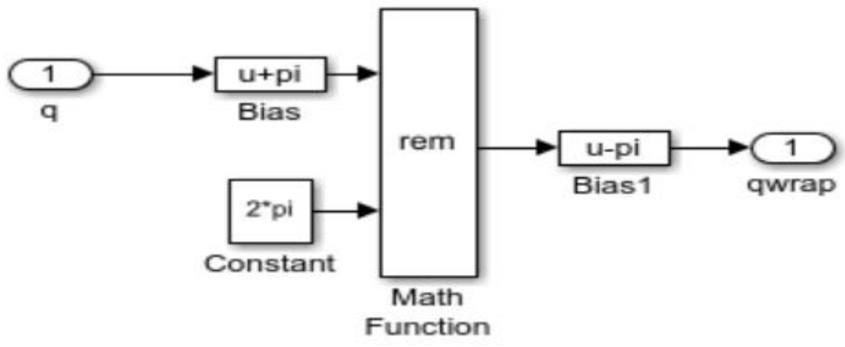
We need to limit the measured angle to stay between -pi and pi radians.

- Add a Subsystem block to your model
- Rename the Subsytem block "Wrap Angle"
- Double-click to enter the Wrap Angle subsystem

In this subsystem we will add pi radians to the measurement, find the remainder when the signal is divided by 2*pi, and then subtract pi radians.

- Rename the input block "q"
- Delete the signal connection between the inport and the outport
- Add a Bias block to the model
- Set parameter **Bias** to "pi"
- Connect the q block to the Bias block
- Add a Math Function block to the model
- Double-click on the Math Function block and set **Function** to "rem"
- Connect the output of Bias to the first input of the Math Function block
- Add a Constant block to the model
- Set parameter **Constant** to "2*pi"
- Connect the Constant block to the second input of the Math Function
- Make a copy of the Bias block
- Set parameter **Bias** in the new block to "-pi"
- Connect Math Function output to the input of the new Bias block
- Connect the output of the new Bias block to the outport
- Rename the outport "qwrap"
- Go up one level in the diagram and rename the subsytem "Wrap Angle"

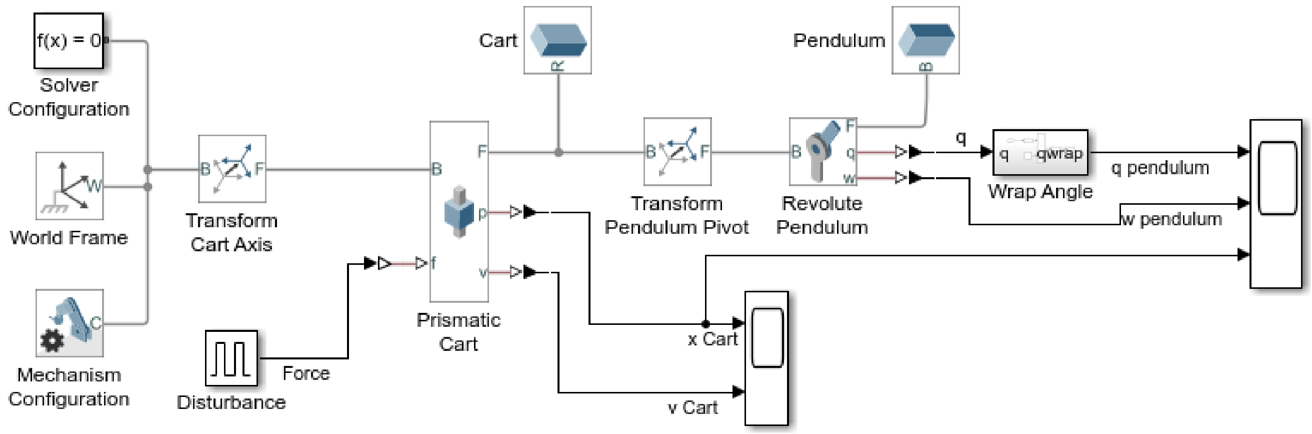
Here you can see the resulting subsystem for wrapping the angle.



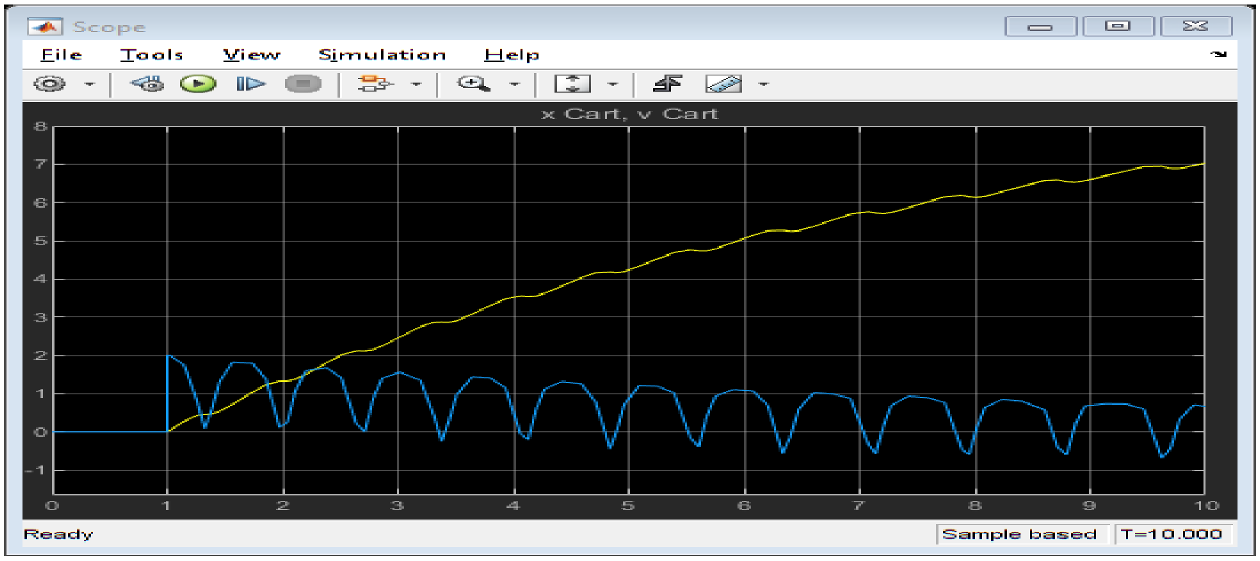
I will plot the new signals on a Scope

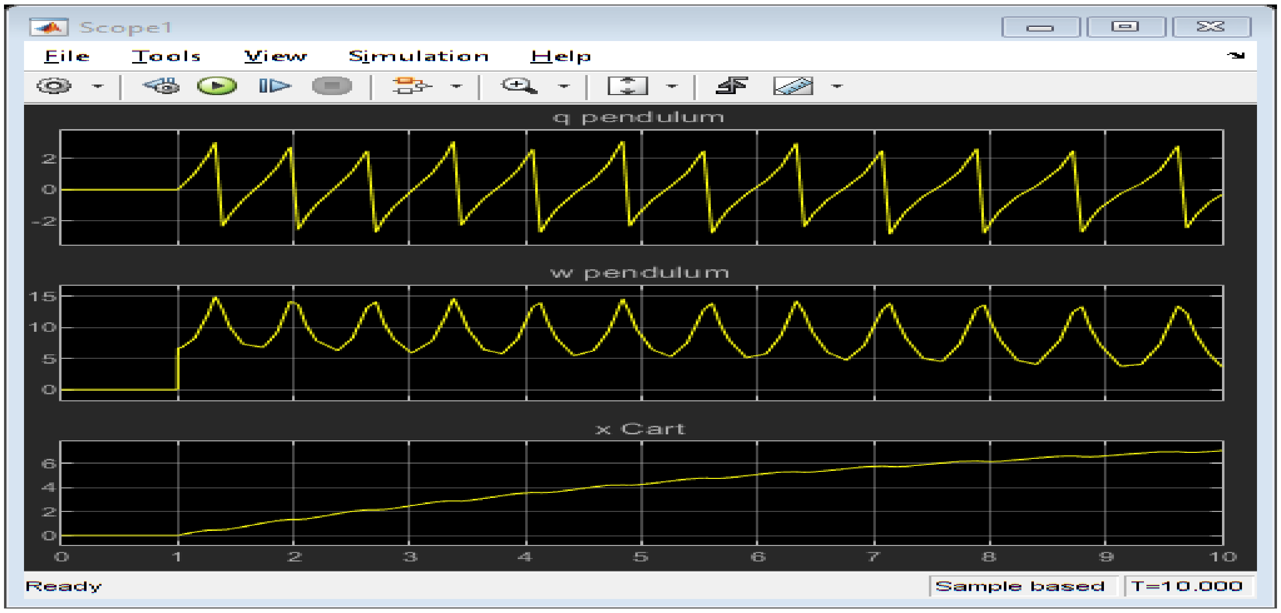
- Connect PS-Simulink output for the q measurement of Revolute Pendulum to the input of Wrap Angle
- Add a new Scope
- Connect the qwrap output of Wrap Angle to the new Scope and change the name of this signal to "q pendulum"
- Connect the PS-Simulink output for the w measurement of Revolute Pendulum to the new Scope and change the name of this signal to "w pendulum"

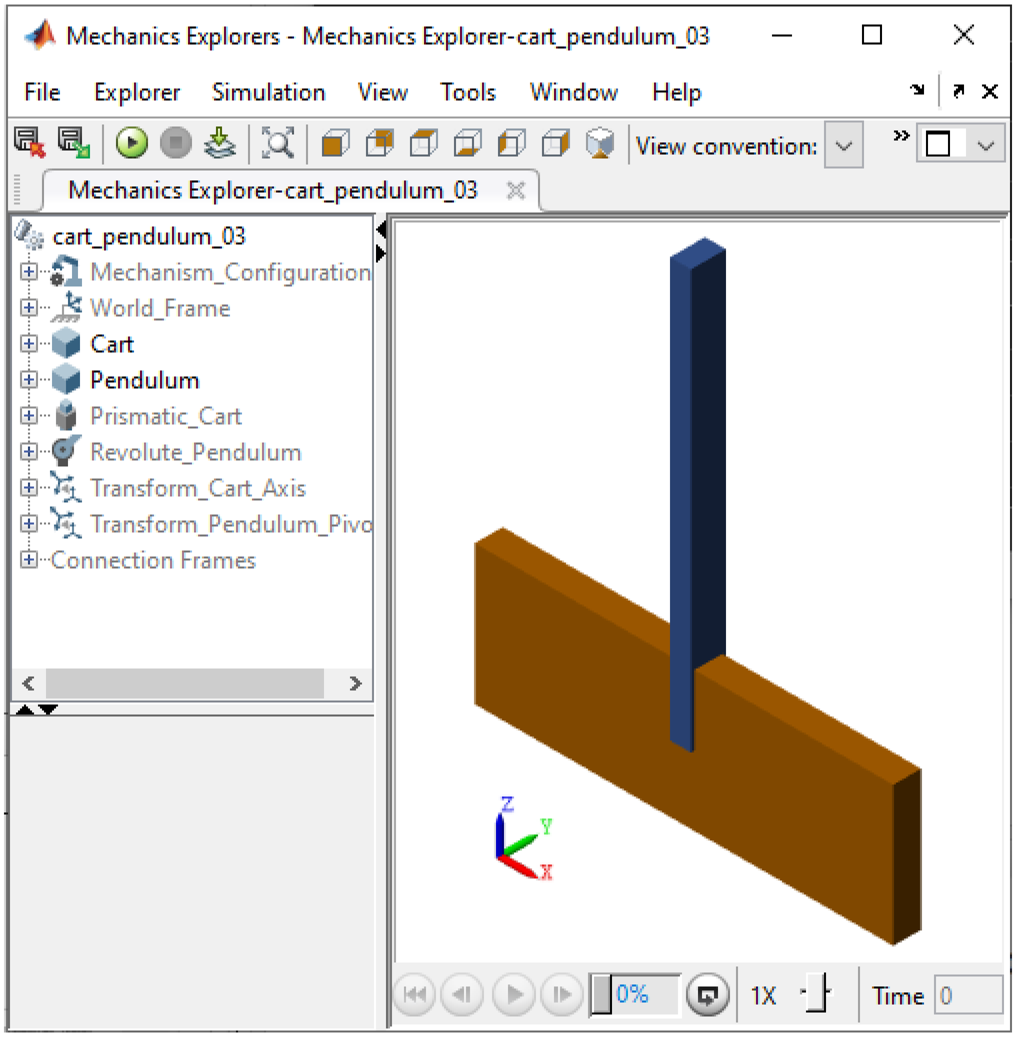
The resulting model should appear as follows:



Running a simulation (type **CTRL-T** or press the green arrow run button), the followings plot are generated. The motion of the cart is the same as before, but now we can see the motion of the pendulum. The associated animation shown below is also generated.




Scope1
File Tools View Simulation Help
q pendulum
2
0
-2
w pendulum
15
10
5
0
x Cart
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000

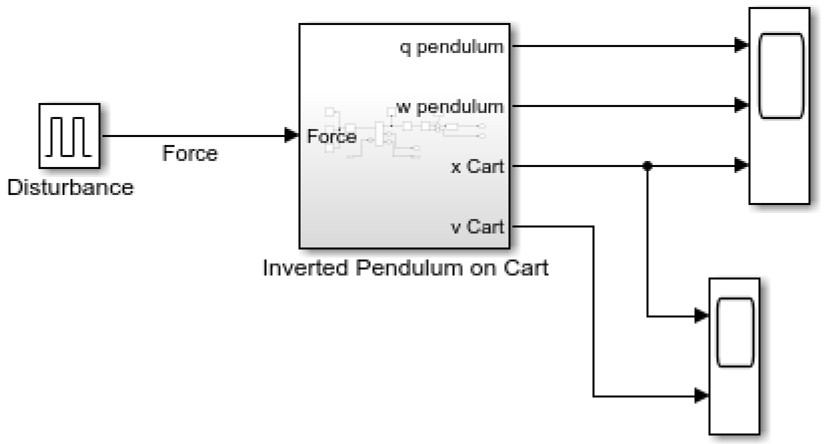

Mechanics Explorers - Mechanics Explorer-cart_pendulum_03
File Explorer Simulation View Tools Window Help
View convention:
Mechanics Explorer-cart_pendulum_03
cart_pendulum_03
Mechanism_Configuration
World_Frame
Cart
Pendulum
Prismatic_Cart
Revolute_Pendulum
Transform_Cart_Axis
Transform_Pendulum_Pivo
Connection Frames
Z
Y
X
0%
1X
Time 0

4. Create subsystem for pendulum and cart

I have now successfully created all the elements of the inverted pendulum system. We will convert this into a subsystem.

- Click once in the diagram (but not on a block) and press **CTRL-A** to select all blocks
- Hold the **Shift** key and click on the Disturbance block and each Scope to unselect those blocks
- Press **CTRL-G** to create a subsystem
- Rename the subsystem "Inverted Pendulum on Cart"

Your model should now appear as shown:



5. Closed-loop setup

I will now add blocks for open and closed-loop testing.

Add the following blocks:

- Subtract
- PID Controller
- Constant
- Manual Switch
- Sum

The Disturbance also will be added to the control signal.

- Delete the signal connecting Disturbance to the cart subsystem
- Connect the output of the sum block to the Force input of the cart subsystem
- Connect Disturbance to the bottom + port of the Sum block

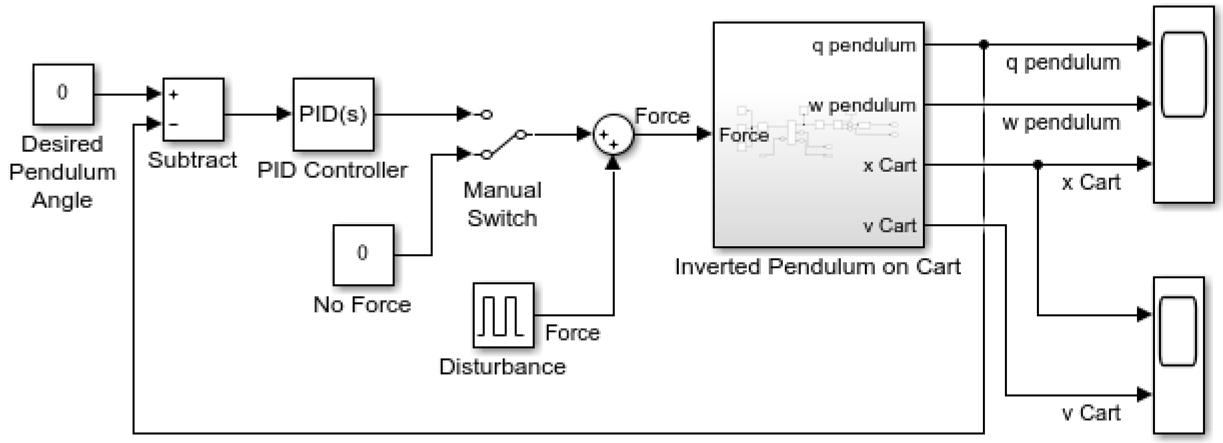
We wish to manually select open or closed-loop behavior.

- Connect the Manual Switch output to the + input of the Sum block
- Connect the Constant block to the lower input of the Manual Switch, then set the parameter **Constant** to "0" and rename block "No Force"
- Connect the output of the PID Controller to the upper input of Manual Switch
- Connect the Subtract block output to the input of PID Controller

I will close the loop to control the pendulum angle.

- Connect q pendulum output of the cart subsystem to the - port of the Subtract block
- Make a copy of the Constant block and connect it to the + port of the Subtract block and rename the block "Desired Pendulum Angle"

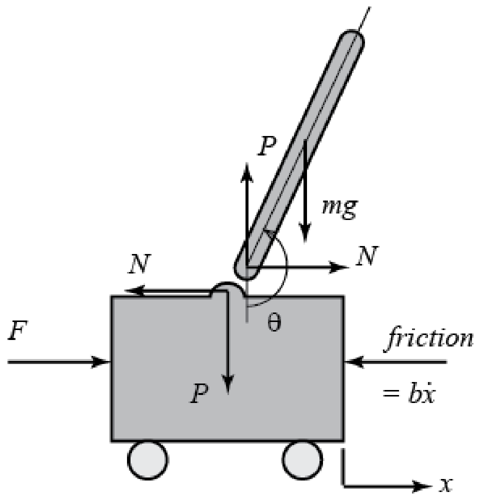
Your model should now appear as follows. The output of the simulation is unchanged from prior results when in open-loop mode.



6. Controller Design

6. 1 System Modeling

Below are the free-body diagrams of the two elements of the inverted pendulum system.



Summing the forces in the free-body diagram of the cart in the horizontal direction, you get the following equation of motion.

$$M\ddot{x} + b\dot{x} + N = F$$

Note that you can also sum the forces in the vertical direction for the cart, but no useful information would be gained.

Summing the forces in the free-body diagram of the pendulum in the horizontal direction, you get the following expression for the reaction force N.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

If you substitute this equation into the first equation, you get one of the two governing equations for this system.

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F$$

To get the second equation of motion for this system, sum the forces perpendicular to the pendulum. Solving the system along this axis greatly simplifies the mathematics. You should get the following equation.

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta$$

To get rid of the P and N terms in the equation above, sum the moments about the centroid of the pendulum to get the following equation.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta}$$

Combining these last two expressions, you get the second governing equation.

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta$$

Since the analysis and control design techniques we will be employing in this example apply only to linear systems, this set of equations needs to be linearized. Specifically, we will linearize the equations about the vertically upward equillibrium position, $\theta = \pi$, and will assume that the system stays within a small neighborhood of this equillbrium. This assumption should be reasonably valid since under control we desire that the pendulum not deviate more than 20 degrees from the vertically upward position. Let ϕ represent the deviation of the pedulum's position from equilibrium, that is, $\theta = \pi + \phi$. Again presuming a small deviation (ϕ) from equilibrium, we can use the following small angle approximations of the nonlinear functions in our system equations:

$$\cos\theta = \cos(\pi + \phi) \approx -1$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

After substituting the above approximations into our nonlinear governing equations, we arrive at the two linearized equations of motion. Note u has been substituted for the input F.

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u$$

6. 2 Transfer Function

To obtain the transfer functions of the linearized system equations, we must first take the Laplace transform of the system equations assuming zero initial conditions. The resulting Laplace transforms are shown below.

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2$$

$$(M + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s)$$

Recall that a transfer function represents the relationship between a single input and a single output at a time. To find our first transfer function for the output $\Phi(s)$ and an input of $U(s)$ we need to eliminate $X(s)$ from the above equations. Solve the first equation for $X(s)$.

$$X(s) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)$$

Then substitute the above into the second equation.

$$(M + m)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s)$$

Rearranging, the transfer function is then the following

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s}$$

where,

$$q = [(M + m)(I + ml^2) - (ml)^2]$$

From the transfer function above it can be seen that there is both a pole and a zero at the origin. These can be cancelled and the transfer function becomes the following.

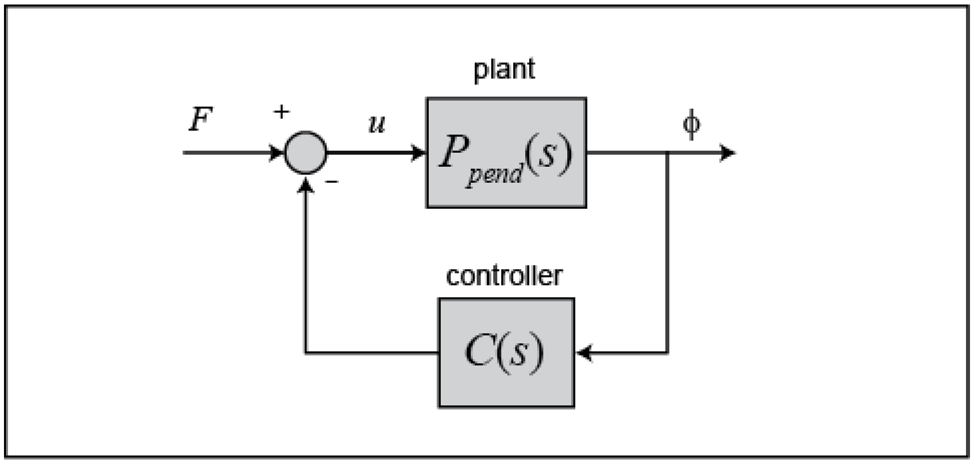
$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}]$$

Second, the transfer function with the cart position $X(s)$ as the output can be derived in a similar manner to arrive at the following.

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2-gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad [\frac{m}{N}]$$

6. 3 System structure

The structure of the controller for this problem is a little different than the standard control problems you may be used to. Since we are attempting to control the pendulum's position, which should return to the vertical after the initial disturbance, the reference signal we are tracking should be zero. This type of situation is often referred to as a Regulator problem. The external force applied to the cart can be considered as an impulsive disturbance. The schematic for this problem is depicted below.



The resulting transfer function $T(s)$ for the closed-loop system from an input of force F to an output of pendulum angle ϕ is then determined to be the following.

$$T(s) = \frac{\Phi(s)}{F(s)} = \frac{P_{pend}(s)}{1 + C(s)P_{pend}(s)}$$

We first create our MATLAB code for the transfer functions.

```
M = 0.5;

m = 0.2;

b = 0.1;

I = 0.006;

g = 9.8;

l = 0.3;

q = (M+m)*(I+m*l^2)-(m*l)^2;

s = tf('s');
P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*l)*s/q - b*m*g*l/q);
```

Next we will define a PID controller.

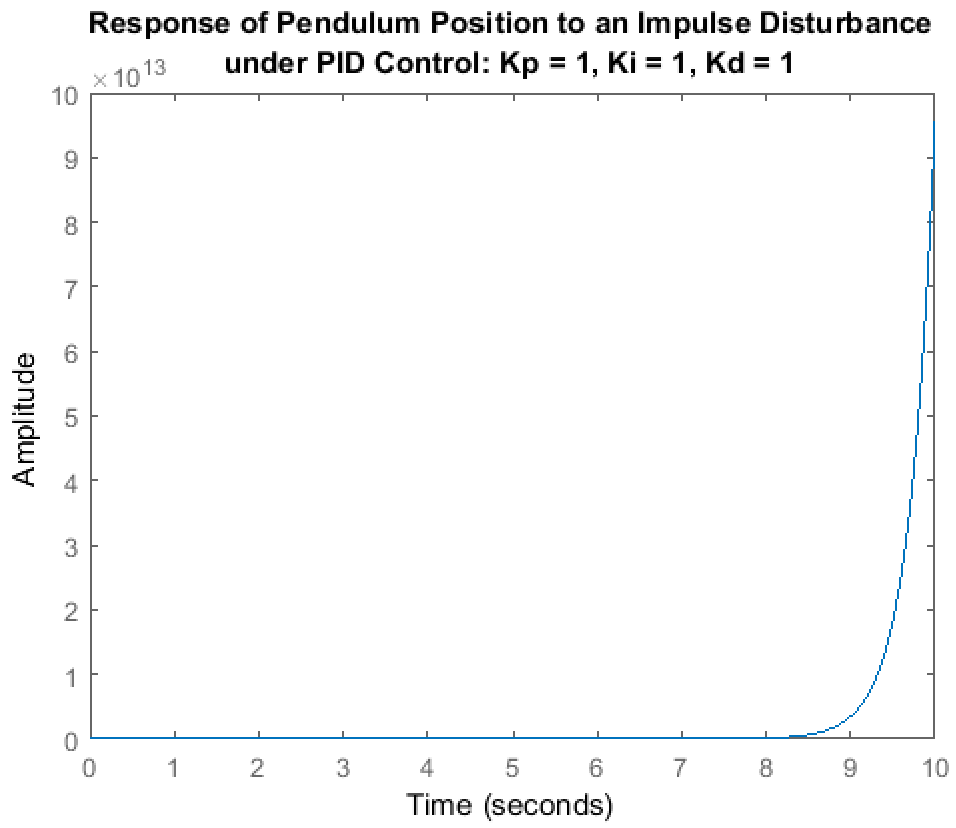
6. 4 PID control

This closed-loop transfer function can be modeled in MATLAB by copying the following code to the end of your m-file (whether you're using the transfer function form or the state-space representation of the plant). Specifically, we define our controller using the PID object within MATLAB. We then use the feedback command to generate the closed-loop transfer function as depicted in the figure above where the disturbance force is the input and the deviation of the pendulum angle from the vertical ϕ is the output.

```
Kp = 1;

Ki = 1;

Kd = 1;

C = pid(Kp,Ki,Kd);

T = feedback(P_pend,C);
```

Now we can begin to tune our controller. First let's examine the response of the closed-loop system to an impulse disturbance for this initial set of control gains. Enter the following code to the end of your m-file and run in the MATLAB command window. You should generate the response plot shown below.

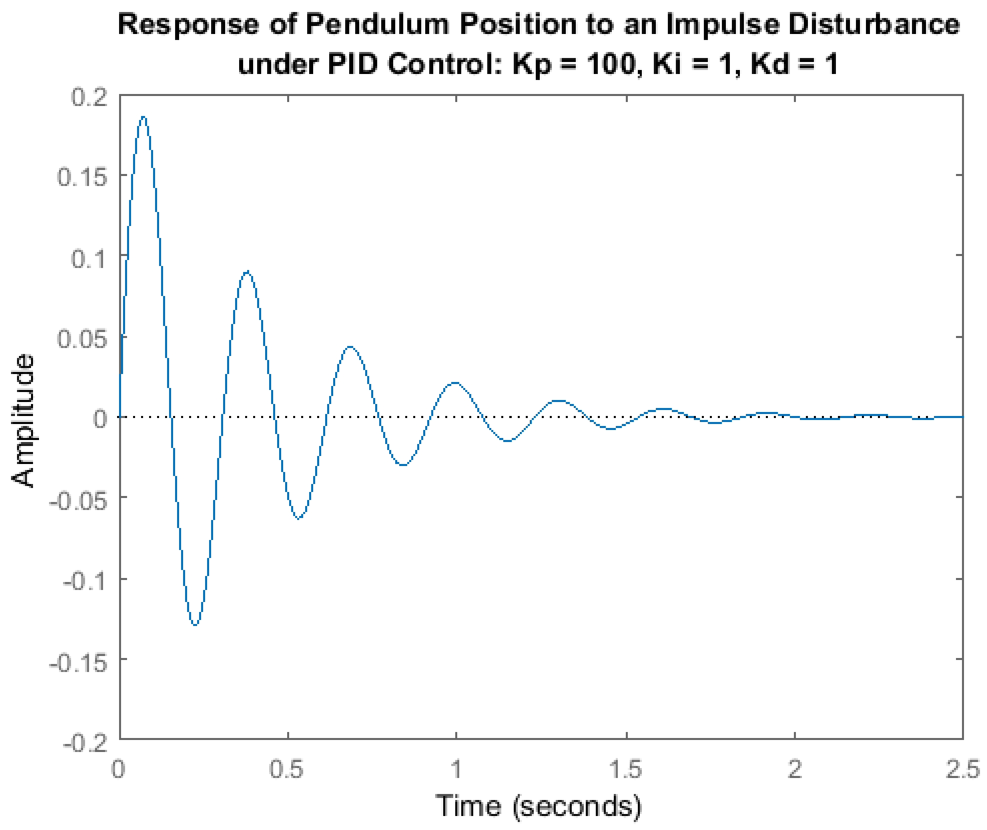
```
t=0:0.01:10;

impulse(T,t)

title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 1, Ki = 1, Kd = 1'});
```



This response is still not stable. Let's begin to modify the response by increasing the proportional gain. Increase the K variable to see what effect it has on the response. If you modify your m-file to the following and run in the command window, you should get the response plot shown below.

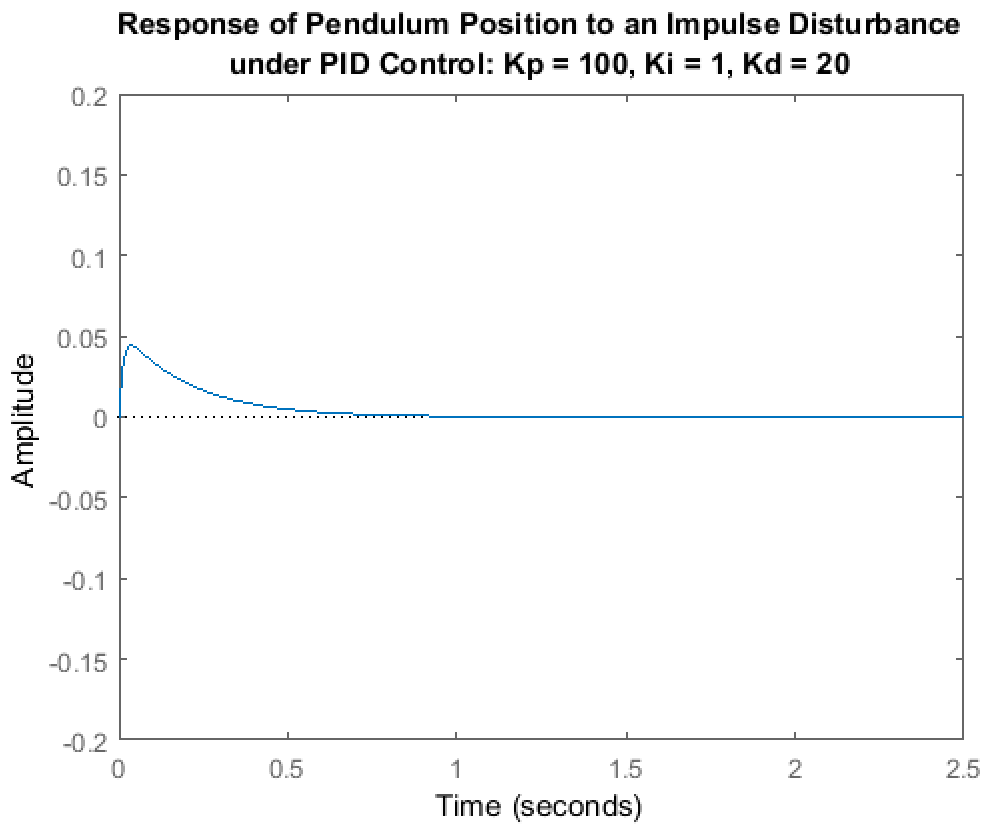
```
Kp = 100;

Ki = 1;

Kd = 1;
```

```
C = pid(Kp,Ki,Kd);

T = feedback(P_pend,C);

t=0:0.01:10;

impulse(T,t)

axis([0, 2.5, -0.2, 0.2]);

title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 100,
Ki = 1, Kd = 1'});
```



Right-clicking on the resulting plot and choosing **Characteristics** from the resulting menu allows you to identify important characteristics of the response. Specifically, the settling time of the response is determined to be 1.64 seconds, which is less than the requirement of 5 seconds. Since the steady-state error approaches zero in a sufficiently fast manner, no additional integral action is needed. You can set the integral gain constant to zero to see for yourself that some integral control is needed. The peak response, however, is larger than the requirement of 0.05 radians. Recall that overshoot often can be reduced by increasing the amount of derivative control. After some trial and error it is found that a derivative gain of K_d = 20 provides a satisfactory response. Modifying your m-file as follows and re-running should produce the response plot shown below

```
Kp = 100;

Ki = 1;

Kd = 20;

C = pid(Kp,Ki,Kd);

T = feedback(P_pend,C);

t=0:0.01:10;

impulse(T,t)

axis([0, 2.5, -0.2, 0.2]);

title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 100, Ki = 1, Kd = 20'});
```



As you can see, the overshoot has been reduced so that the pendulum does not move more than 0.05 radians away from the vertical. Since all of the given design requirements have been met, no further iteration is needed.

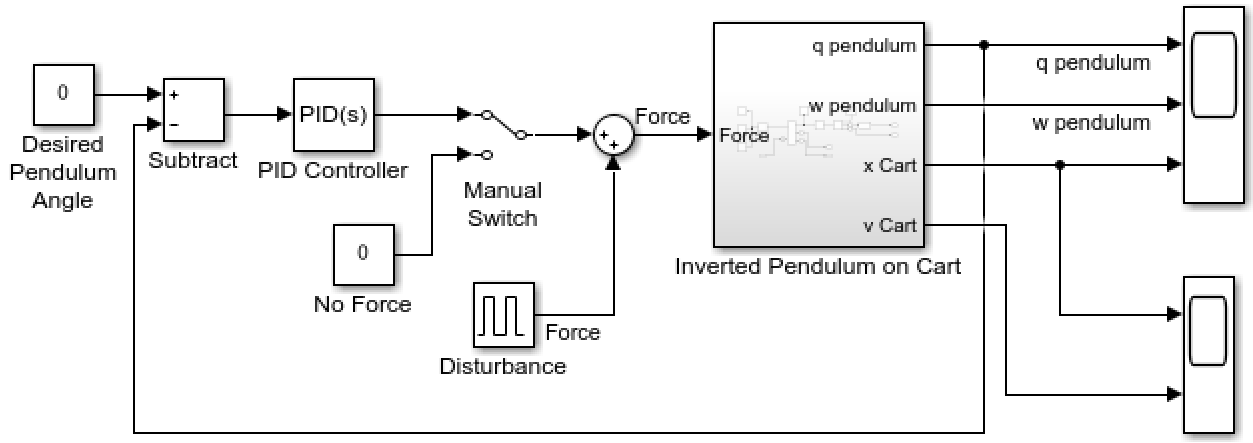
The cart moves in the negative direction with approximately constant velocity. Therefore, although the PID controller stabilizes the angle of the pendulum, this design would not be feasible to implement on an actual physical system.

7. Controller Implementation

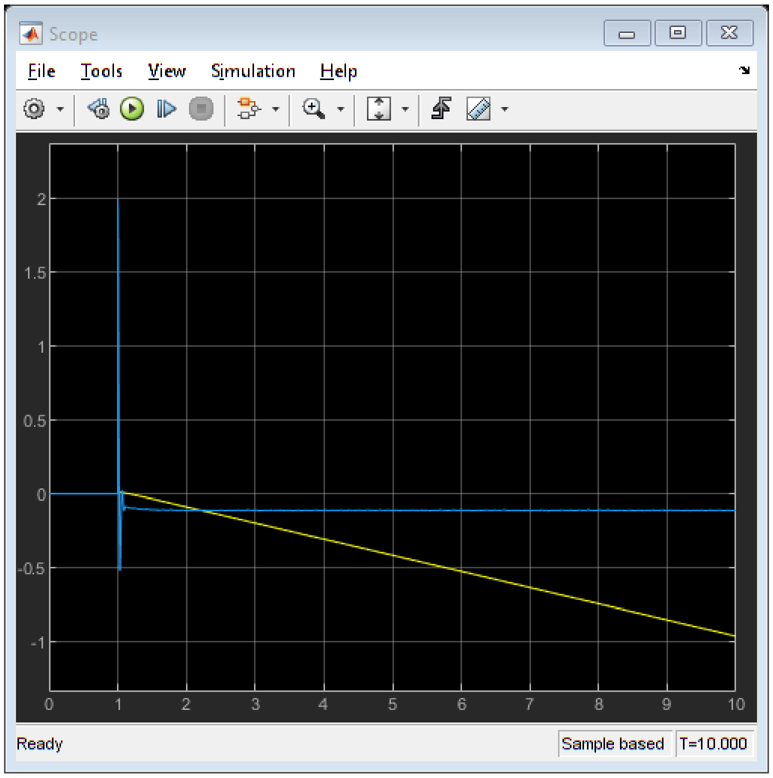
I will now implement the PID control gains developed in the

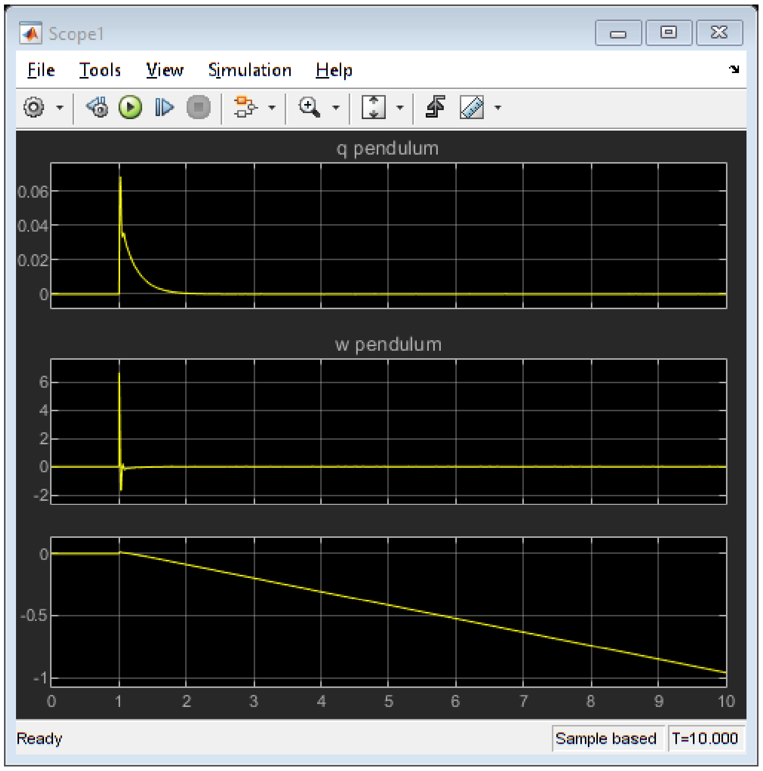
- Double-click on the PID Controller block
- Set parameter **Proportional (P)** to "100"
- Set parameter **Integral (I)** to "1"
- Set parameter **Derivative (D)** to "20"

Double-click on Manual Switch until the input from the PID Controller is selected.



Running a simulation produces the following two plots show the controlled response of the system. After the initial impact the controller was able to quickly bring down the pendulum angle to zero and the pendulum velocity is also zero. The cart moves slowly and with a constant velocity in the negative X direction to keep the pendulum balanced.


Scope
File Tools View Simulation Help
2
1.5
1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000


Scope1
File Tools View Simulation Help
q pendulum
0.06
0.04
0.02
0
w pendulum
6
4
2
0
-2
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000