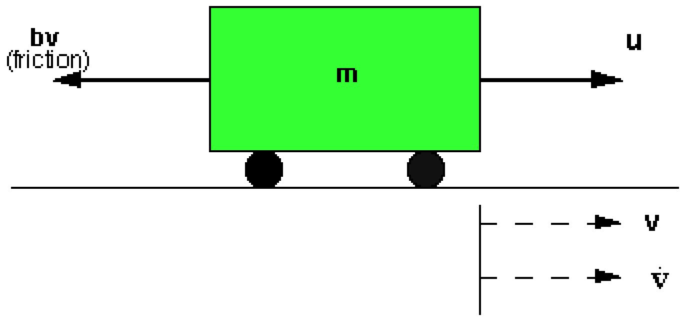
Example: Modeling a Cruise Control System in Simulink

Physical setup and system equations

The model of the cruise control system is relatively simple. If the inertia of the wheels is neglected, and it is assumed that friction (which is proportional to the car's speed) is what is opposing the motion of the car, then the problem is reduced to the simple mass and damper system shown below.



Using Newton's law, modeling equations for this system becomes:

$$m\frac{dv}{dt} = u - bv \quad (1)$$

where u is the force from the engine. For this example, let's assume that

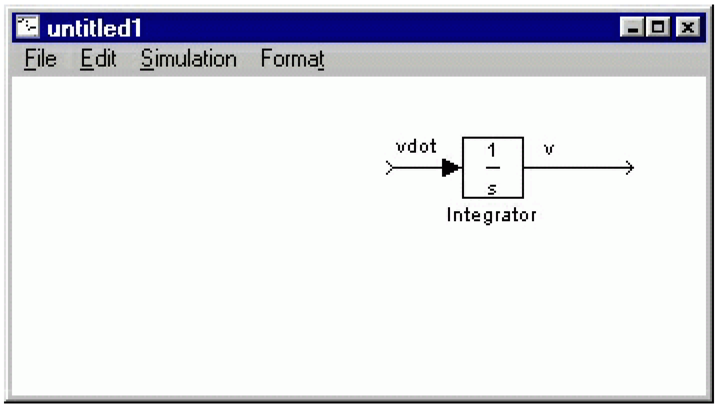
m = 1000kg
b = 50Nsec/m
u = 500N

Building the Model

This system will be modeled by summing the forces acting on the mass and integrating the acceleration to give the velocity. Open Simulink and open a new model window. First, we will model the integral of acceleration

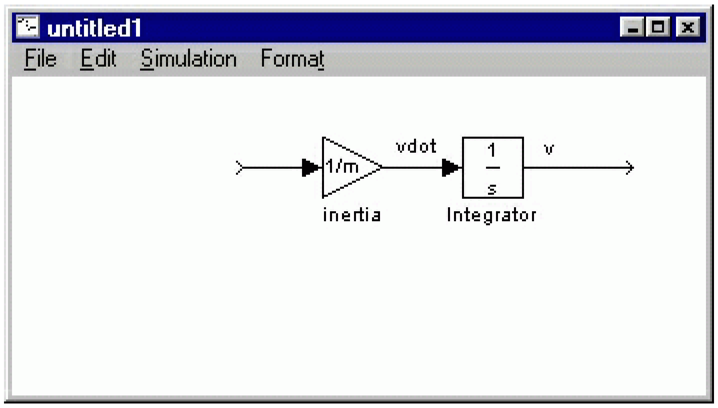
$$\int \frac{dv}{dt} = v$$

- Insert an Integrator Block (from the Linear block library) and draw lines to and from its input and output terminals.
- Label the input line "vdot" and the output line "v" as shown below. To add such a label, double click in the empty space just above the line.



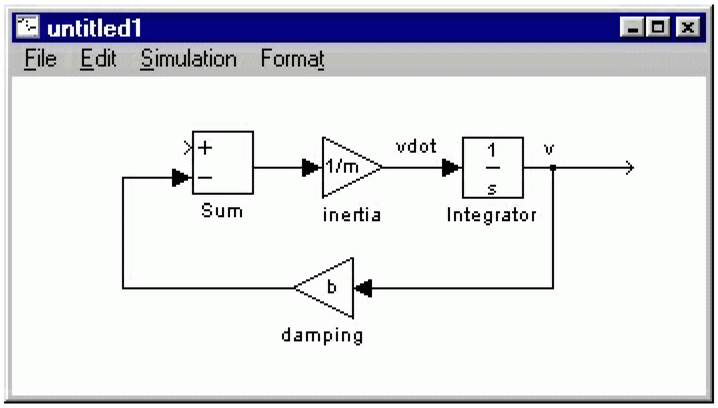
Since the acceleration (dv/dt) is equal to the sum of the forces divided by mass, we will divide the incoming signal by the mass.

- Insert a Gain block (from the Linear block library) connected to the integrators input line and draw a line leading to the input of the gain.
- Edit the gain block by double-clicking on it and change its value to "1/m".
- Change the label of the Gain block to "inertia" by clicking on the word "Gain" underneath the block.



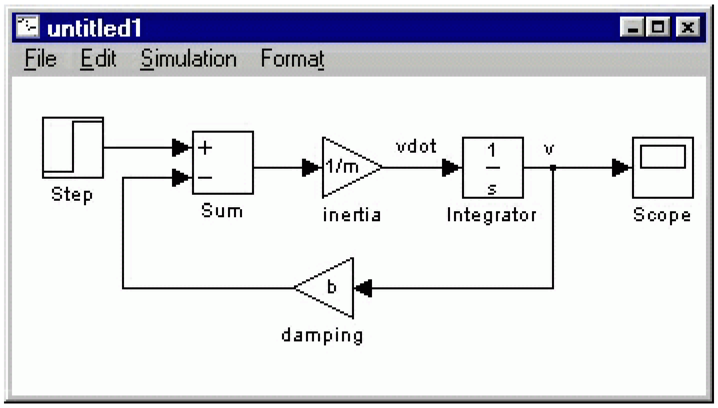
Now, we will add in the forces which are represented in Equation (1). First, we will add in the damping force.

- Attach a Sum block (from the Linear block library) to the line leading to the inertia gain.
- Change the signs of this block to "+-".
- Insert a gain block below the inertia block, select it by single-clicking on it, and select Flip from the Format menu (or type Ctrl-F) to flip it left-to-right.
- Set the gain value to "b" and rename this block to "damping".
- Tap a line (hold Ctrl while drawing) off the integrator's output and connect it to the input of the damping gain block.
- Draw a line from the damping gain output to the negative input of the Sum Block.

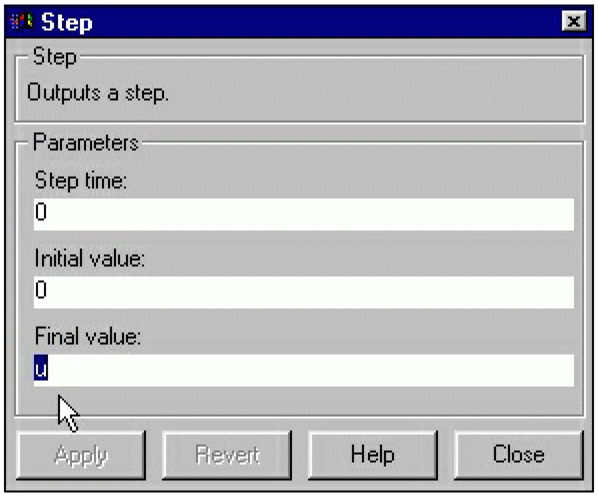


The second force acting on the mass is the control input, u. We will apply a step input.

- Insert a Step block (from the Sources block library) and connect it with a line to the positive input of the Sum Block.
- To view the output velocity, insert a Scope (from the Sinks block library) connected to the output of the integrator.



- To provide a appropriate step input of 500 at t=0, double-click the Step block and set the Step Time to "0" and the Final Value to "u".

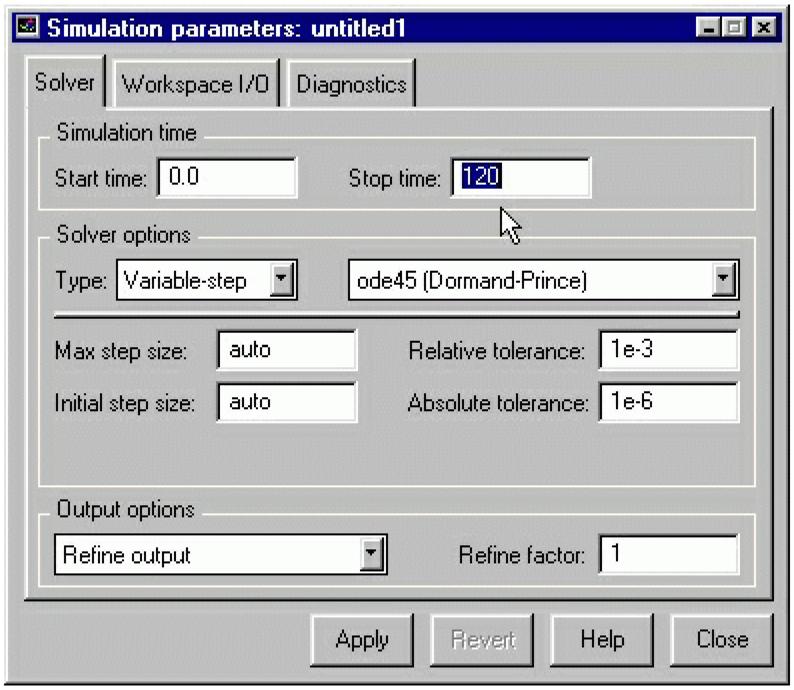


Step time: 0
Initial value: 0
Final value: u

Open-loop response

To simulate this system, first, an appropriate simulation time must be set.

- Select Parameters from the Simulation menu and enter "120" in the Stop Time field. 120 seconds is long enough to view the open-loop response.

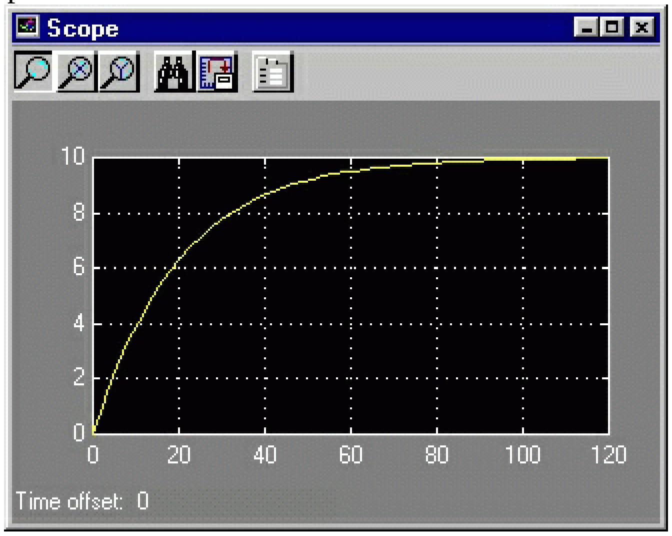


Start time: 0.0
Stop time: 120
Type: Variable-step
ode45 (Dormand-Prince)
Max step size: auto
Relative tolerance: 1e-3
Initial step size: auto
Absolute tolerance: 1e-6
Refine output
Refine factor: 1

The physical parameters must now be set. Run the following commands at the MATLAB prompt:

```
m=1000;
b=50;
u=500;
```

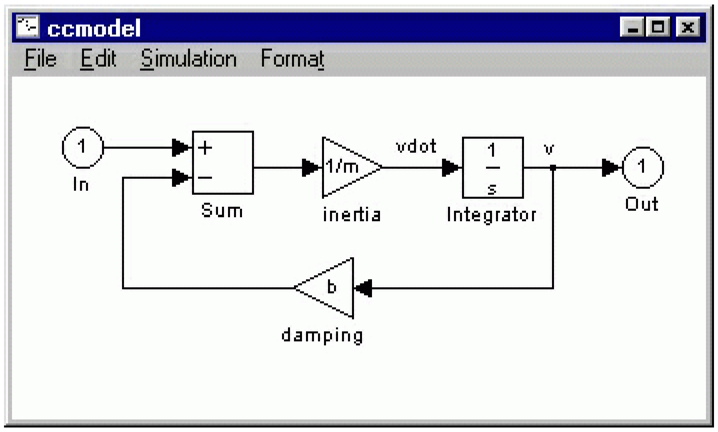
Run the simulation (Ctrl-t or Start on the Simulation menu). When the simulation is finished, double-click on the scope and hit its autoscale button. You should see the following output.



Extracting a Linear Model into MATLAB

A linear model of the system (in state space or transfer function form) can be extracted from a Simulink model into MATLAB. This is done through the use of In and Out Connection blocks and the MATLAB function linmod.

- Replace the Step Block and Scope Block with an In Connection Block and an Out Connection Block, respectively (these blocks can be found in the Connections block library). This defines the input and output of the system for the extraction process.



Save your file as "ccmodel.mdl" (select Save As from the File menu). MATLAB will extract the linear model from the saved model file, not from the open model window. At the MATLAB prompt, enter the following commands:

```
[A,B,C,D]=linmod('ccmodel')
[num,den]=ss2tf(A,B,C,D)
```

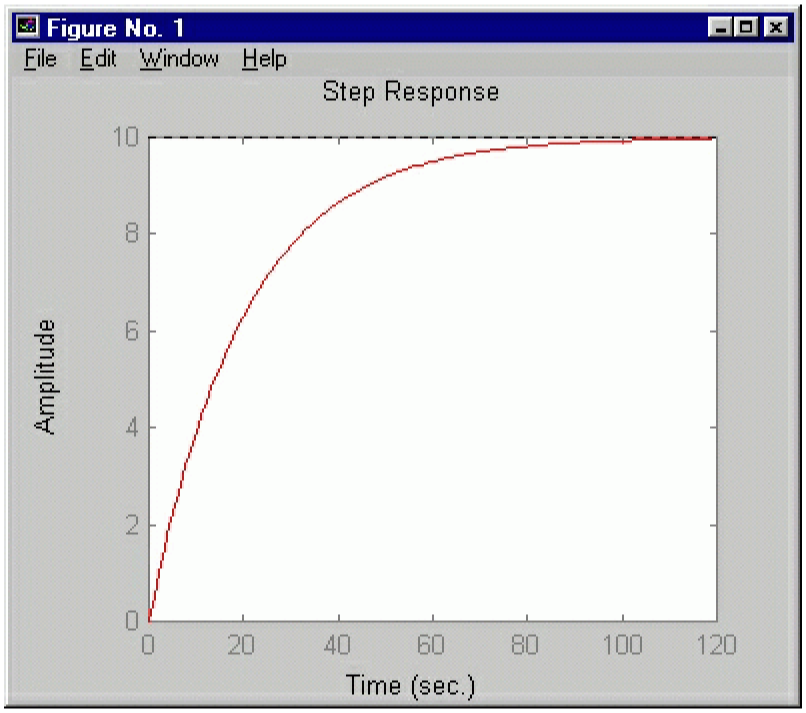
You should see the following output, providing both state-space and transfer function models of the system.

```
      A =

         -0.0500


      B =

        1.0000e-003


      C =

           1


      D =

           0


      num =

               0    0.0010


      den =

          1.0000    0.0500
```

To verify the model extraction, we will generate an open-loop step response of the extracted transfer function in MATLAB. We will multiply the numerator by 500 to simulate a step input of 500N. Enter the following command in MATLAB.

```
      step(500*num,den);
```

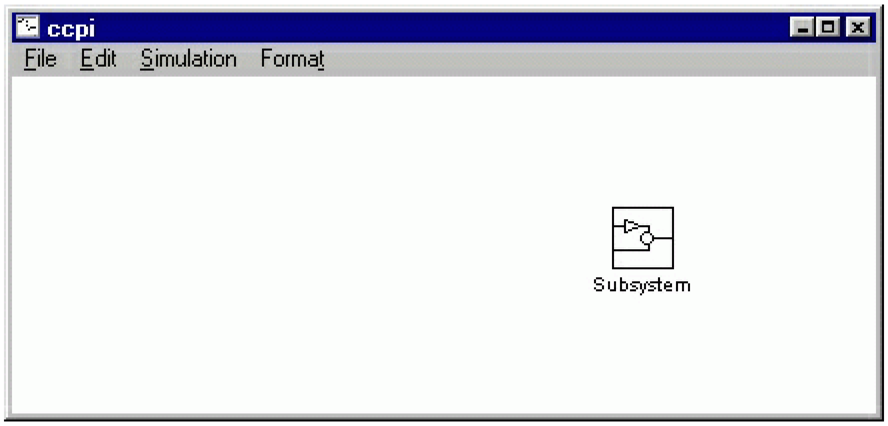
You should see the following plot which is equivalent to the Scope's output.



Implementing PI Control

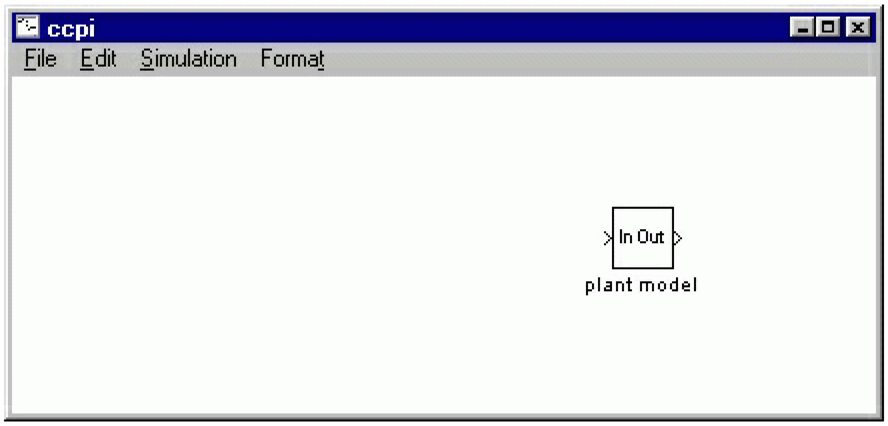
In the cruise control example a PI controller was designed with Kp=800 and Ki=40 to give the desired response. We will implement this in Simulink by first containing the open-loop system from earlier in this page in a Subsystem block.

- Create a new model window.
- Drag a Subsystem block from the Connections block library into your new model window.



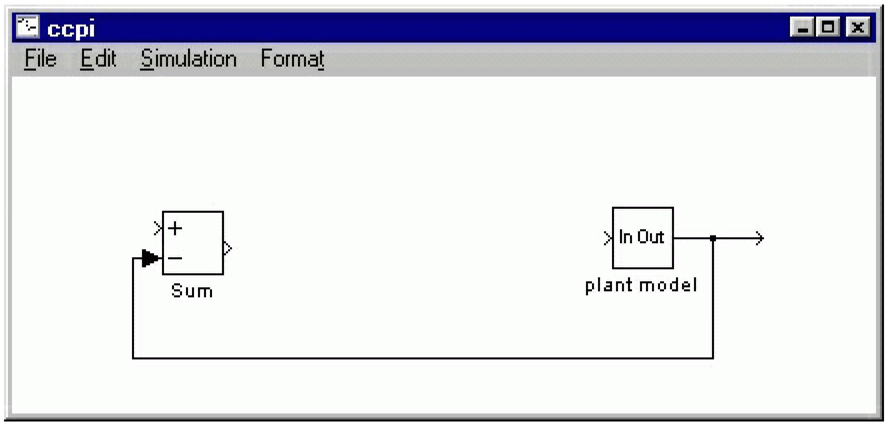
- Double click on this block. You will see a blank window representing the contents of the subsystem (which is currently empty).
- Open your previously saved model of the Cruise Control system, ccmodel.mdl.

- Select Select All from the Edit menu (or Ctrl-A), and select Copy from the Edit menu (or Ctrl-C).
- Select the blank subsystem window from your new model and select Paste from the Edit menu (or Ctrl-V). You should see your original system in this new subsystem window. Close this window.
- You should now see input and output terminals on the Subsystem block. Name this block "plant model".



Now, we will build a PI controller around the plant model. First, we will feed back the plant output.

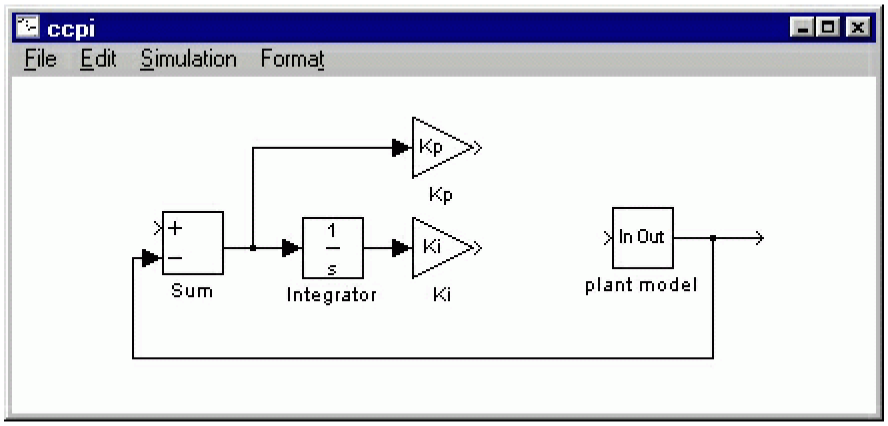
- Draw a line extending from the plant output.
- Insert a Sum block and assign "+-" to it's inputs.
- Tap a line of the output line and draw it to the negative input of the Sum block.



The output of the Sum block will provide the error signal. From this, we will generate proportional and integral components.

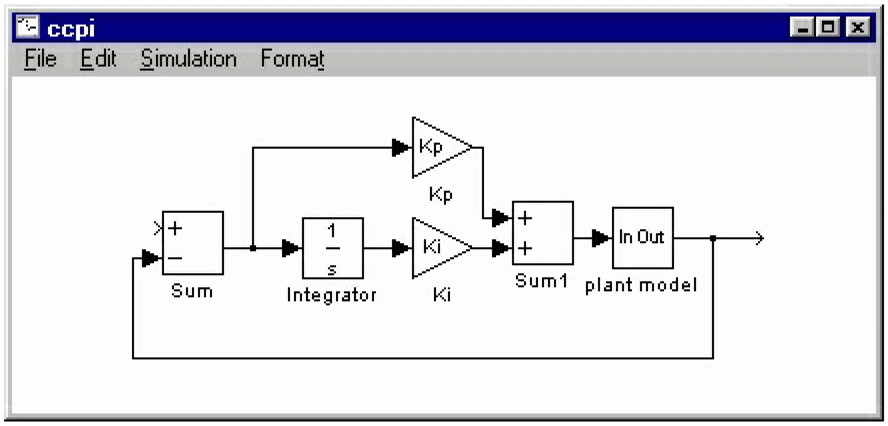
- Insert an integrator after the summer and connect them with a line.
- Insert and connect a gain block after the integrator to provide the integral gain.
- Label this integrator Ki and assign it a value of Ki.
- Insert a new Gain block and connect it with a line tapped off the output of the Sum block.

- Label this gain Kp and assign it a value of Kp.



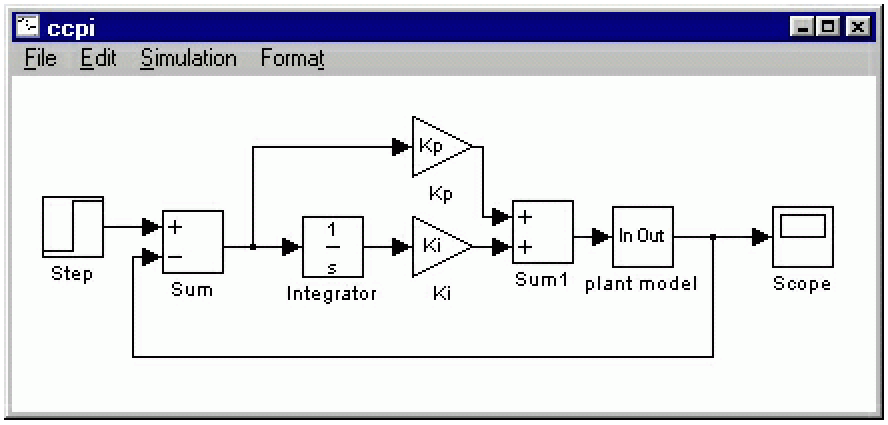
Now we will add the proportional and integral components and apply the sum to the plant.

- Insert a summer between the Ki block and the plant model and connect the outputs of the two gain blocks to the summer inputs.
- Connect the summer output to the input of the plant.



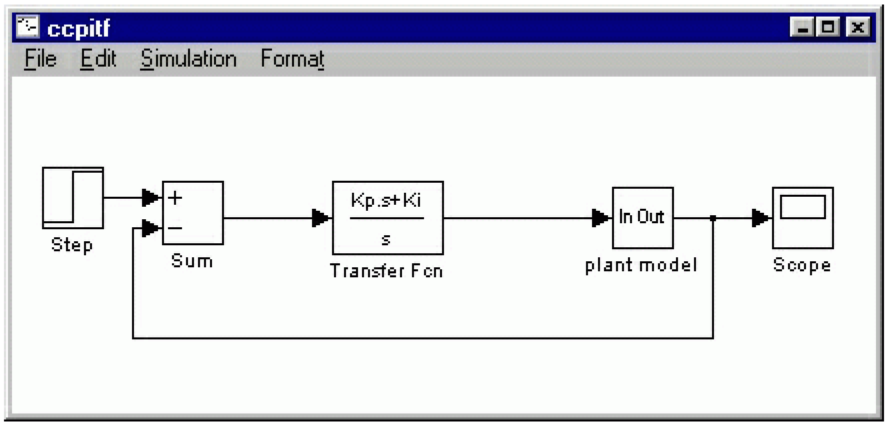
Finally, we will apply a step input and view the output on a scope.

- Attach a step block to the free input of the feedback Sum block.
- Attach a Scope block to the plant output.
- Double-click the Step block and set the Step Time to "0" and the Final Value to "u". This allows the input magnitude to be changed outside of simulink.



You can download our version of the closed-loop system here.

In this example, we constructed a PI controller from fundamental blocks. As an alternative, we could have used a Transfer Function block (from the Linear block library) to implement this in one step, as shown below.



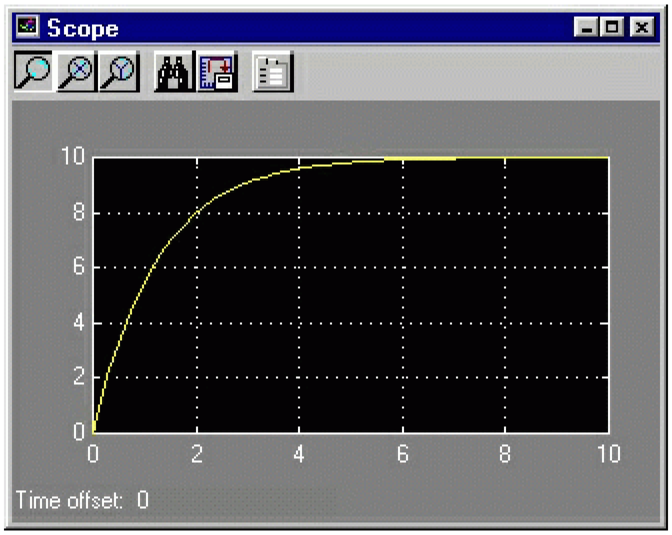
You can download this model here.

Closed-loop response

To simulate this system, first, an appropriate simulation time must be set. Select Parameters from the Simulation menu and enter "10" in the Stop Time field. The design requirements included a rise time of less than 5 sec, so we simulate for 10 sec to view the output. The physical parameters must now be set. Run the following commands at the MATLAB prompt:

```
m=1000;
b=50;
u=10;
Kp=800;
Ki=40;
```

Run the simulation (Ctrl-t or Start on the Simulation menu). When the simulation is finished, double-click on the scope and hit its autoscale button. You should see the following output.


Scope
10
8
6
4
2
0
0
2
4
6
8
10
Time offset: 0