

# Control Tutorials for MATLAB® and Simulink®

## Example: Modeling an Inverted Pendulum in Simulink

[Problem setup and design requirements](#)

[Force analysis and system equation setup](#)

[Building the model](#)

[Open-loop response](#)

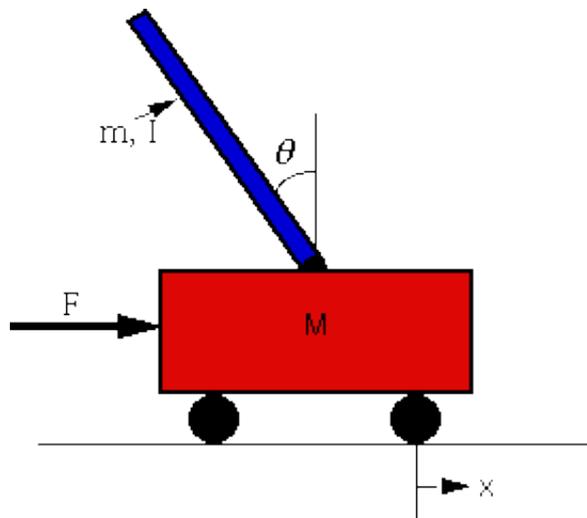
[Extracting a linearized model](#)

[Implementing PID control](#)

[Closed-loop response](#)

### Problem setup and design requirements

The cart with an inverted pendulum, shown below, is "bumped" with an impulse force,  $F$ . Determine the dynamic equations of motion for the system, and linearize about the pendulum's angle,  $\theta = 0$  (in other words, assume that pendulum does not move more than a few degrees away from the vertical, chosen to be at an angle of 0). Find a controller to satisfy all of the design requirements given below.



For this example, let's assume that

$M$	mass of the cart	0.5 kg
$m$	mass of the pendulum	0.2 kg
$b$	friction of the cart	0.1 N/m/sec
$l$	length to pendulum center of mass	0.3 m

I	inertia of the pendulum	0.006 kg*m <sup>2</sup>
F	force applied to the cart	
x	cart position coordinate	
theta	pendulum angle from vertical	

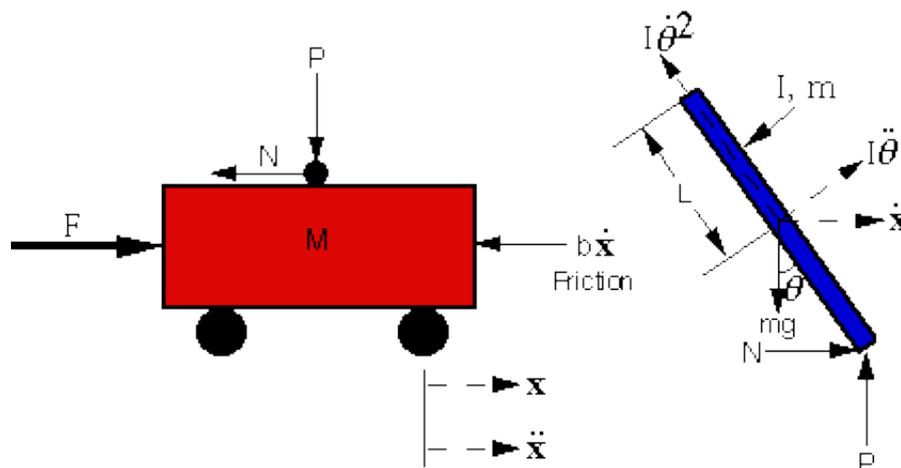
In this example, we will implement a PID controller which can only be applied to a single-input-single-output (SISO) system, so we will be only interested in the control of the pendulum's angle. Therefore, none of the design criteria deal with the cart's position. We will assume that the system starts at equilibrium, and experiences an impulse force of 1N. The pendulum should return to its upright position within 5 seconds, and never move more than 0.05 radians away from the vertical.

The design requirements for this system are:

- Settling time of less than 5 seconds.
- Pendulum angle never more than 0.05 radians from the vertical.

## Force analysis and system equation setup

Below are the two Free Body Diagrams of the system.



This system is tricky to model in Simulink because of the physical constraint (the pin joint) between the cart and pendulum which reduces the degrees of freedom in the system. Both the cart and the pendulum have one degree of freedom ( $X$  and  $\theta$ , respectively). We will then model Newton's equation for these two degrees of freedom.

$$\frac{d^2x}{dt^2} = \frac{1}{M} \sum_{\text{cart}} F_x = \frac{1}{M} \left( F - N - b \frac{dx}{dt} \right)$$

$$\frac{d^2\theta}{dt^2} = \frac{1}{I} \sum_{\text{pend}} \tau = \frac{1}{I} (NL \cos(\theta) + PL \sin(\theta))$$

It is necessary, however, to include the interaction forces  $N$  and  $P$  between the cart and the pendulum in order to model the dynamics. The inclusion of these forces requires modeling the  $x$  and  $y$  dynamics of the pendulum in addition to its  $\theta$  dynamics. In the MATLAB tutorial [pendulum modeling example](#) the interaction forces were solved for algebraically. Generally, we would like to exploit the modeling power of Simulink and let the simulation take care of the algebra. Therefore, we will model the additional  $x$  and

y equations for the pendulum.

$$m \frac{d^2 x_p}{dt^2} = \sum_{\text{pend}} F_x = N$$

$$\implies N = m \frac{d^2 x_p}{dt^2}$$

$$m \frac{d^2 y_p}{dt^2} = \sum_{\text{pend}} F_y = P - mg$$

$$\implies P = m \left( \frac{d^2 y_p}{dt^2} + g \right)$$

However,  $x_p$  and  $y_p$  are exact functions of  $\theta$ . Therefore, we can represent their derivatives in terms of the derivatives of  $\theta$ .

$$x_p = x - L \sin(\theta)$$

$$\frac{dx_p}{dt} = \frac{dx}{dt} - L \cos \theta \frac{d\theta}{dt}$$

$$\frac{d^2 x_p}{dt^2} = \frac{d^2 x}{dt^2} + L \sin \theta \left( \frac{d\theta}{dt} \right)^2 - L \cos \theta \frac{d^2 \theta}{dt^2}$$

$$y_p = L \cos(\theta)$$

$$\frac{dy_p}{dt} = -L \sin \theta \frac{d\theta}{dt}$$

$$\frac{d^2 y_p}{dt^2} = -L \cos \theta \left( \frac{d\theta}{dt} \right)^2 - L \sin \theta \frac{d^2 \theta}{dt^2}$$

These expressions can then be substituted into the expressions for  $N$  and  $P$ . Rather than continuing with algebra here, we will simply represent these equations in Simulink.

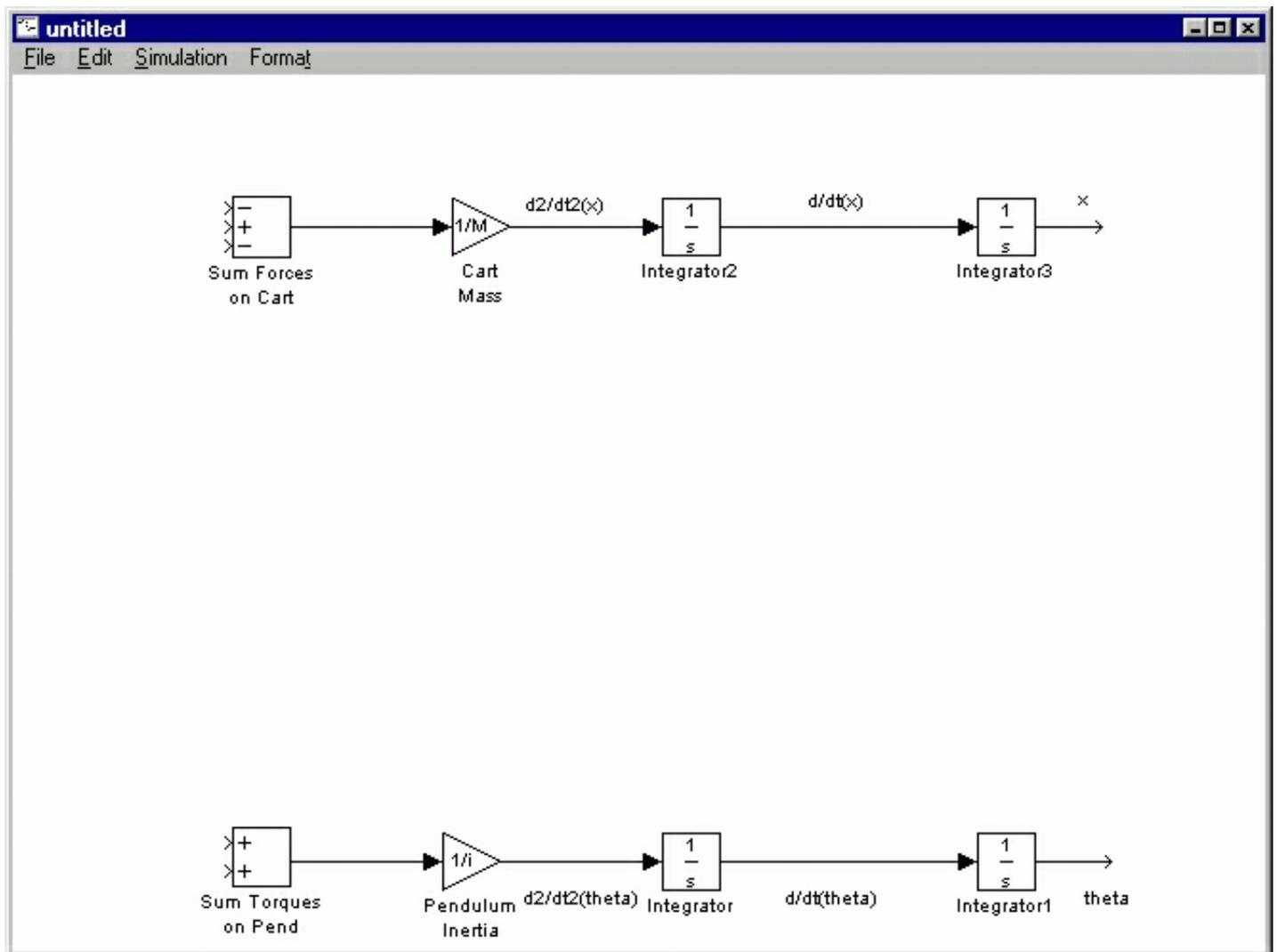
Simulink can work directly with nonlinear equations, so it is unnecessary to linearize these equations as it was in the [MATLAB tutorials](#).

## Building the Model in Simulink

First, we will model the states of the system in  $\theta$  and  $x$ . We will represent Newton's equations for the pendulum rotational inertia and the cart mass.

- Open a new model window in Simulink, and resize it to give plenty of room (this is a large model).
- Insert two integrators (from the Linear block library) near the bottom of your model and connect them in series.
- Draw a line from the second integrator and label it "theta". (To insert a label, double-click where you want the label to go.)
- Label the line connecting the integrators "d/dt(theta)".

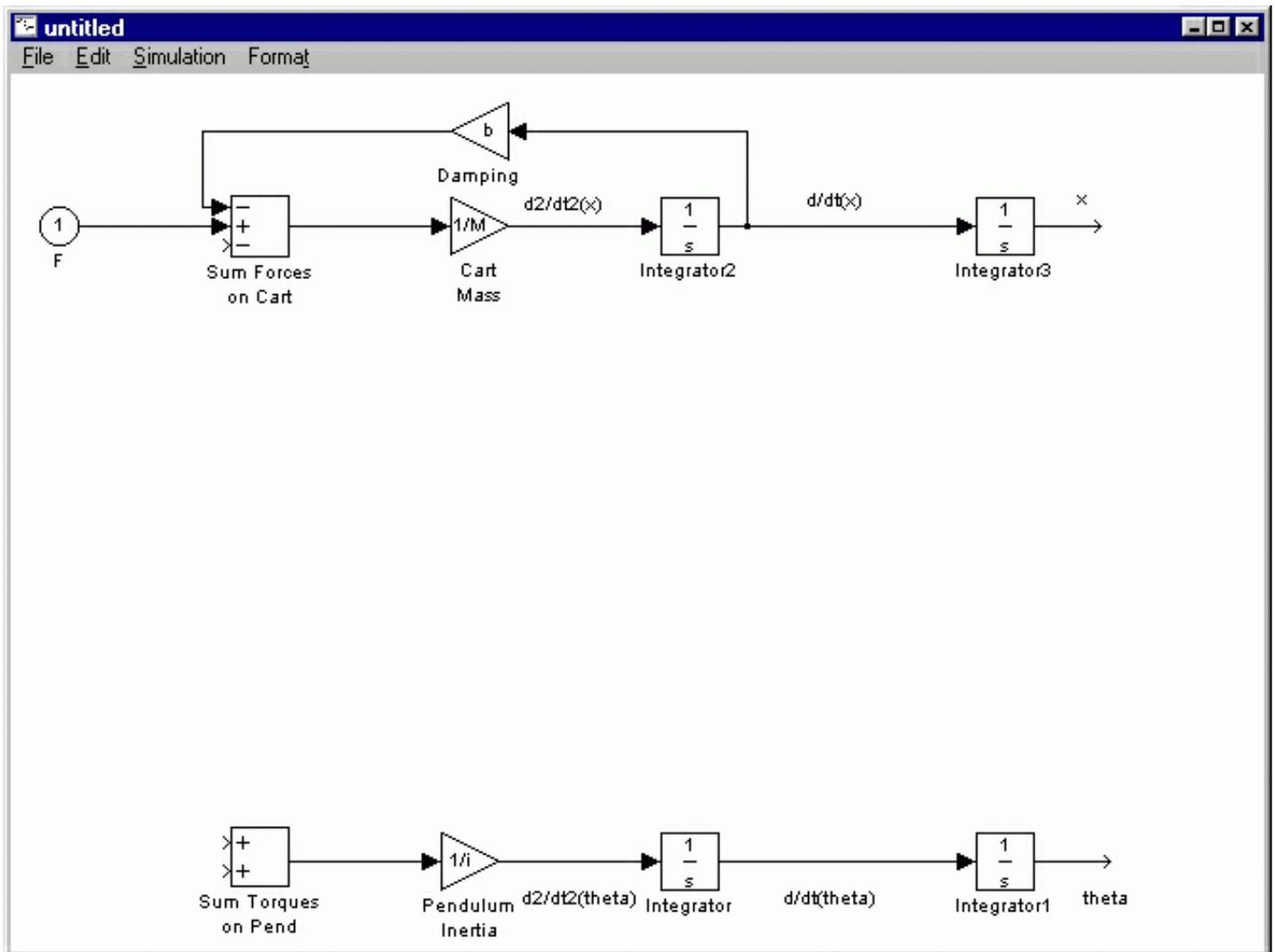
- Draw a line leading to the first integrator and label it " $d^2/dt^2(\theta)$ ".
- Insert a Gain block (from the Linear block library) to the left of the first integrator and connect its output to the  $d^2/dt^2(\theta)$  line.
- Edit the gain value of this block by double clicking it and change it to " $1/I$ ".
- Change the label of this block to "Pendulum Inertia" by clicking on the word "Gain". (you can insert a newline in the label by hitting return).
- Insert a Sum block (from the Linear block library) to the left of the Pendulum Inertia block and connect its output to the inertia's input.
- Change the label of this block to Sum Torques on Pend.
- Construct a similar set of elements near the top of your model with the signals labeled with " $x$ " rather than " $\theta$ ". The gain block should have the value " $1/M$ " with the label "Cart Mass", and the Sum block should have the label "Sum Forces on Cart".
- Edit the Sum Forces block and change its signs to "-+-". This represents the signs of the three horizontal forces acting on the cart.



Now, we will add in two of the forces acting on the cart.

- Insert a Gain block above the Cart Mass block. Change its value to " $b$ " and its label to "damping".
- Flip this block left-to-right by single clicking on it (to select it) and selecting Flip Block from the Format menu (or hit Ctrl-F).
- Tap a line off the  $d/dt(x)$  line (hold Ctrl while drawing the line) and connect it to the input of the damping block.

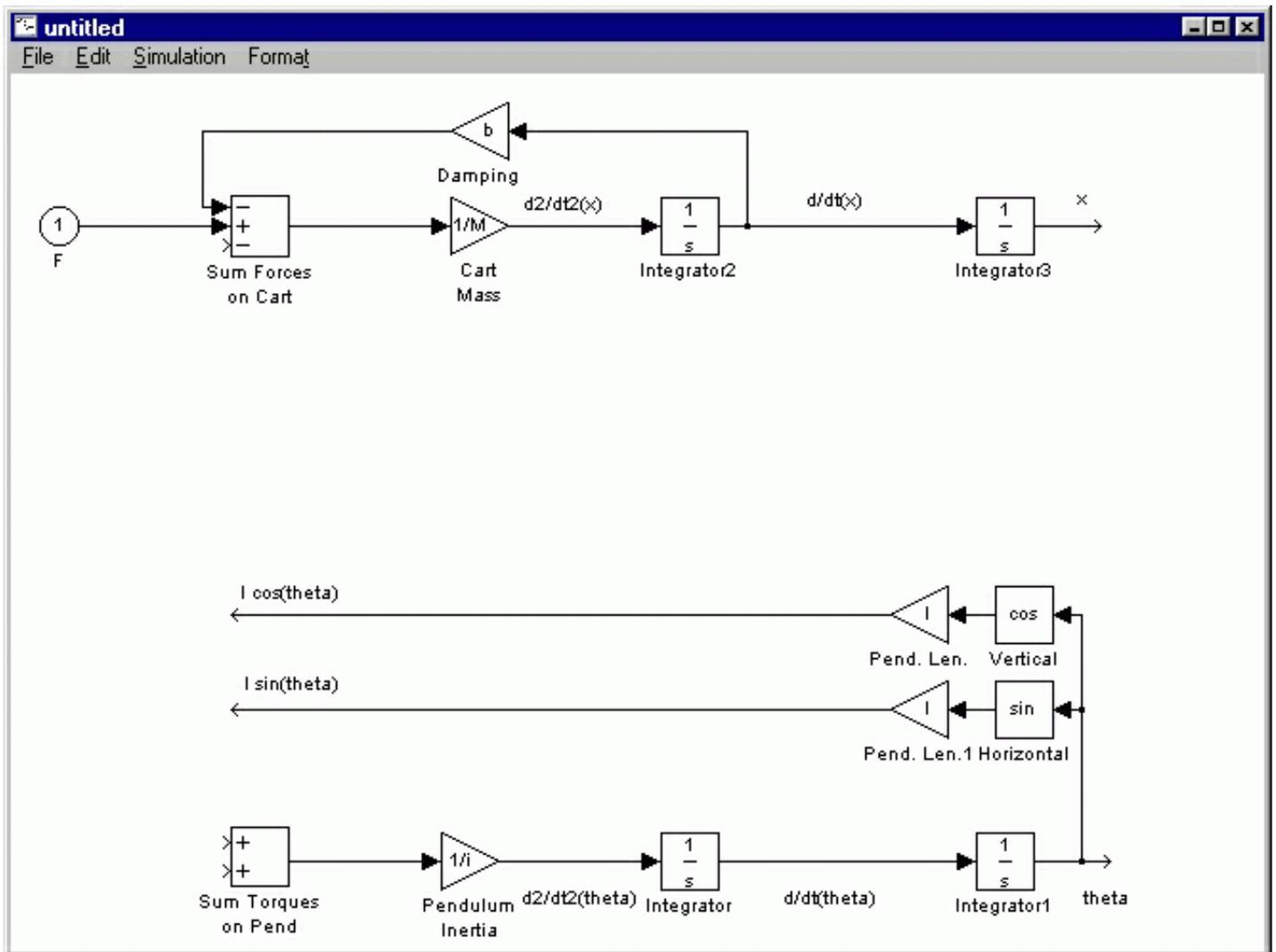
- Connect the output of the damping block to the topmost input of the Sum Forces block. The damping force then has a negative sign.
- Insert an In block (from the Connections block library) to the left of the Sum Forces block and change its label to "F".
- Connect the output of the F in block to the middle (positive) input of the Sum Forces block.



Now, we will apply the forces  $N$  and  $P$  to both the cart and the pendulum. These forces contribute torques to the pendulum with components " $N L \cos(\theta)$  and  $P L \sin(\theta)$ ". Therefore, we need to construct these components.

- Insert two Elementary Math blocks (from the Nonlinear block library) and place them one above the other above the second  $\theta$  integrator. These blocks can be used to generate simple functions such as  $\sin$  and  $\cos$ .
- Edit upper Math block's value to " $\cos$ " and leave the lower Math block's value " $\sin$ ".
- Label the upper ( $\cos$ ) block "Vertical" and the lower ( $\sin$ ) block "Horizontal" to identify the components.
- Flip each of these blocks left-to-right.
- Tap a line off the  $\theta$  line and connect it to the input of the  $\cos$  block.
- Tap a line of the line you just drew and connect it to the input of the  $\sin$  block.
- Insert a Gain block to the left of the  $\cos$  block and change its value to " $l$ " (lowercase  $L$ ) and its label to "Pend. Len."
- Flip this block left-to-right and connect it to the output of the  $\cos$  block.

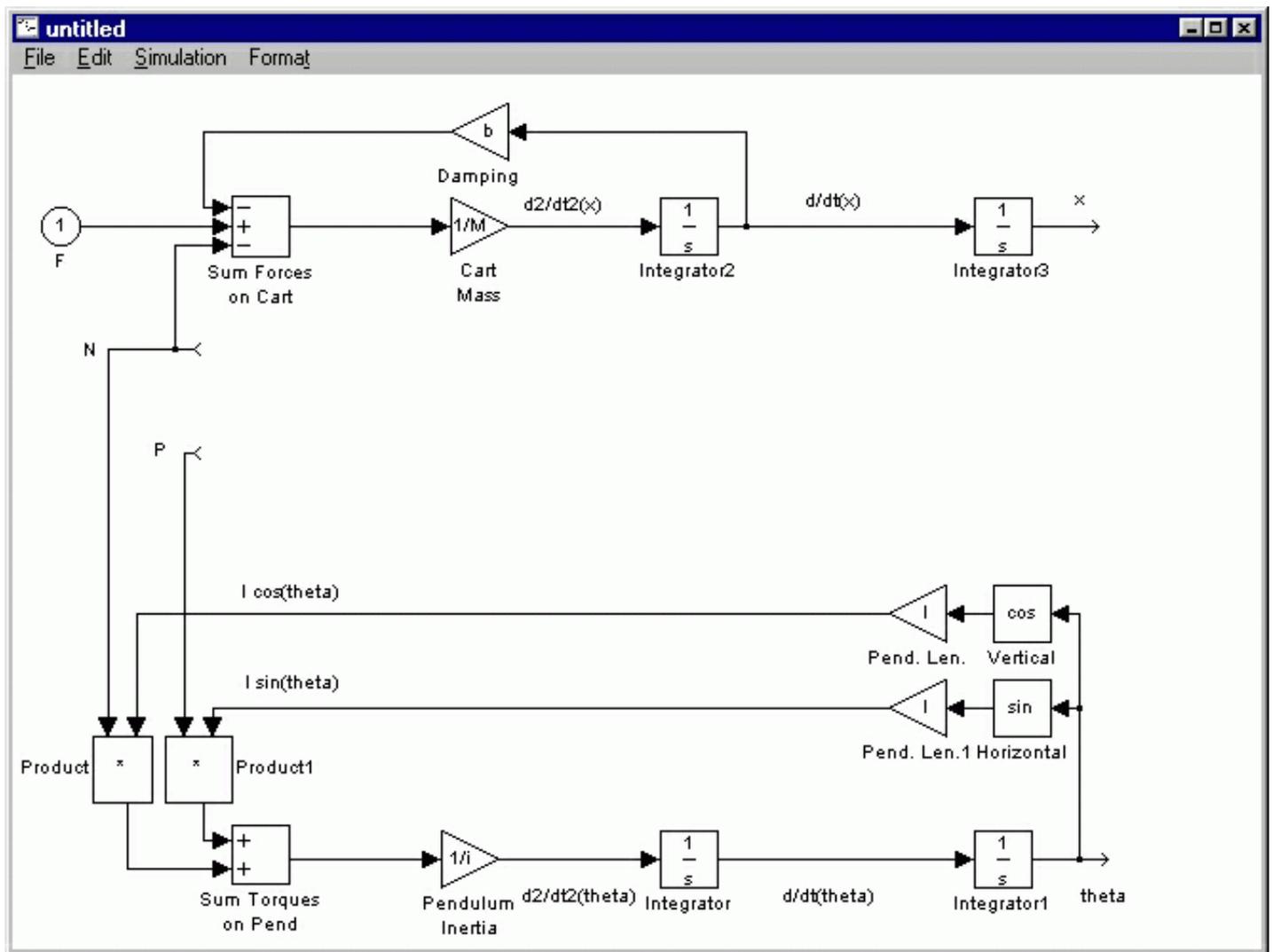
- Copy this block to a position to the left of the sin block. To do this, select it (by single-clicking) and select Copy from the Edit Menu and then Paste from the Edit menu (or hit Ctrl-C and Ctrl-V). Then, drag it to the proper position.
- Connect the new Pend. Len.1 block to the output of the sin block.
- Draw long horizontal lines leading from both these Pend. Len. blocks and label the upper one " $l \cos(\theta)$ " and the lower one " $l \sin(\theta)$ ".



Now that the pendulum components are available, we can apply the forces N and P. We will assume we can generate these forces, and just draw them coming from nowhere for now.

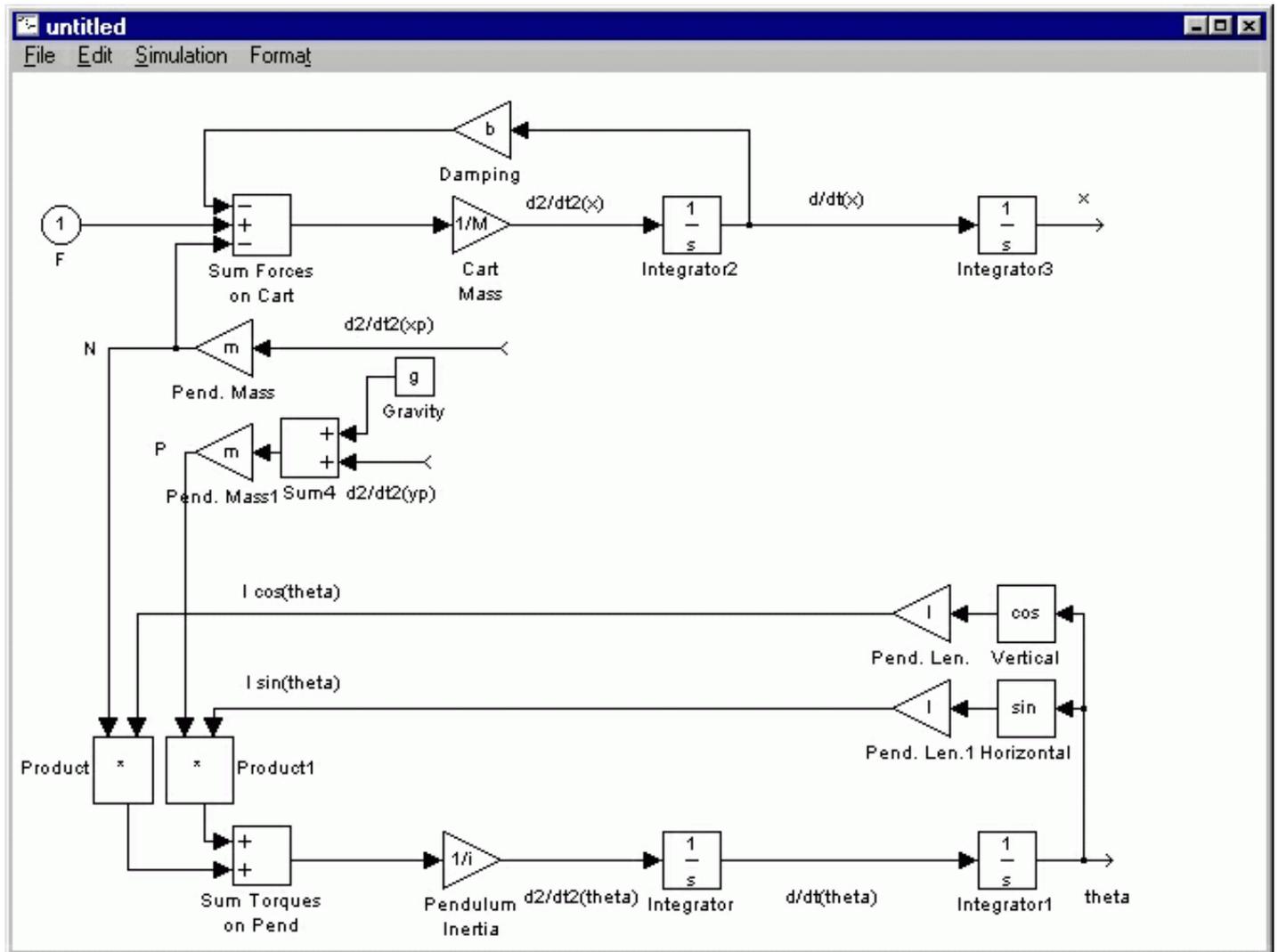
- Insert two Product blocks (from the Nonlinear block library) next to each other to the left and above the Sum Torques block. These will be used to multiply the forces N and P by their appropriate components.
- Rotate the left Product block 90 degrees. To do this, select it and select Rotate Block from the Format menu (or hit Ctrl-R).
- Flip the other product block left-to-right and also rotate it 90 degrees.
- Connect the left Product block's output to the lower input of the Sum Torques block.
- Connect the right Product block's output to the upper input of the Sum Torques block.
- Continue the  $l \cos(\theta)$  line and attach it to the right input of the left Product block.
- Continue the  $l \sin(\theta)$  line and attach it to the right input of the right Product block.
- Begin drawing a line from the open input of the right product block. Extend it up and the to the right. Label the open end of this line "P".

- Begin drawing a line from the open input of the left product block. Extend it up and then to the right. Label the open end of this line "N".
- Tap a line off the N line and connect it to the open input of the Sum forces block.



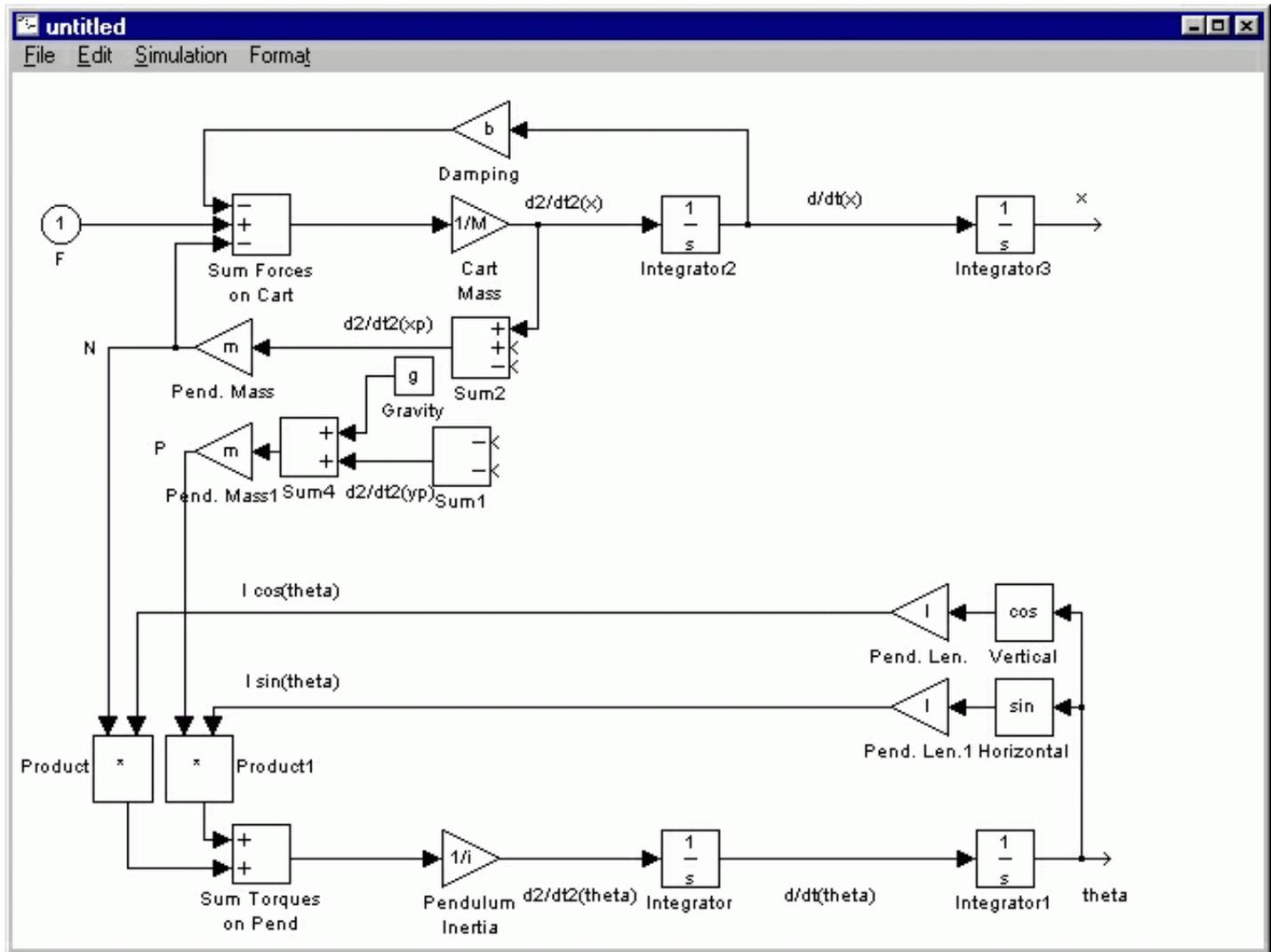
Next, we will represent the force  $N$  and  $P$  in terms of the pendulum's horizontal and vertical accelerations from Newton's laws.

- Insert a Gain block to the right of the  $N$  open ended line and change its value to " $m$ " and its label to "Pend. Mass".
- Flip this block left-to-right and connect it's to  $N$  line.
- Copy this block to a position to the right of the open ended  $P$  line and attach it to the  $P$  line.
- Draw a line leading to the upper Pend. Mass block and label it " $d^2/dt^2(x_p)$ ".
- Insert a Sum block to the right of the lower Pend. Mass block.
- Flip this block left-to-right and connect its output to the input of the lower Pend. Mass block.
- Insert a Constant block (from the Sources block library) to the right of the new Sum block, change its value to " $g$ " and label it "Gravity".
- Connect the Gravity block to the upper (positive) input of the newest Sum block.
- Draw a line leading to the open input of the new Sum block and label it " $d^2/dt^2(y_p)$ ".



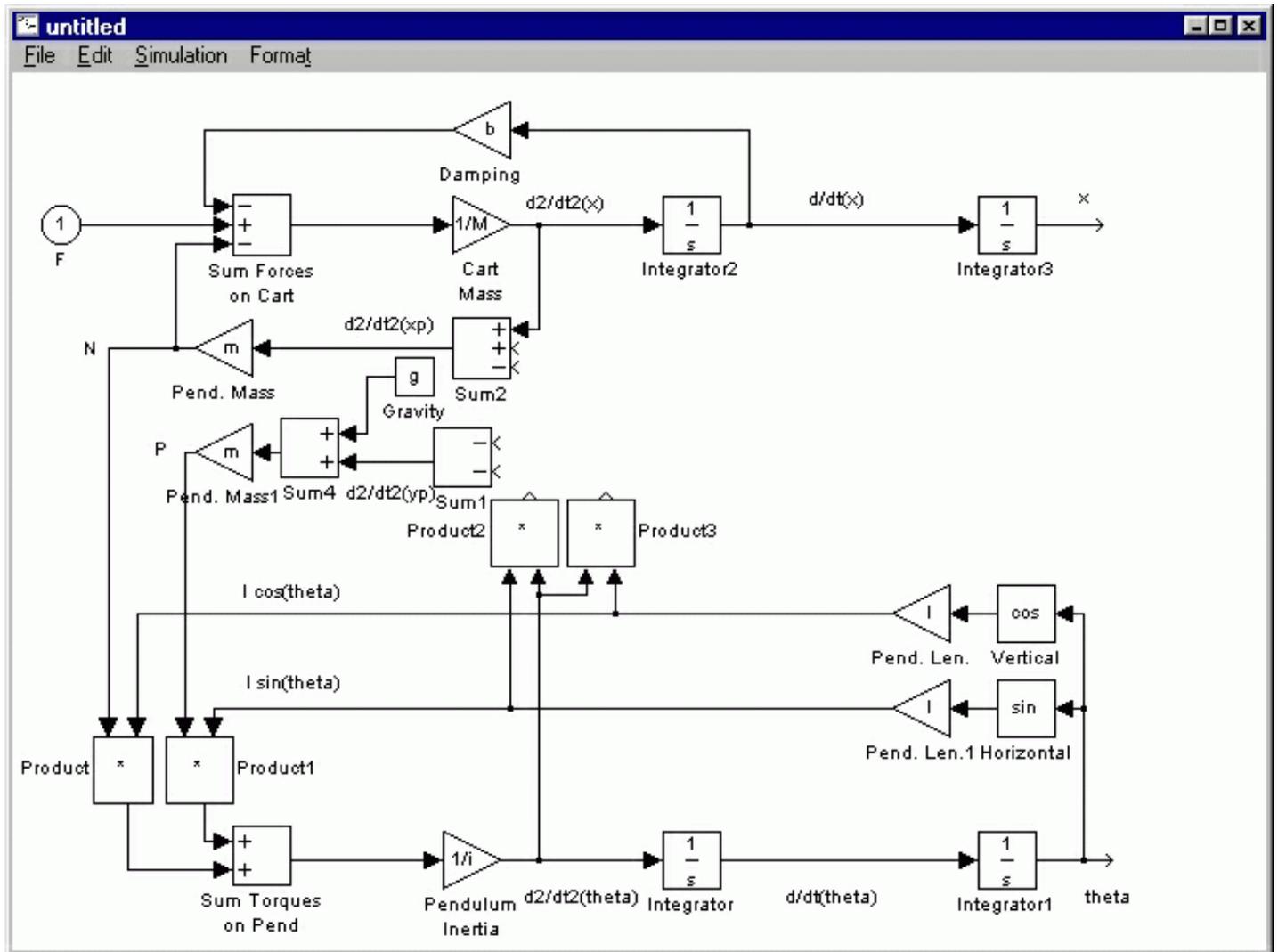
Now, we will begin to produce the signals which contribute to  $\ddot{x}_p$  and  $\ddot{y}_p$ .

- Insert a Sum block to the right of the  $\ddot{y}_p$  open end.
- Change the Sum block's signs to "--" to represent the two terms contributing to  $\ddot{y}_p$ .
- Flip the Sum block left-to-right and connect its output to the  $\ddot{y}_p$  signal.
- Insert a Sum block to the right of the  $\ddot{x}_p$  open end.
- Change the Sum block's signs to "++-" to represent the three terms contributing to  $\ddot{x}_p$ .
- Flip the Sum block left-to-right and connect its output to the  $\ddot{x}_p$  signal.
- The first term of  $\ddot{x}_p$  is  $\ddot{x}$ . Tap a line off the  $\dot{x}$  signal and connect it to the topmost (positive) input of the newest Sum block.



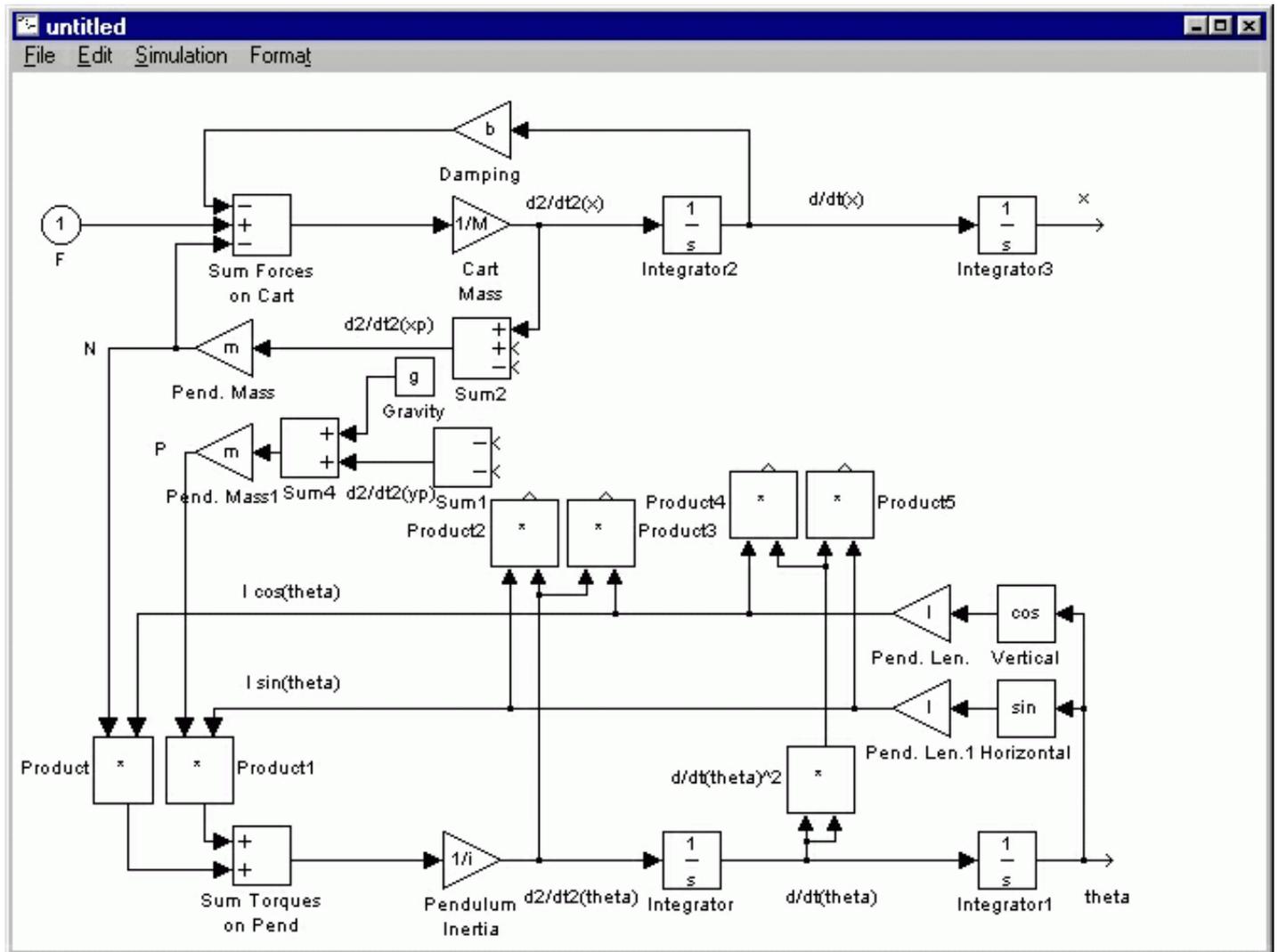
Now, we will generate the terms  $d^2/dt^2(\theta) * l \sin(\theta)$  and  $d^2/dt^2(\theta) * l \cos(\theta)$ .

- Insert two Product blocks next to each other to the right and below the Sum block associated with  $d^2/dt^2(\theta)$ .
- Rotate the left Product block 90 degrees.
- Flip the other product block left-to-right and also rotate it 90 degrees.
- Tap a line off the  $l \sin(\theta)$  signal and connect it to the left input of the left Product block.
- Tap a line off the  $l \cos(\theta)$  signal and connect it to the right input of the right Product block.
- Tap a line off the  $d^2/dt^2(\theta)$  signal and connect it to the right input of the left Product block.
- Tap a line of this new line and connect it to the left input of the right Product block.



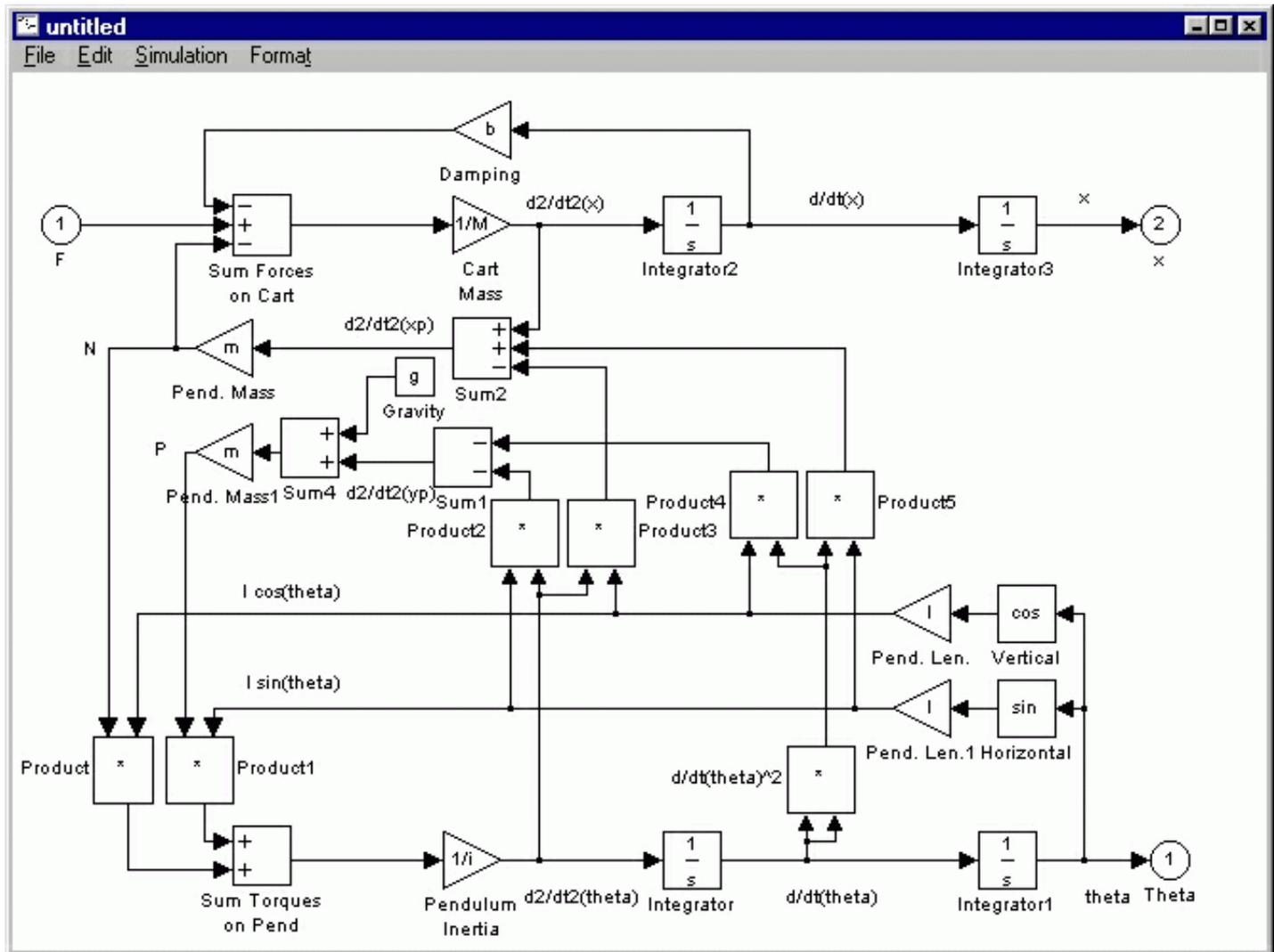
Now, we will generate the terms  $(d/dt(\theta))^2 \cdot l \sin(\theta)$  and  $(d/dt(\theta))^2 \cdot l \cos(\theta)$ .

- Insert two Product blocks next to each other to the right and slightly above the previous pair of Product blocks.
- Rotate the left Product block 90 degrees.
- Flip the other product block left-to-right and also rotate it 90 degrees.
- Tap a line off the  $l \cos(\theta)$  signal and connect it to the left input of the left Product block.
- Tap a line off the  $l \sin(\theta)$  signal and connect it to the right input of the right Product block.
- Insert a third Product block and insert it slightly above the  $d/dt(\theta)$  line. Label this block " $d/dt(\theta)^2$ ".
- Tap a line off the  $d/dt(\theta)$  signal and connect it to the left input of the lower Product block.
- Tap a line of this new line and connect it to the right input of the lower Product block.
- Connect the output of the lower Product block to the free input of the right upper Product block.
- Tap a line of this new line and connect it to the free input of the left upper Product block.



Finally, we will connect these signals to produce the pendulum acceleration signals. In addition, we will create the system outputs  $x$  and  $\theta$ .

- Connect the  $d^2/dt^2(\theta) \cdot l \sin(\theta)$  Product block's output to the lower (negative) input of the  $d^2/dt^2(y_p)$  Sum block.
- Connect the  $d^2/dt^2(\theta) \cdot l \cos(\theta)$  Product block's output to the lower (negative) input of the  $d^2/dt^2(x_p)$  Sum block.
- Connect the  $d/dt(\theta)^2 \cdot l \cos(\theta)$  Product block's output to the upper (negative) input of the  $d^2/dt^2(y_p)$  Sum block.
- Connect the  $d/dt(\theta)^2 \cdot l \sin(\theta)$  Product block's output to the middle (positive) input of the  $d^2/dt^2(x_p)$  Sum block.
- Insert an Out block (from the Connections block library) attached to the  $\theta$  signal. Label this block "Theta".
- Insert an Out block attached to the  $x$  signal. Label this block "x". It should automatically be numbered 2.

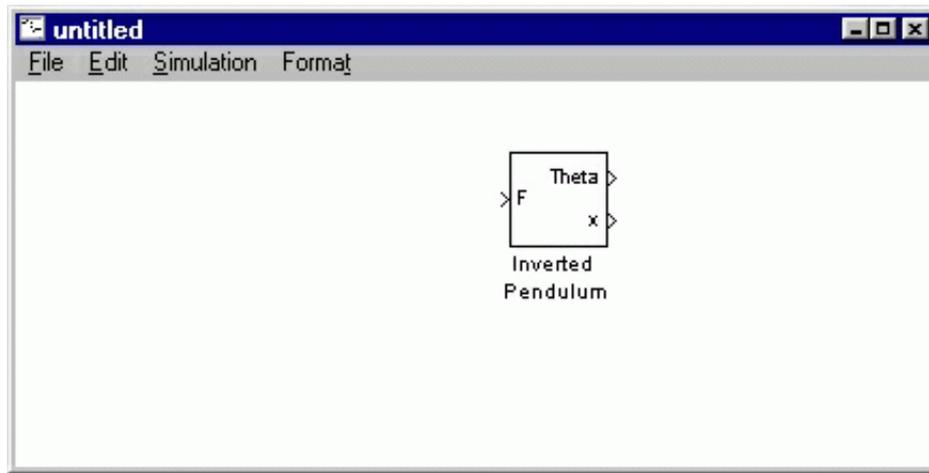


Now, save this model as [pend.mdl](#).

## Open-loop response

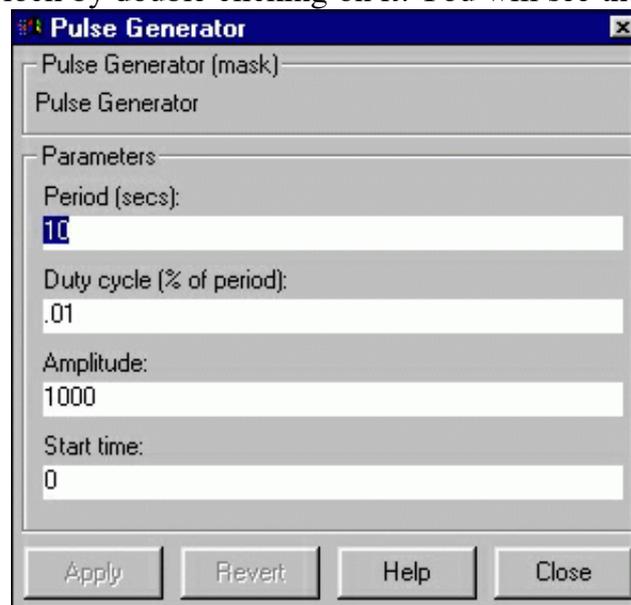
To generate the open-loop response, it is necessary to contain this model in a subsystem block.

- Create a new model window (select New from the File menu in Simulink or hit Ctrl-N).
- Insert a Subsystem block from the Connections block library.
- Open the Subsystem block by double clicking on it. You will see a new model window labeled "Subsystem".
- Open your previous model window named [pend.mdl](#). Select all of the model components by selecting Select All from the Edit menu (or hit Ctrl-A).
- Copy the model into the paste buffer by selecting Copy from the Edit menu (or hit Ctrl-C).
- Paste the model into the Subsystem window by selecting Paste from the Edit menu (or hit Ctrl-V) in the Subsystem window
- Close the Subsystem window. You will see the Subsystem block in the untitled window with one input terminal labeled F and two output terminals labeled Theta and x.
- Resize the Subsystem block to make the labels visible by selecting it and dragging one of the corners.
- Label the Subsystem block "Inverted Pendulum".

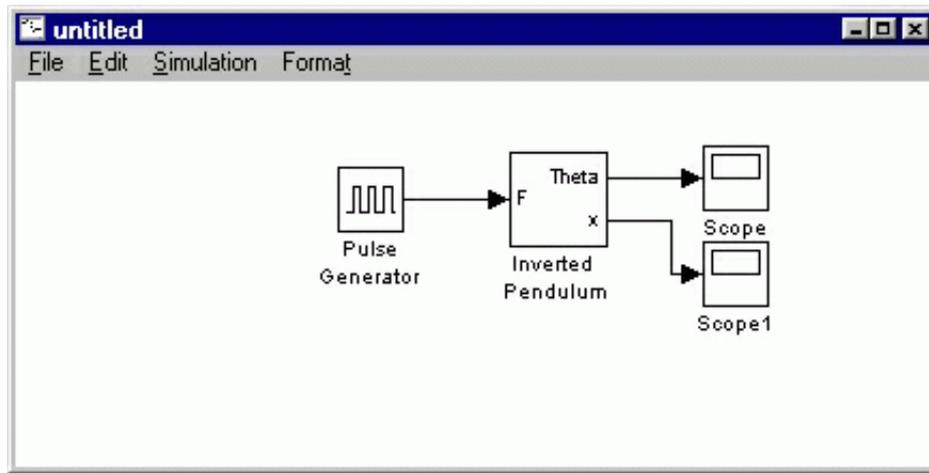


Now, we will apply a unit impulse force input, and view the pendulum angle and cart position. An impulse can not be exactly simulated, since it is an infinite signal for an infinitesimal time with time integral equal to 1. Instead, we will use a pulse generator to generate a large but finite pulse for a small but finite time. The magnitude of the pulse times the length of the pulse will equal 1.

- Insert a Pulse Generator block from the Sources block library and connect it to the F input of the Inverted Pendulum block.
- Insert a Scope block (from the Sinks block library) and connect it to the Theta output of the Inverted Pendulum block.
- Insert a Scope block and connect it to the x output of the Inverted Pendulum block.
- Edit the Pulse Generator block by double clicking on it. You will see the following dialog box.



- Change the Period value to "10" (a long time between a chain of impulses - we will be interested in only the first pulse).
- Change the Duty Cycle value to ".01" this corresponds to .01% of 10 seconds, or .001 seconds.
- Change the Amplitude to 1000. 1000 times .001 equals 1, providing an approximate unit impulse.
- Close this dialog box. Your system will appear as shown below.



We now need to set an appropriate simulation time to view the response.

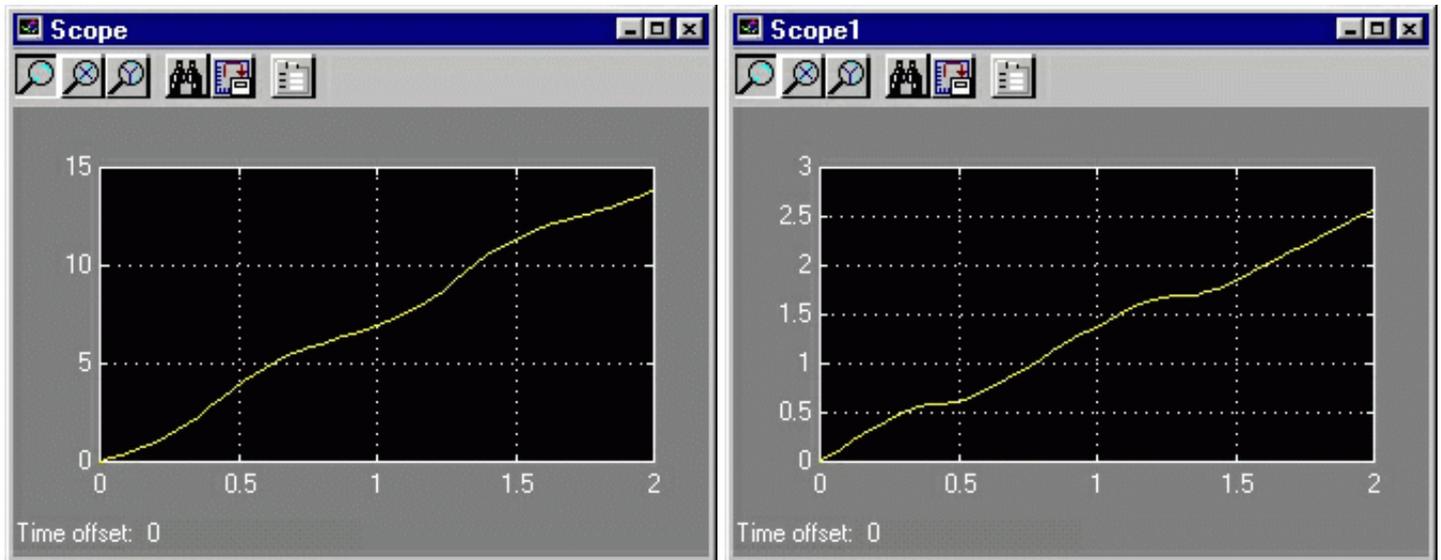
- Select Parameters from the Simulation menu.
- Change the Stop Time value to 2 seconds.
- Close this dialog box

You can download a version of the system [here](#). Before running it, it is necessary to set the physical constants. Enter the following commands at the MATLAB prompt.

```
M = .5;
m = 0.2;
b = 0.1;
i = 0.006;
g = 9.8;
l = 0.3;
```

Now, start the simulation (select Start from the Simulation menu or hit Ctrl-t). If you look at the MATLAB prompt, you will see some error messages concerning algebraic loops. Due to the algebraic constraint in this system, there are closed loops in the model with no dynamics which must be resolved completely at each time step before dynamics are considered. In general, this is not a problem, but often algebraic loops slow down the simulation, and can cause real problems if discontinuities exist within the loop (such as saturation, sign functions, etc.)

Open both Scopes and hit the autoscale buttons. You will see the following for theta (left) and x (right).



Notice that the pendulum swings all the way around due to the impact, and the cart travels along with a jerky motion due to the pendulum. These simulations differ greatly from the MATLAB [open loop simulations](#) because Simulink allows for fully nonlinear systems.

## Extracting the linearized model into MATLAB

Since MATLAB can't deal with nonlinear systems directly, we cannot extract the exact model from Simulink into MATLAB. However, a *linearized* model can be extracted. This is done through the use of In and Out Connection blocks and the MATLAB function `linmod`. In the case of this example, will use the equivalent command `linmod2`, which can better handle the numerical difficulties of this problem.

To extract a model, it is necessary to start with a model file with inputs and outputs defined as In and Out blocks. Earlier in this tutorial this was done, and the file was saved as [pend.mdl](#). In this model, one input,  $F$  (the force on the cart) and two outputs,  $\theta$  (pendulum angle from vertical) and  $x$  (position of the cart), were defined. When linearizing a model, it is necessary to choose an operating point about which to linearize. By default, the `linmod2` command linearizes about a state vector of zero and zero input. Since this is the point about which we would like to linearize, we do not need to specify any extra arguments in the command. Since the system has two outputs, we will generate two transfer functions.

At the MATLAB prompt, enter the following commands

```
[A,B,C,D]=linmod2('pend')
[nums,den]=ss2tf(A,B,C,D)
numtheta=nums(1,:)
numx=nums(2,:)
```

You will see the following output (along with algebraic loop error messages) providing a state-space model, two transfer function numerators, and one transfer function denominator (both transfer functions share the same denominator).

```
A =
      0      0      1.0000      0
      0      0          0      1.0000
  31.1818  0.0000  0.0000 -0.4545
   2.6727  0.0000  0.0000 -0.1818
```

B =

```

      0
      0
  4.5455
  1.8182

```

C =

```

  1   0   0   0
  0   1   0   0

```

D =

```

  0
  0

```

nums =

```

      0   0.0000   4.5455   0.0000   0.0000
      0   0.0000   1.8182   0.0000  -44.5455

```

den =

```

  1.0000   0.1818  -31.1818  -4.4545   0.0000

```

numtheta =

```

      0   0.0000   4.5455   0.0000   0.0000

```

numx =

```

      0   0.0000   1.8182   0.0000  -44.5455

```

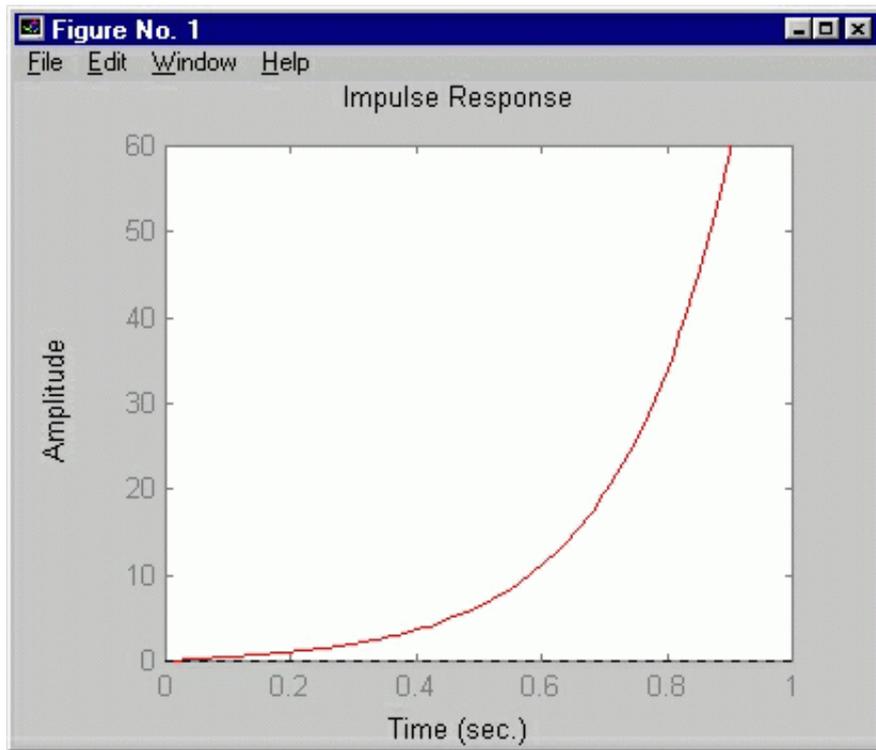
To verify the model, we will generate an open-loop response. At the MATLAB command line, enter the following commands.

```

t=0:0.01:5;
impulse(numtheta,den,t);
axis([0 1 0 60]);

```

You should get the following response for the angle of the pendulum.

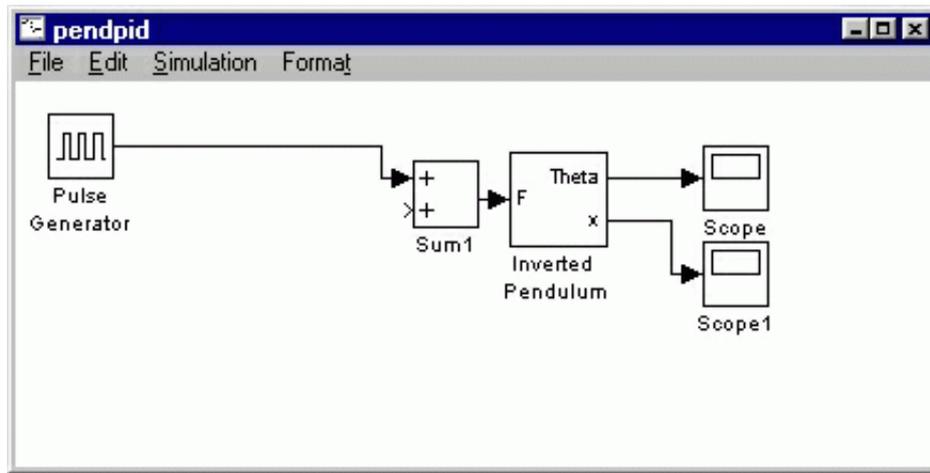


Note that this is identical to the impulse response obtained in the MATLAB tutorial [pendulum modeling](#) example. Since it is a linearized model, however, it is not the same as the fully-nonlinear impulse response obtained in Simulink.

## Implementing PID control

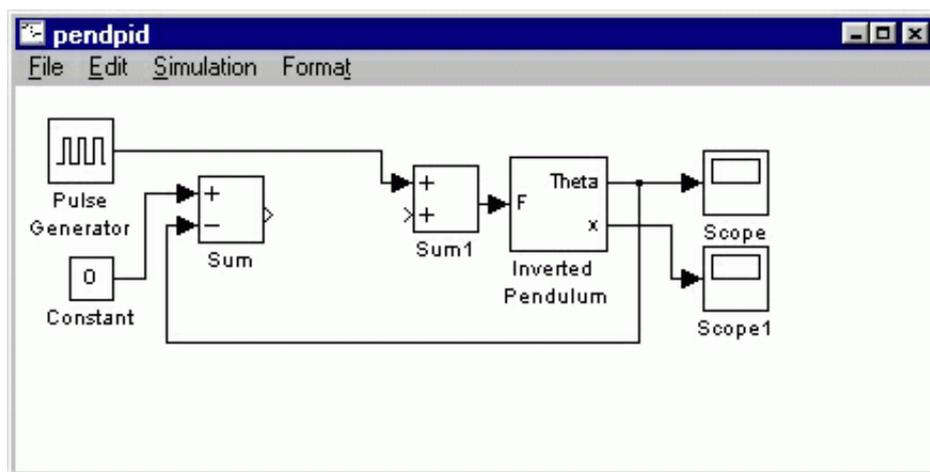
In the [pendulum PID control](#) example, a PID controller was designed with proportional, integral, and derivative gains equal to 100, 1, and 20, respectively. To implement this, we will start with our open-loop model of the inverted pendulum. And add in both a control input and the disturbance impulse input to the plant.

- Open your Simulink model window you used to obtain the nonlinear open-loop response. ([pendol.mdl](#))
- Delete the line connecting the Pulse Generator block to the Inverted Pendulum block. (single-click on the line and select Cut from the Edit menu or hit Ctrl-X).
- Move the Pulse Generator block to the upper left of the window.
- Insert a Sum block to the left of the Inverted Pendulum block.
- Connect the output of the Sum block to the Inverted Pendulum block.
- Connect the Pulse generator to the upper (positive) input of the Sum block.



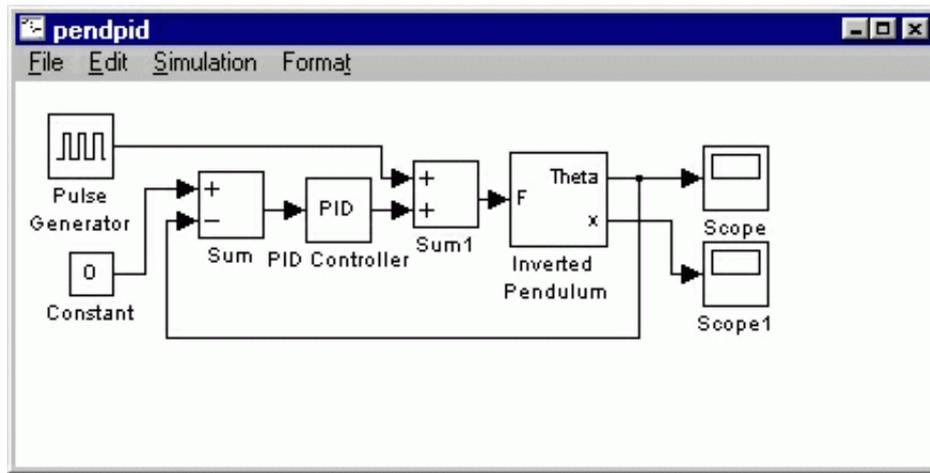
Now, we will feed back the angle output.

- Insert a Sum block to the right and below the Pulse Generator block.
- Change the signs of the Sum block to "+-".
- Insert a Constant block (from the Sources block library) below the Pulse generator. Change its value to "0". This is the reference input.
- Connect the constant block to the upper (positive) input of the second Sum block.
- Tap a line off the Theta output of the Inverted Pendulum block and draw it down and to the left. Extend this line and connect it to the lower (negative) input of the second Sum block.

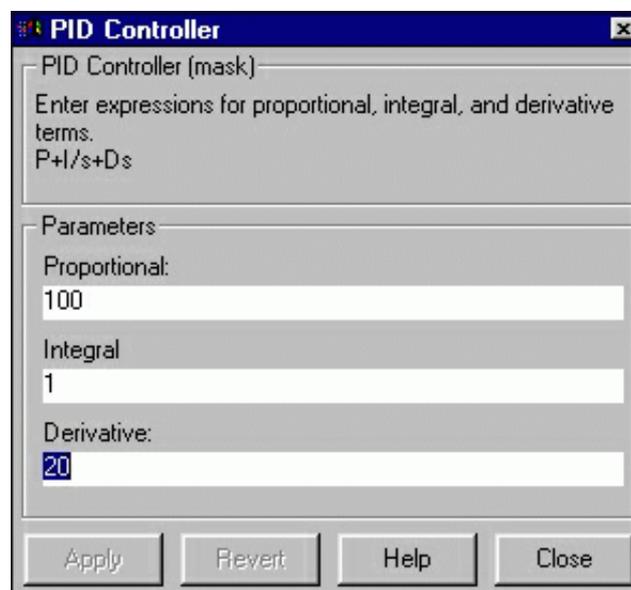


Now, we will insert a PID controller.

- Double-click on the Blocksets & Toolboxes icon in the main Simulink window. This will open a new window with two icons.
- In this new window, double-click on the SIMULINK extras icon. This will open a window with icons similar to the main Simulink window.
- Double-click on the Additional Linear block library icon. This will bring up a library of Linear blocks to augment the standard Linear block library.
- Drag a PID Controller block into your model between the two Sum blocks.
- Connect the output of the second Sum block to the input of the PID block.
- Connect the PID output to the first Sum block's free input.



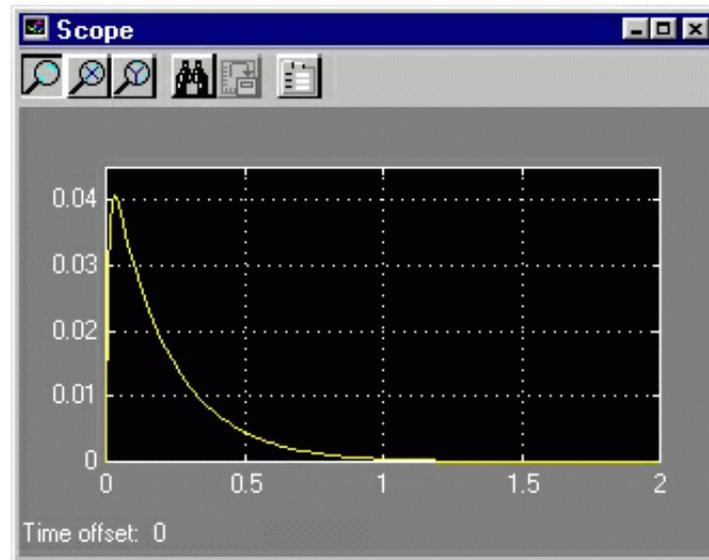
- Edit the PID block by doubleclicking on it.
- Change the Proportional gain to 100, leave the Integral gain 1, and change the Derivative gain to 20.
- Close this window.



You can download our version of the closed-loop system [here](#)

## Closed-loop response

We can now simulate the closed-loop system. Be sure the physical parameters are set (if you just ran the open-loop response, they should still be set.) Start the simulation, double-click on the Theta scope and hit the autoscale button. You should see the following response:



This is identical to the closed-loop response obtained in the MATLAB tutorials. Note that the PID controller handles the nonlinear system very well because the angle is very small (.04 radians).

## Simulink Examples

[Cruise Control](#) | [Motor Speed](#) | [Motor Position](#) | [Bus Suspension](#) | [Inverted Pendulum](#) | [Pitch Controller](#) | [Ball and Beam](#)

## Inverted Pendulum Examples

[Modeling](#) | [PID](#) | [Root Locus](#) | [Frequency Response](#) | [State Space](#) | [Digital Control](#) | [Simulink](#)

## Tutorials

[MATLAB Basics](#) | [MATLAB Modeling](#) | [PID](#) | [Root Locus](#) | [Frequency Response](#) | [State Space](#) | [Digital Control](#) | [Simulink Basics](#) | [Simulink Modeling](#) | [Examples](#)

[Home Page](#)

[MATLAB®  
Commands](#)

[Simulink®  
Blocks](#)

[Tutorial  
Index](#)