Control Tutorials for MATLAB® and Simulink®

Simulink Modeling Tutorial

In Simulink, it is very straightforward to represent a physical system or a model. In general, a dynamic system can be constructed from just basic physical laws. We will demonstrate through an example. __

Train system

In this example, we will consider a toy train consisting of an engine and a car. Assuming that the train only travels in one direction, we want to apply control to the train so that it has a smooth start-up and stop, along with a constant-speed ride.

The mass of the engine and the car will be represented by M1 and M2, respectively. The two are held together by a spring, which has the stiffness coefficient of k. F represents the force applied by the engine, and the Greek letter, mu (which will also be represented by the letter u), represents the coefficient of rolling friction.
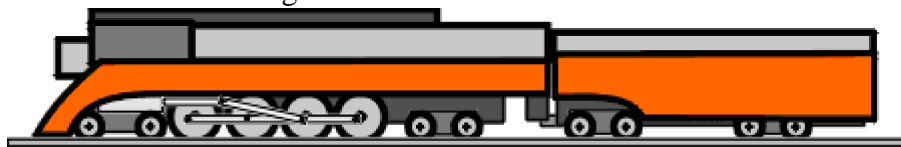
Free body diagram and Newton's law

The system can be represented by following Free Body Diagrams.
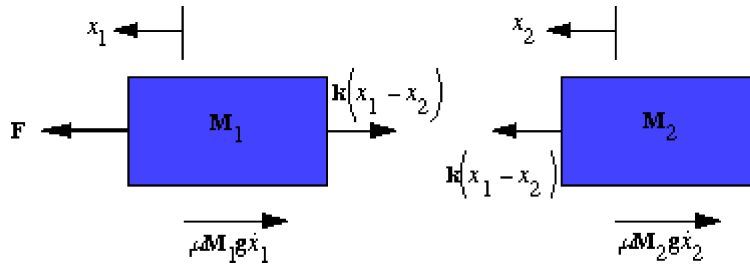
From Newton's law, you know that the sum of forces acting on a mass equals the mass times its acceleration. In this case, the forces acting on M1 are the spring, the friction and the force applied by the engine. The forces acting on M2 are the spring and the friction. In the vertical direction, the gravitational force is canceled by the normal force applied by the ground, so that there will be no acceleration in the vertical direction. We will begin to construct the model simply from the expressions:

```
Sum(forces_on_M1)=M1*x1''
Sum(forces_on_M1)=M1*x1''
```

Constructing The Model

This set of system equations can now be represented graphically, without further manipulation. First, we will construct two copies (one for each mass) of the expressions sum_F=Ma or a=1/M*sum_F. Open a new model window, and drag two Sum blocks (from the Linear library), one above the other. Label these Sum blocks "Sum_F1" and "Sum_F2".
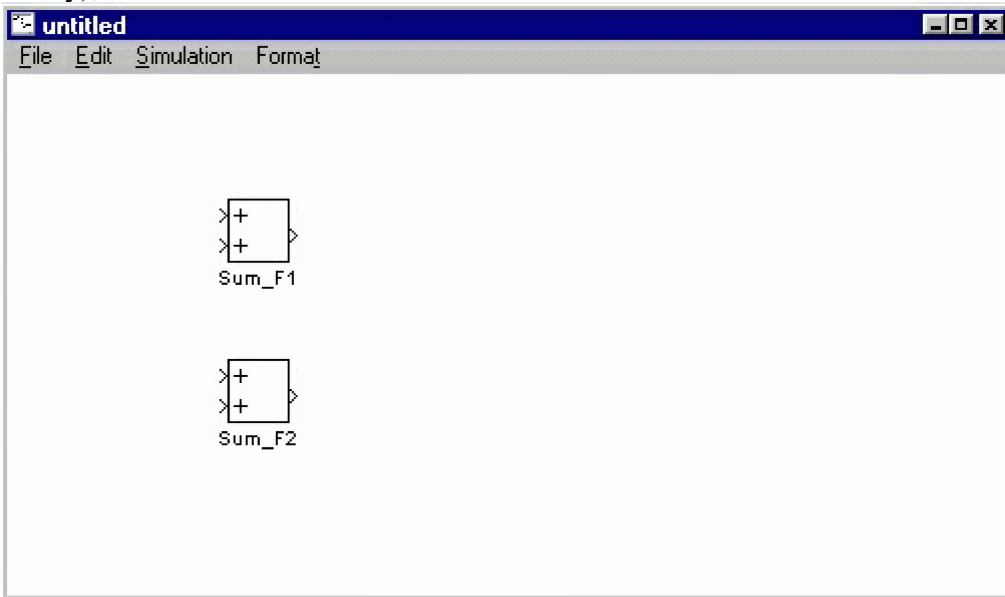
The outputs of each of these Sum blocks represents the sum of the forces acting on each mass. Multiplying by 1/M will give us the acceleration. Drag two Gain blocks into your model and attach each one with a line to the outputs of the Sum blocks.
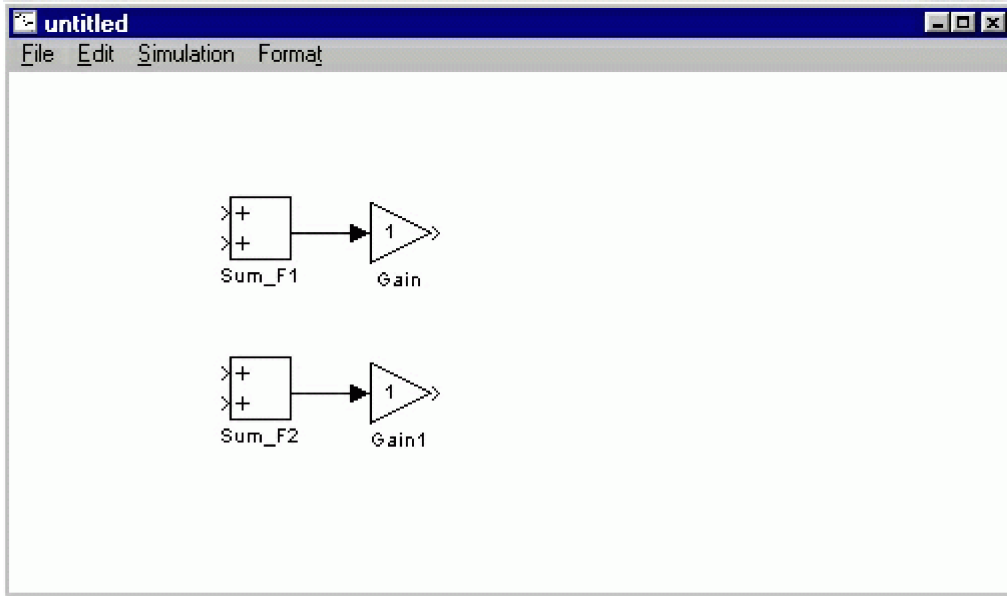
These Gain blocks should contain 1/M for each of the masses. We will be taking these variab
as M1 and M2 from the MATLAB environment, so we can just enter the variab in the Gain
blocks. Double-click on the upper Gain block and enter the following into the Gain field.

```
1/M1
```

Similarly, change the second Gain block to the following.

```
1/M2
```

Now, you will notice that the gains did not appear in the Gain blocks, and just "-K-" shows
up. This is because the blocks are two small on the screen to show 1/M2 inside the triangle.
The blocks can be resized so that the actual gain can be seen. To resize a block, select it by
clicking on it once. Small squares will appear at the corners. Drag one of these squares to
stretch the block.
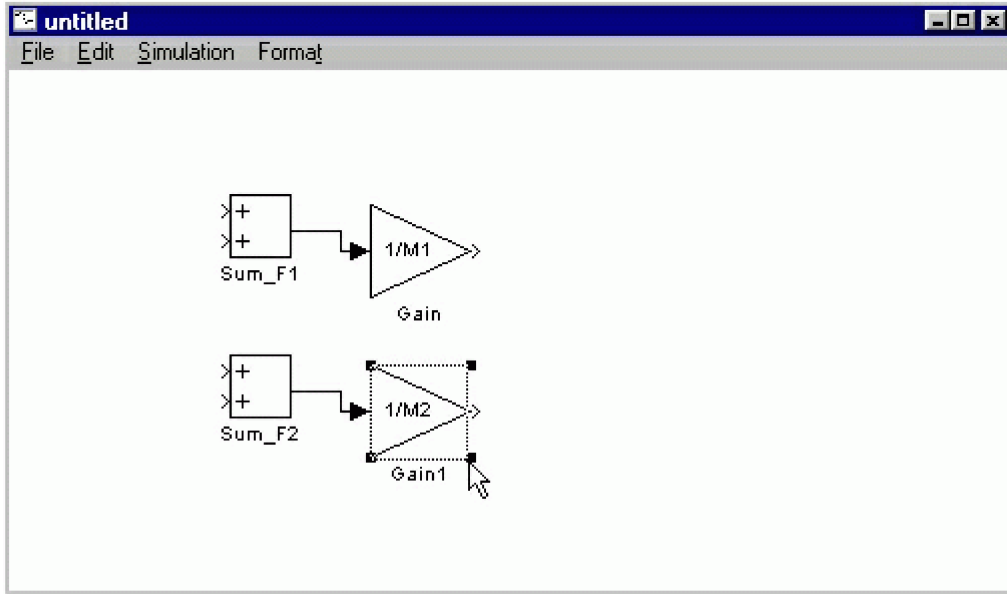
When the Gain blocks are of sufficient size to display the actual gains, re-align them with the signal line output from the Sum blocks. Also, label these two Gain blocks "a1" and "a2".
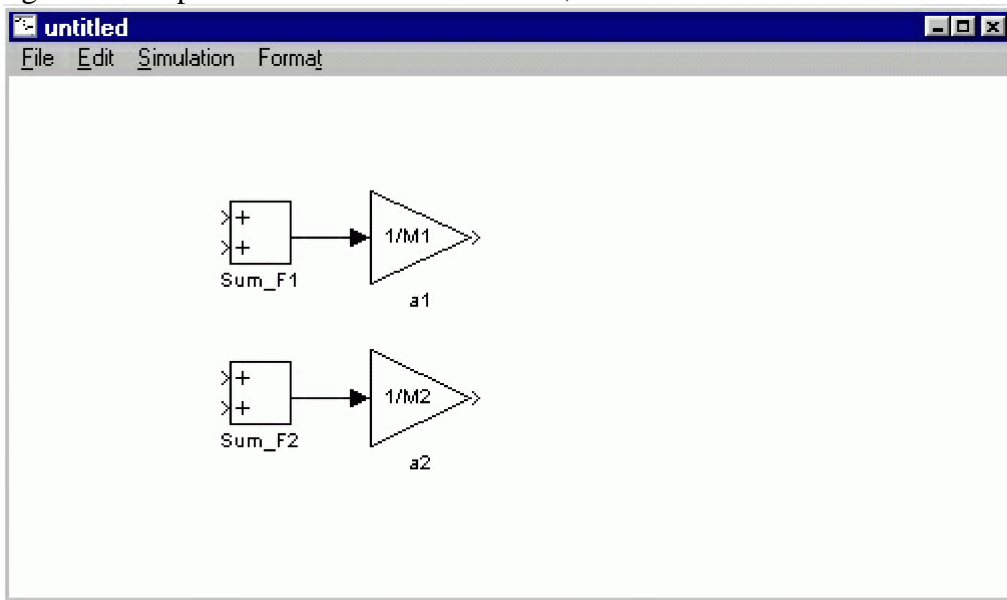

The outputs of these gain blocks are the accelerations of each of the masses. We are interested in both the velocities and the positions of the masses. Since velocity is the integral of acceleration, and position is the integral of velocity, we can generate these signals using integrator blocks. Drag two integrator blocks into your model for each of the two accelerations. Connect them with lines in two chains as shown below. Label these integrators "v1", "x1", "v2", and "x2" since these are the signals these integrators will generate.

CTMS: Simulink Modeling Tutorial
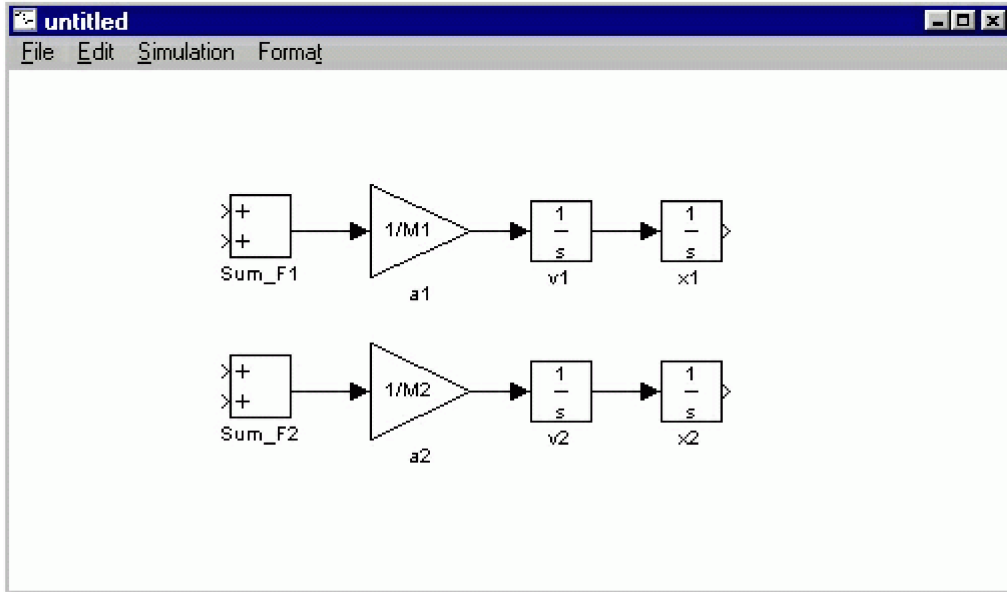
Now, drag two Scopes from the Sinks library into your model and connect them to the outputs of these integrators. Label them "View_x1" and "View_x2".
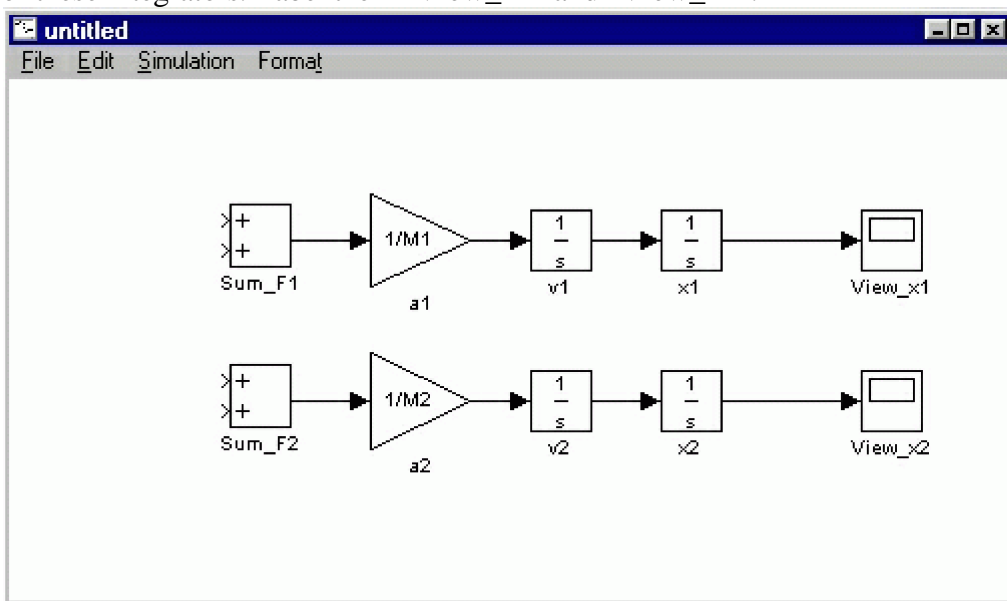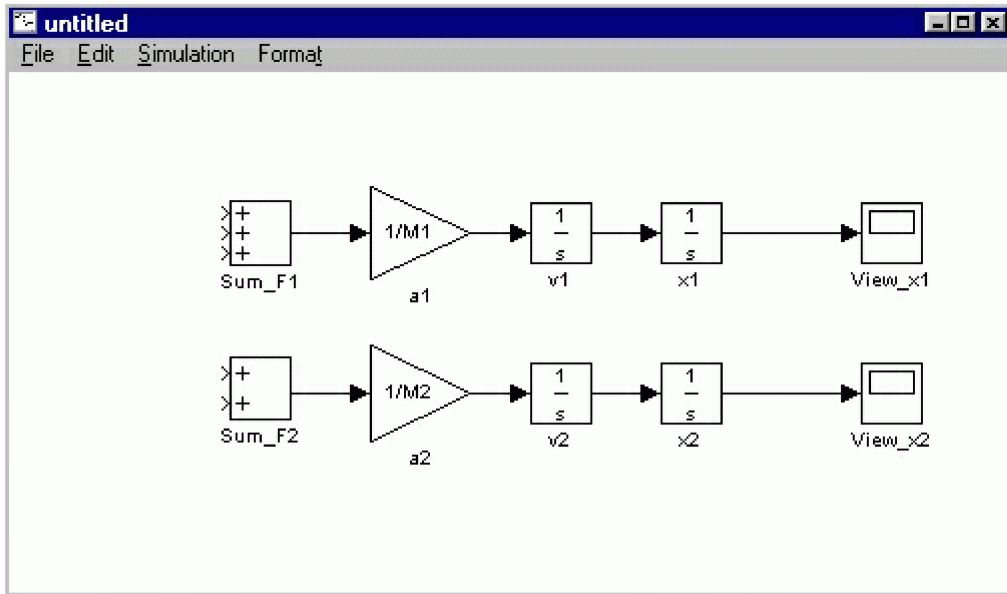

Now we are ready to add in the forces acting on each mass. First, you need to adjust the inputs on each Sum block to represent the proper number (we will worry about the sign later) of forces. There are a total of 3 forces acting on M1, so change the Sum_F1 block's dialog box entry to:
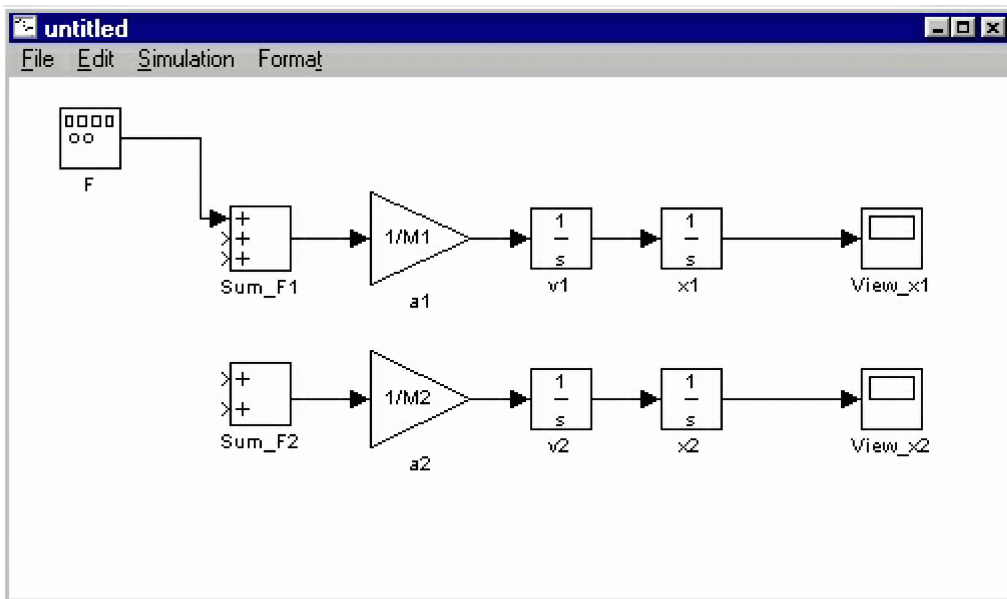
    +++

There are only 2 forces acting on M2, so we can leave Sum_F1 alone for now.

Page 5

The first force acting on M1 is just the input force, F. Drag a Signal Generator block from the Sources library and connect it to the uppermost input of the Sum_F1 block. Label the Signal Generator "F".

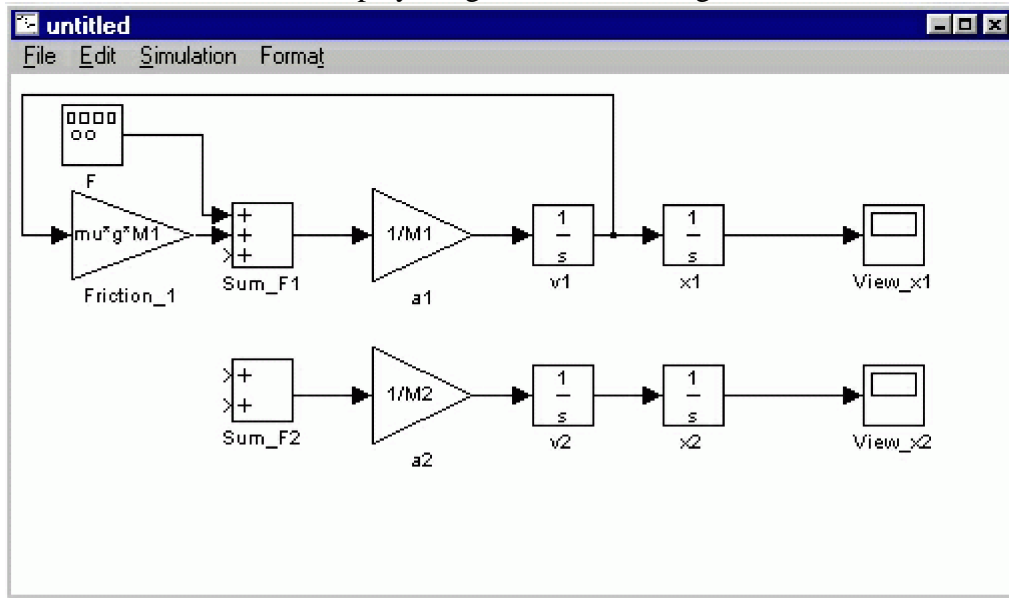The next force acting on M1 is the friction force. This force is equal to:

```
F_friction_1=mu*g*M1*v1
```

To generate this force, we can tap off the velocity signal and multiply by a gain, mu*g*M1. Drag a Gain block into your model window. Tap off the line coming from the v1 integrator and connect it to the input of the Gain block (draw this line in several steps if necessary). Connect the output of the Gain block to the second input of Sum_F1. Change the gain of this
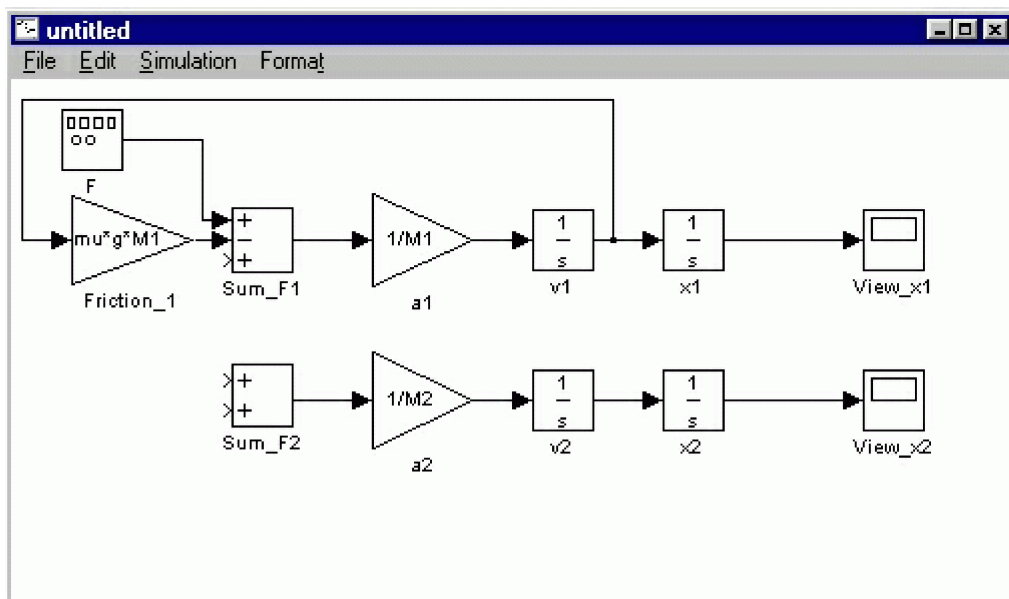
gain block to the following.

```
mu*g*M1
```

Resize the Gain block to display the gain and label the gain block Friction_1.

This force, however, acts in the *negative* x1-direction. Therefore, it must come into the Sum_F1 block with *negative* sign. Change the list of signs of Sum_F1 to

```
+-+
```


The last force acting on M1 is the spring force between masses. This is equal to:

~~The last force acting on M1 is the spring force between masses. This is equal to:~~

```
k*(x1-x2)
```

First, we need to generate (x1-x2) which we can then multiply by k to generate the force. Drag a Sum block below the rest of your model. Label it "(x1-x2)" and change its list of signs to

```
    -+
```

Since this summation comes from right to left, we need to flip the block around. Select the bloc by single-clicking on it and select **Flip** from the **Format** menu (or hit Ctrl-F). You should see the following.
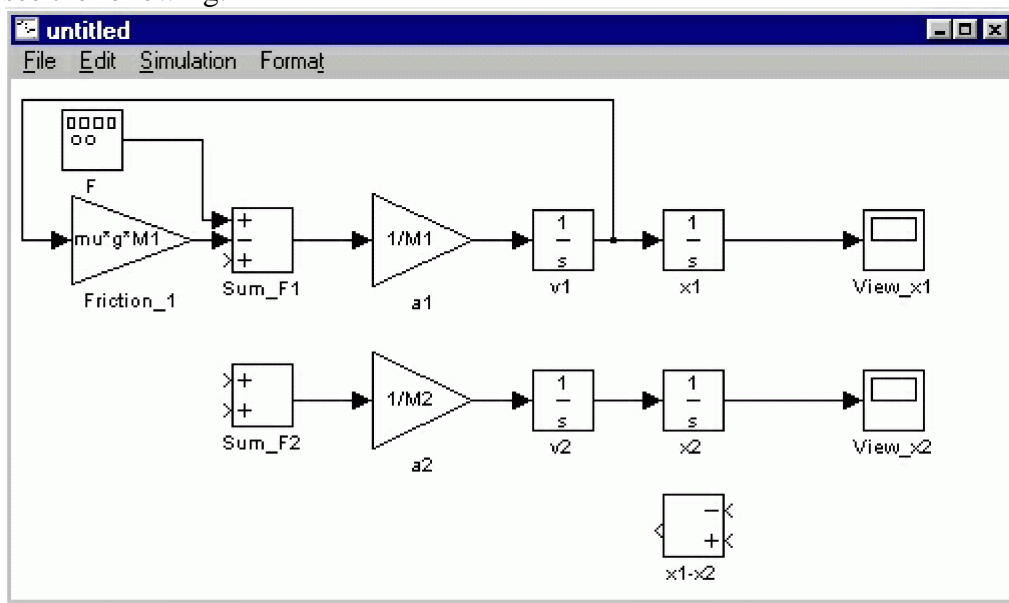

Now, tap off the x2 signal and connect it to the negative input of the (x1-x2) Sum block. Tap off the x1 signal and connect it to the positive input. This will cause the lines to cross. Lines may cross, but they are only actually connected where a small block appears (such as at a tap point).
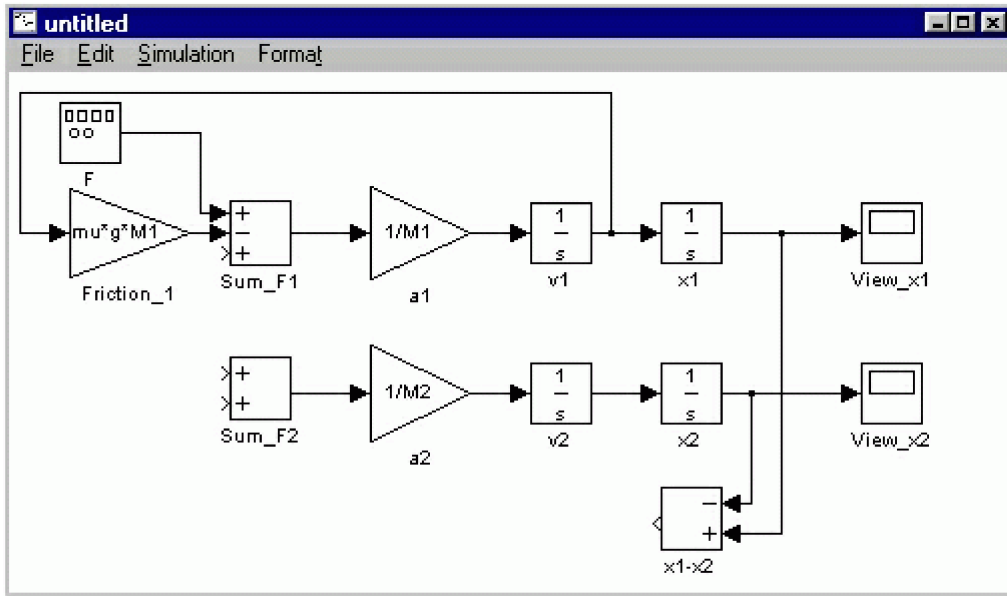
Now, we can multiply this position difference by the spring constant to generate the spring force. Drag a Gain block into your model to the left of the Sum blocks. Change it's value to k and label it "spring". Connect the output of the (x1-x2) block to the input of the spring block, and the output of the spring block to the third input of Sum_F1. Change the third sign of Sum_F1 to negative (use +--).
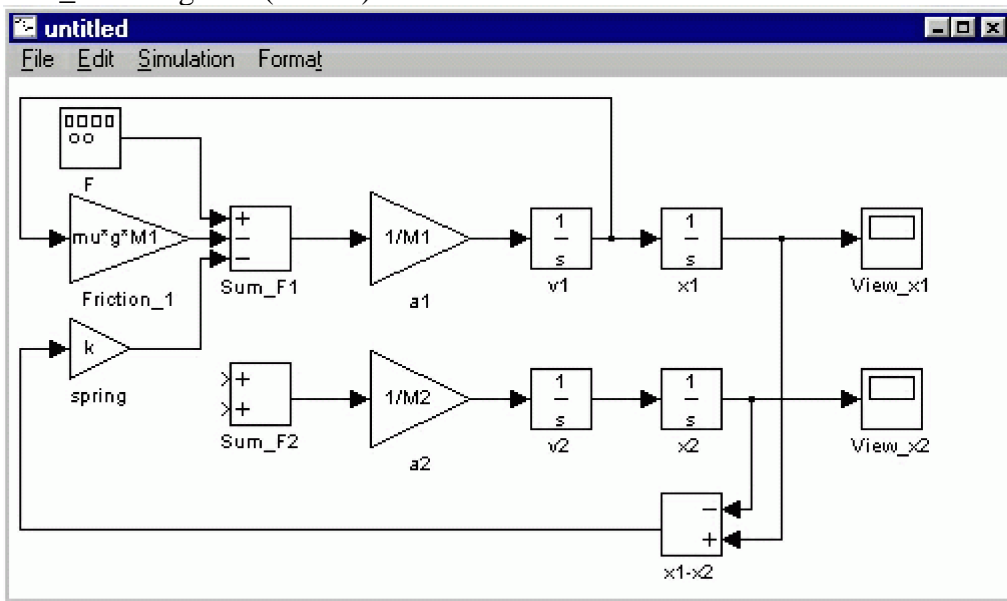

Now, we can apply forces to M2. For the first force, we will use the same spring force we just generated, except that it adds in with positive sign. Simply tap off the output of the spring block and connect it to the first input of Sum_F2.
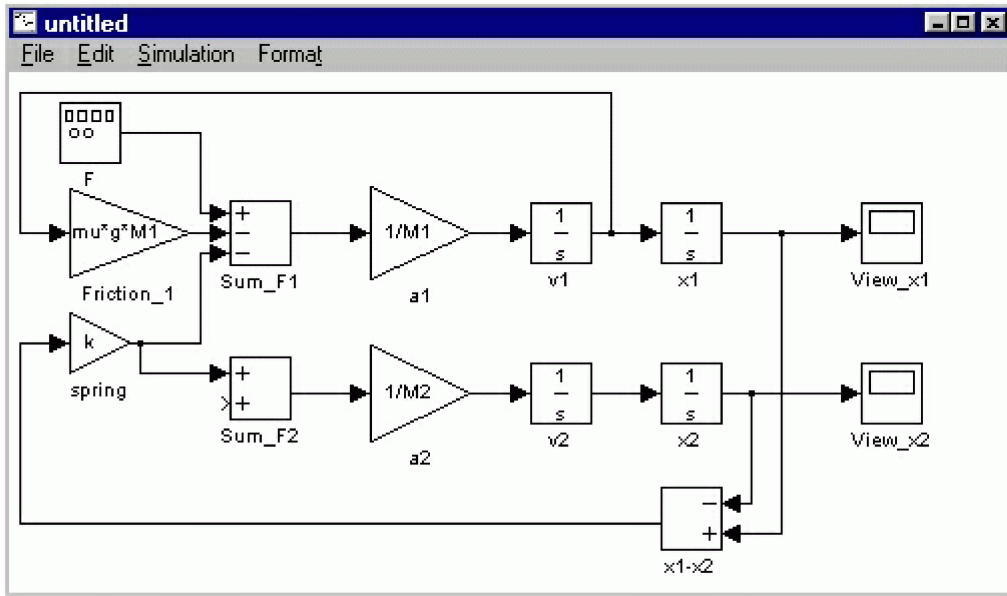
The last force to add in the the friction on M2. This is done in the exact same manner as the friction on M1, tapping off v2, multiplying by a gain of mu*g*M2 and adding to Sum_F2 with negative sign. After constructing this, you should have the following.
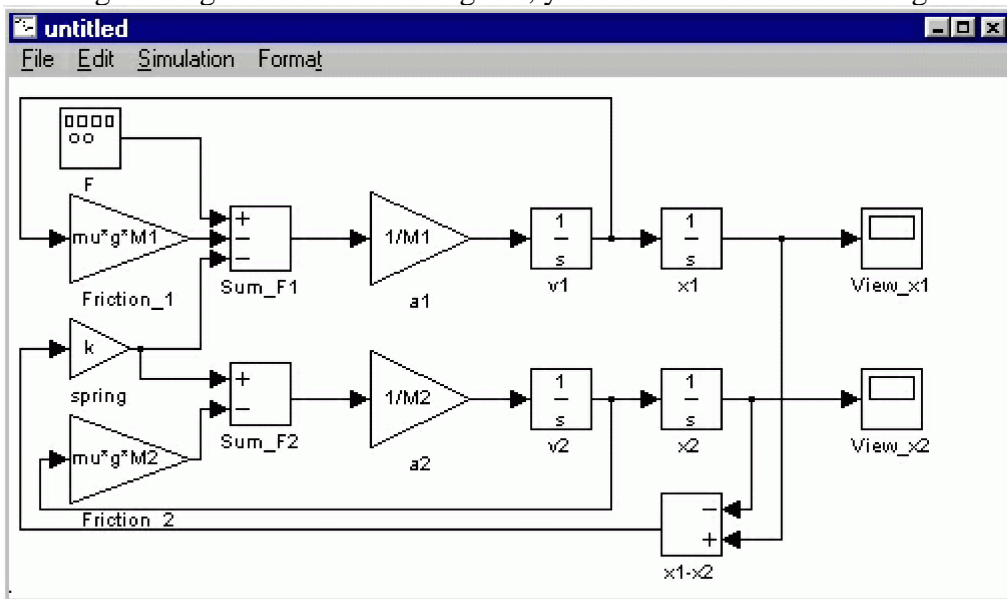
Now the model is complete. We simply need to supply the proper input and view the proper output. The input of the system will be the force, F, provided by the engine. We already have placed the function generator at the input. The output of the system will be the velocity of the engine. Drag a Scope block from the Sinks block library into your model. Tap a line off the output of the "v1" integrator block to view the output. Label the scope "View_v1".
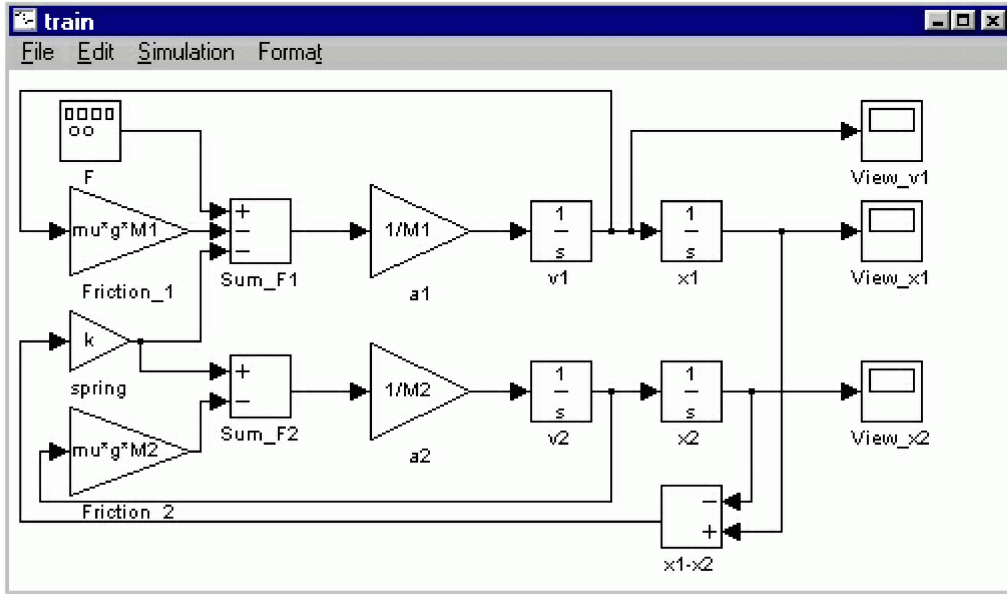
Now, the model is complete. Save your model in any file you like. You can download the completed model here. —

Running the Model

Before running the model, we need to assign numerical values to each of the variab used in the model. For the train system, let

- M1 = 1 kg
- M2 = 0.5 kg
- k = 1 N/sec
- F= 1 N
- u = 0.002 sec/m
- g = 9.8 m/s^2

Create an new m-file and enter the following commands.

```
M1=1;
M2=0.5;
k=1;
F=1;
mu=0.002;
g=9.8;
```

Execute your m-file to define these values. Simulink will recognize MATLAB variab for use in the model.

the model.

Now, we need to give an appropriate input to the engine. Double-click on the function generator (F block). Select a square wave with frequency .001Hz and amplitude -1 (positive amplitude steps negative before stepping positive).
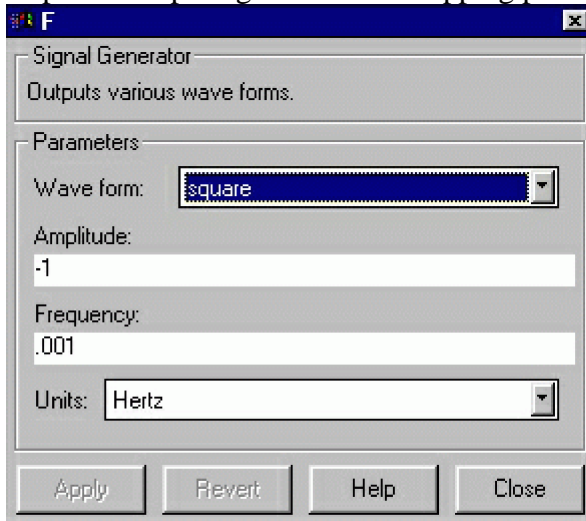

The last step before running the simulation is to select an appropriate simulation time. To view one cycle of the .001Hz square wave, we should simulate for 1000 seconds. Select **Parameters** from the **Simulation** menu and change the Stop Time field to 1000. Close the dialog box.

Now, run the simulation and open the View_v1 scope to examine the velocity output (hit autoscale). The input was a square wave with two steps, one positive and one negative. Physically, this means the engine first went forward, then in reverse. The velocity output reflects this.
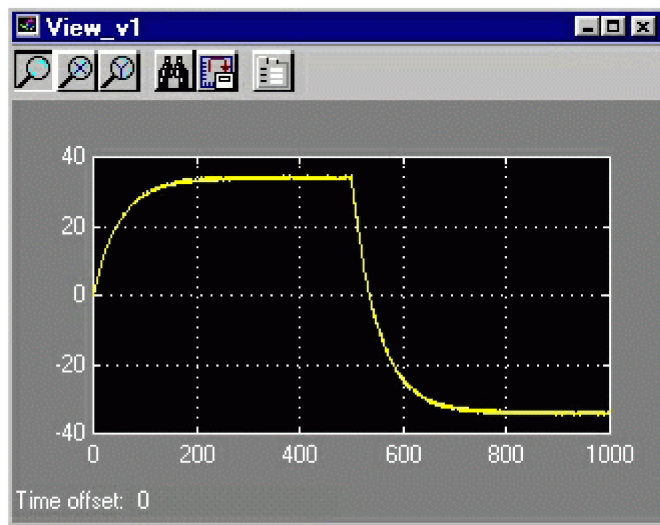
Obtaining MATLAB Model

We can now extract a MATLAB model (state-space or transfer function) from out simulink model. In order to do this, delete the View_v1 scope and put an Out Block (from the Connections library) in its place. Also, delete the F function generator block and put an In Block (from the Connections library) in its place. The in and out blocks define the input and output of the system we would like to extract. For a detailed description of this process and other ways of integrating MATLAB with Simulink, click here.
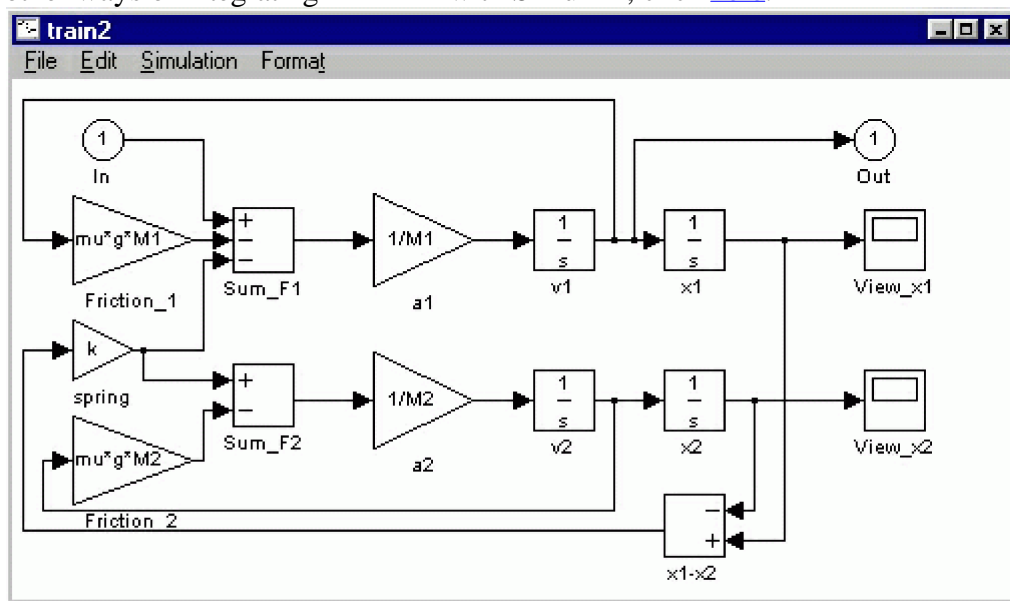
Save this model as **train2.mdl** or download our version here. Now, we can extract the model into MATLAB. Enter the following command at the MATLAB command window to extract a state-space model.

```
[A,B,C,D]=linmod('train2')
```

You should see the following output which shows a state-space model of your Simulink model.

```
A =

   -0.0196         0    1.0000   -1.0000
         0   -0.0196   -2.0000    2.0000
         0    1.0000         0         0
    1.0000         0         0         0


B =

    1
    0
```

```
     0
     0


C =

     1     0     0     0


D =

     0
```

To obtain a transfer function model, enter the following command at the MATLAB command prompt.

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

You will see the following output representing the transfer function of the train system.

```
num =

        0    1.0000    0.0196    2.0000    0.0000


den =

    1.0000    0.0392    3.0004    0.0588    0.0000
```

These models are equivalent (although the states are in different order) to the model obtained by hand in the MATLAB tutorials.

Simulink Examples
Cruise Control | Motor Speed | Motor Position | Bus Suspension | Inverted Pendulum | Pitch Controller | Ball and Beam

Tutorials
MATLAB Basics | MATLAB Modeling | PID | Root Locus | Frequency Response | State Space | Digital Control | Simulink Basics | Simulink Modeling | Examples