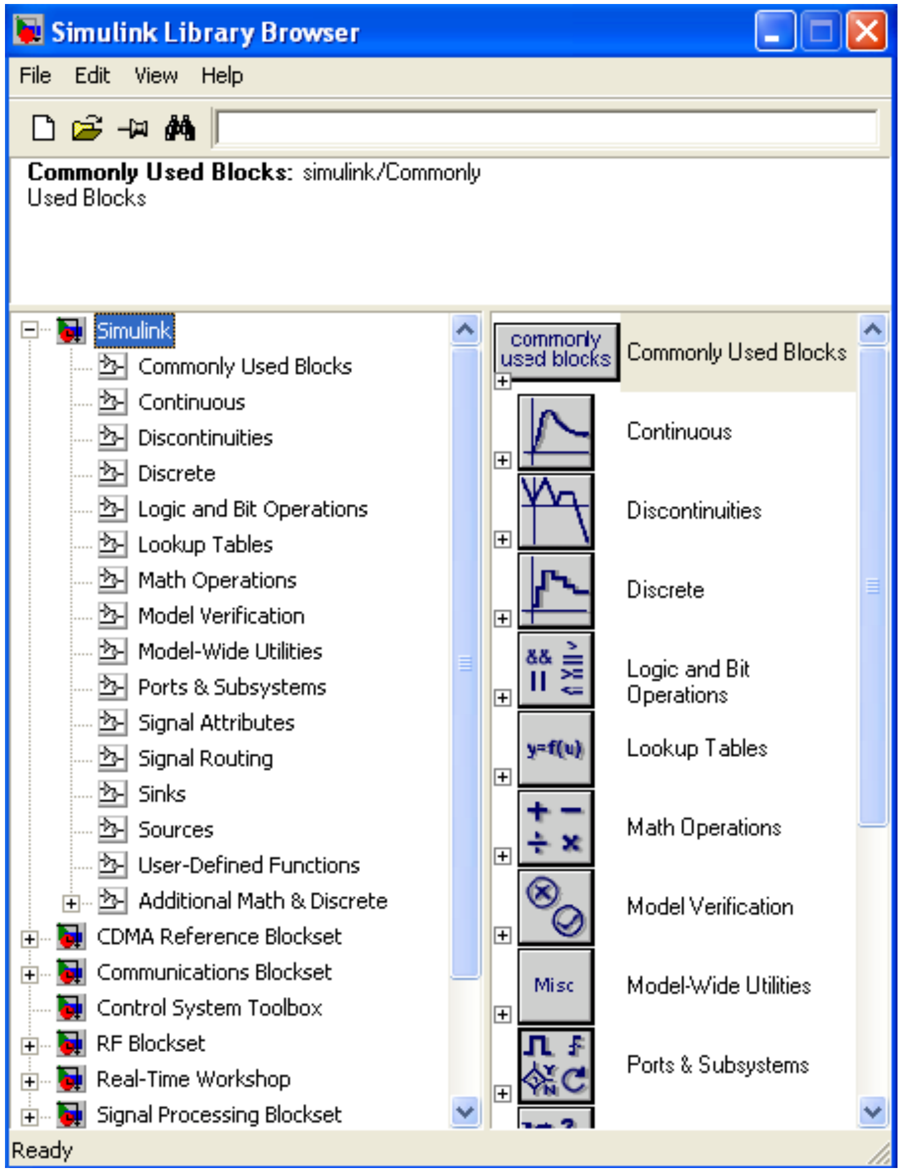
MATLAB Simulink

What is Simulink

- Simulink is an input/output device GUI block diagram simulator.
- Simulink contains a Library Editor of tools from which we can build input/output devices and continuous and discrete time model simulations.
- To open Simulink, type in the MATLAB work space
  - >>simulink

The Library Editor

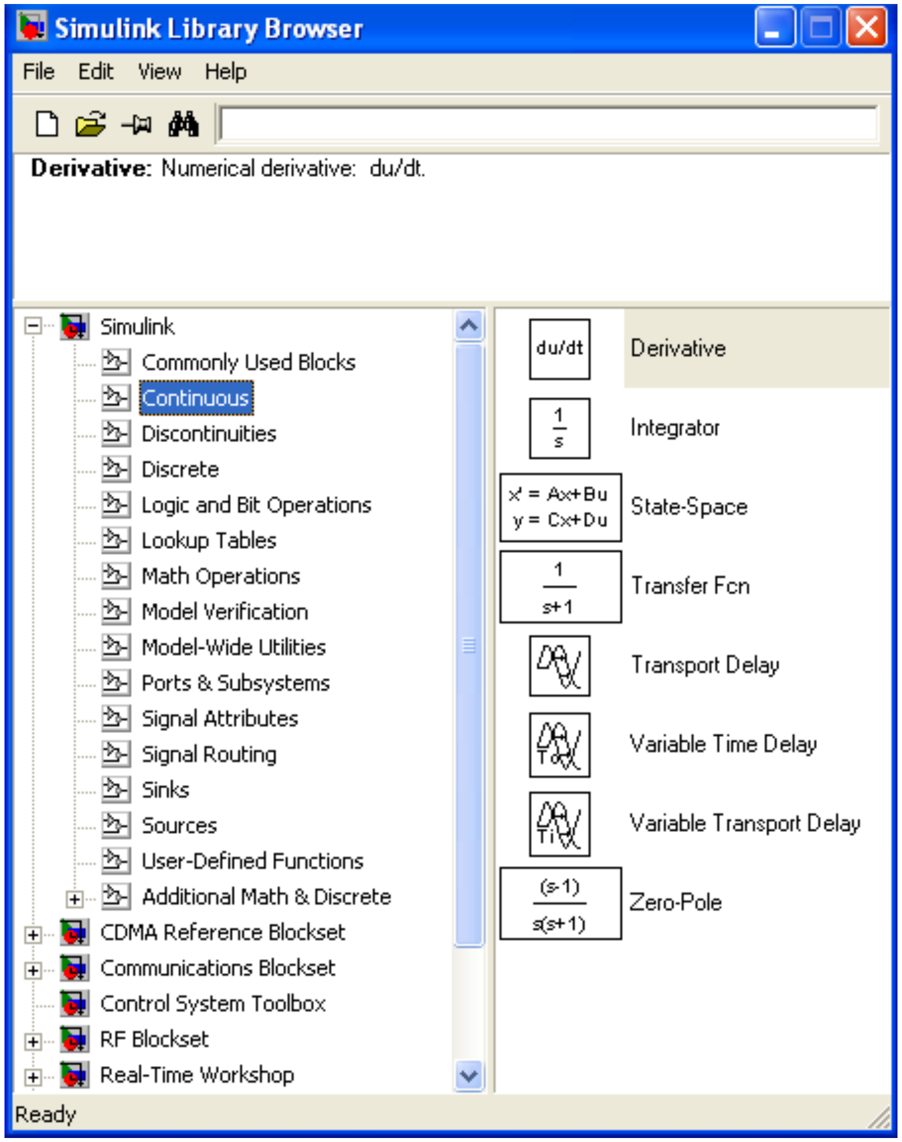
- >>simulink
- Simulink opens with the Library Browser
- Library Browser is used to build simulation models



Commonly Used Blocks: simulink/Commonly Used Blocks

Continuous Elements

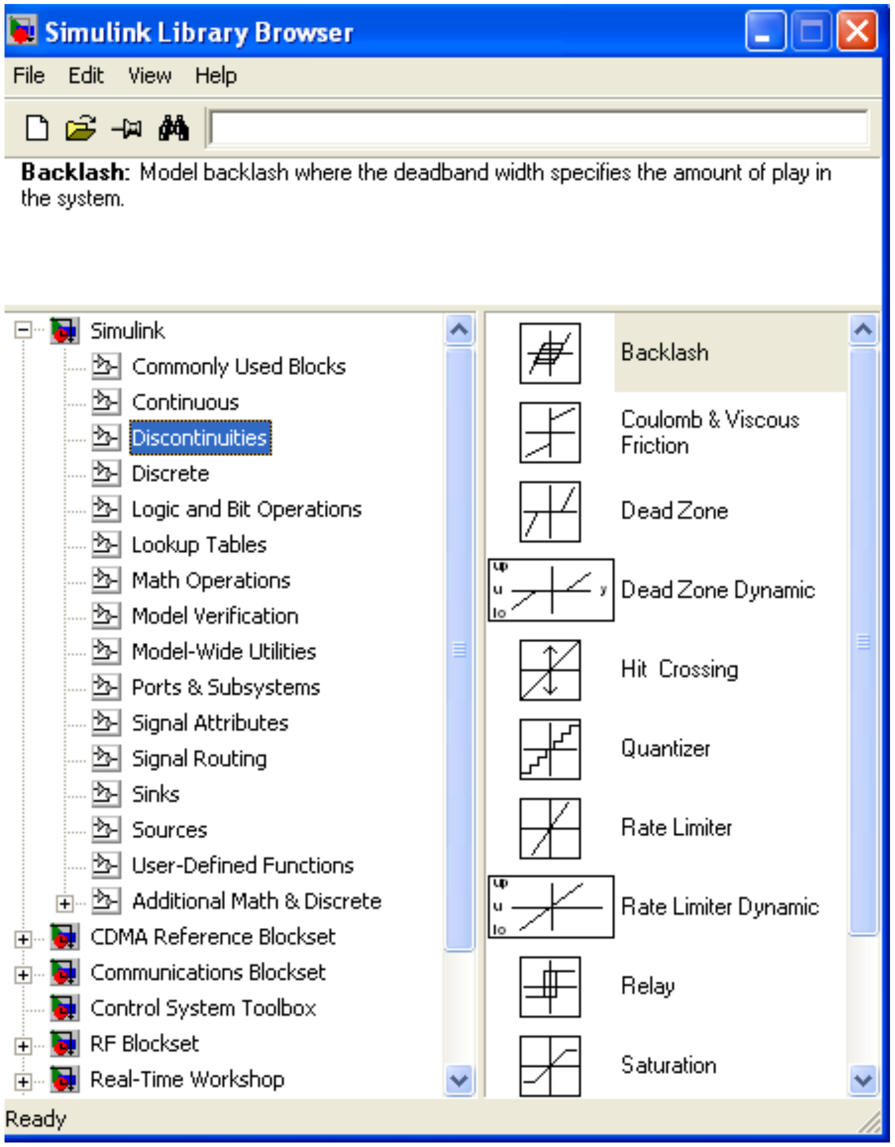
- Contains continuous system model elements



Derivative: Numerical derivative: du/dt.

Discontinuous Elements

- Contains discontinuous system model elements



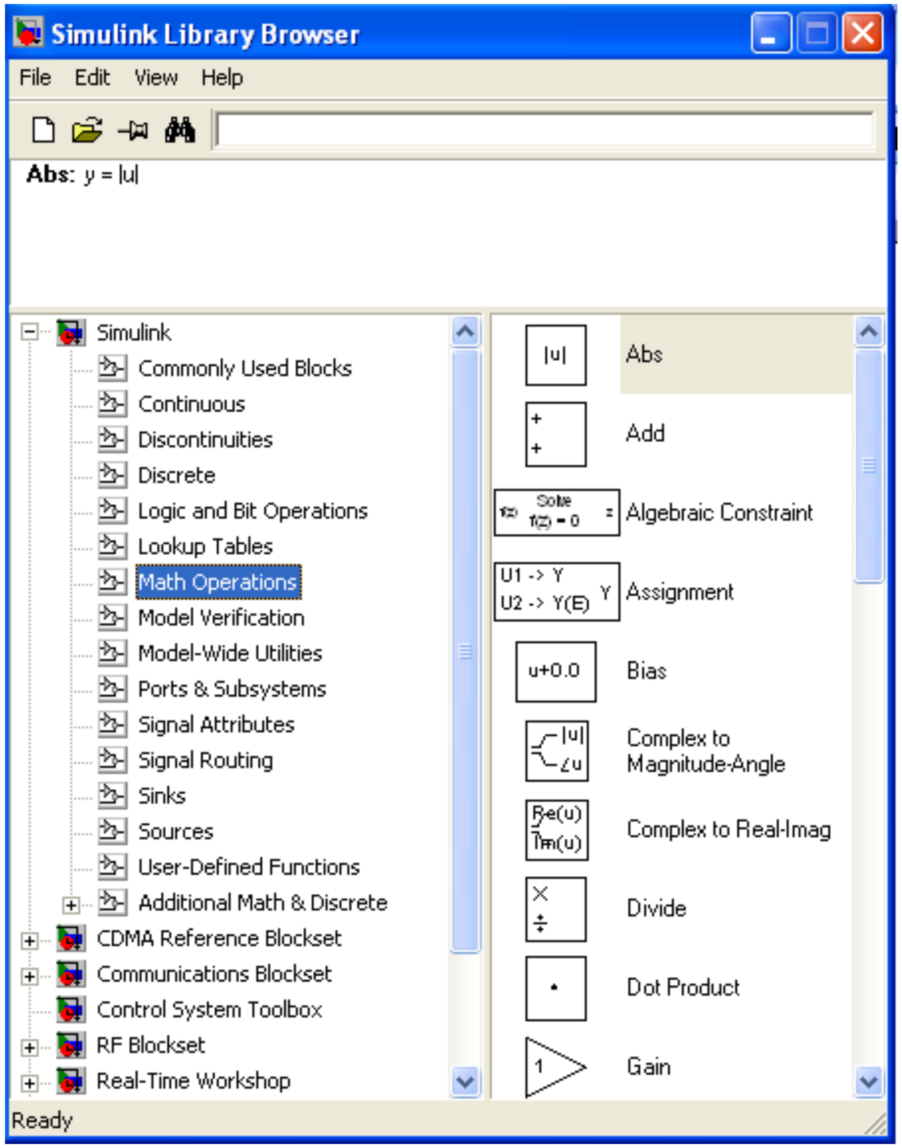
Simulink Library Browser

File Edit View Help

Backlash: Model backlash where the deadband width specifies the amount of play in the system.

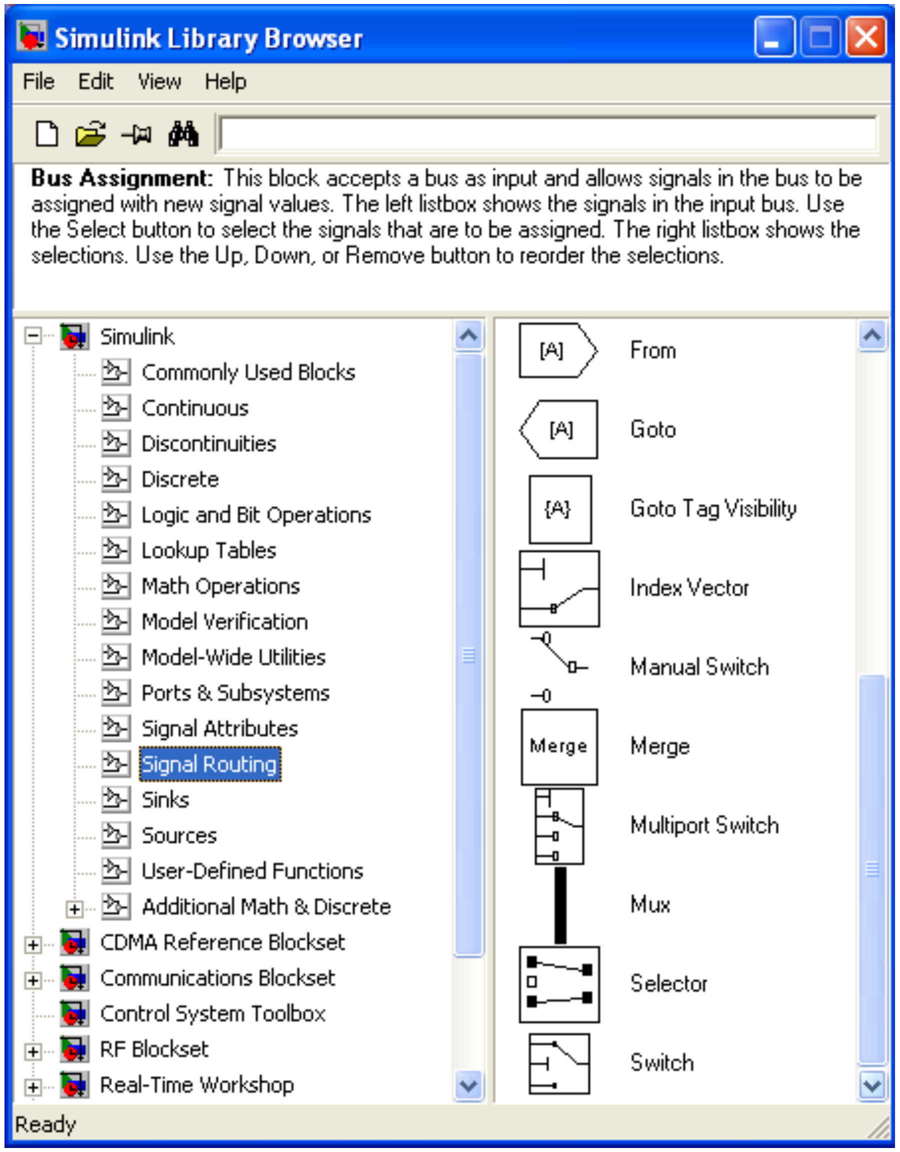
Math Operations

- Contains list of math operation elements



Signal Routing

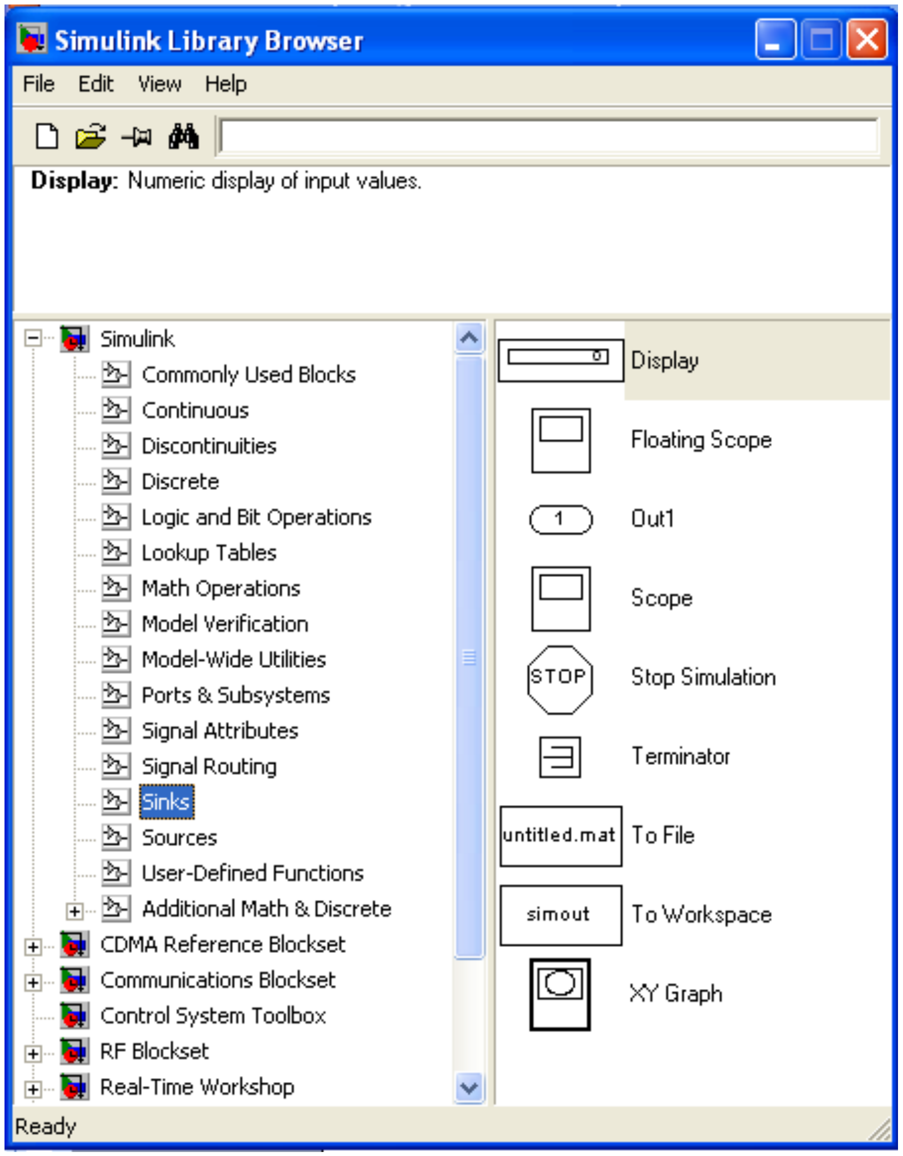
- Signal Routing elements



Bus Assignment: This block accepts a bus as input and allows signals in the bus to be assigned with new signal values. The left listbox shows the signals in the input bus. Use the Select button to select the signals that are to be assigned. The right listbox shows the selections. Use the Up, Down, or Remove button to reorder the selections.

Sink Models

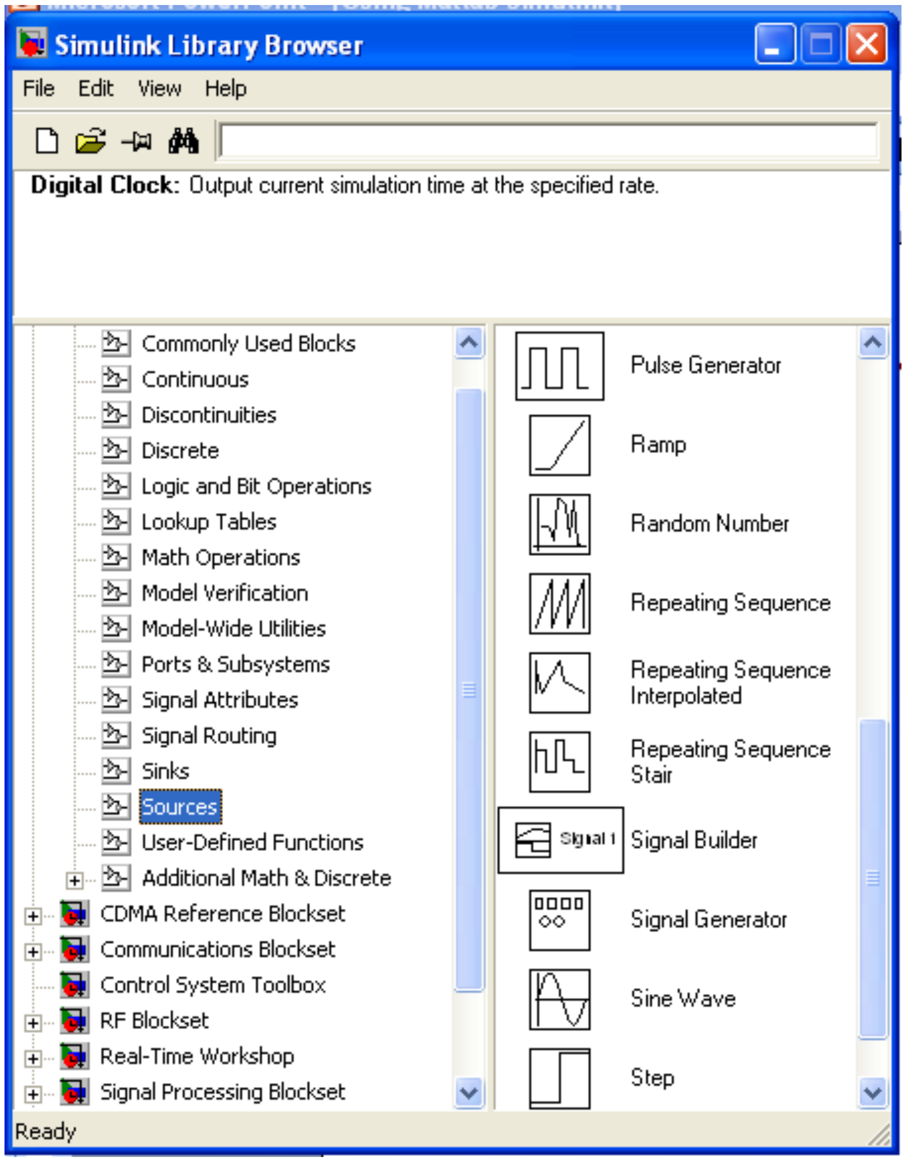
- Sinks: sink device elements. Used for displaying simulation results



Display: Numeric display of input values.

Signal Routing

- Sources: provides list of source elements used for model source functions.



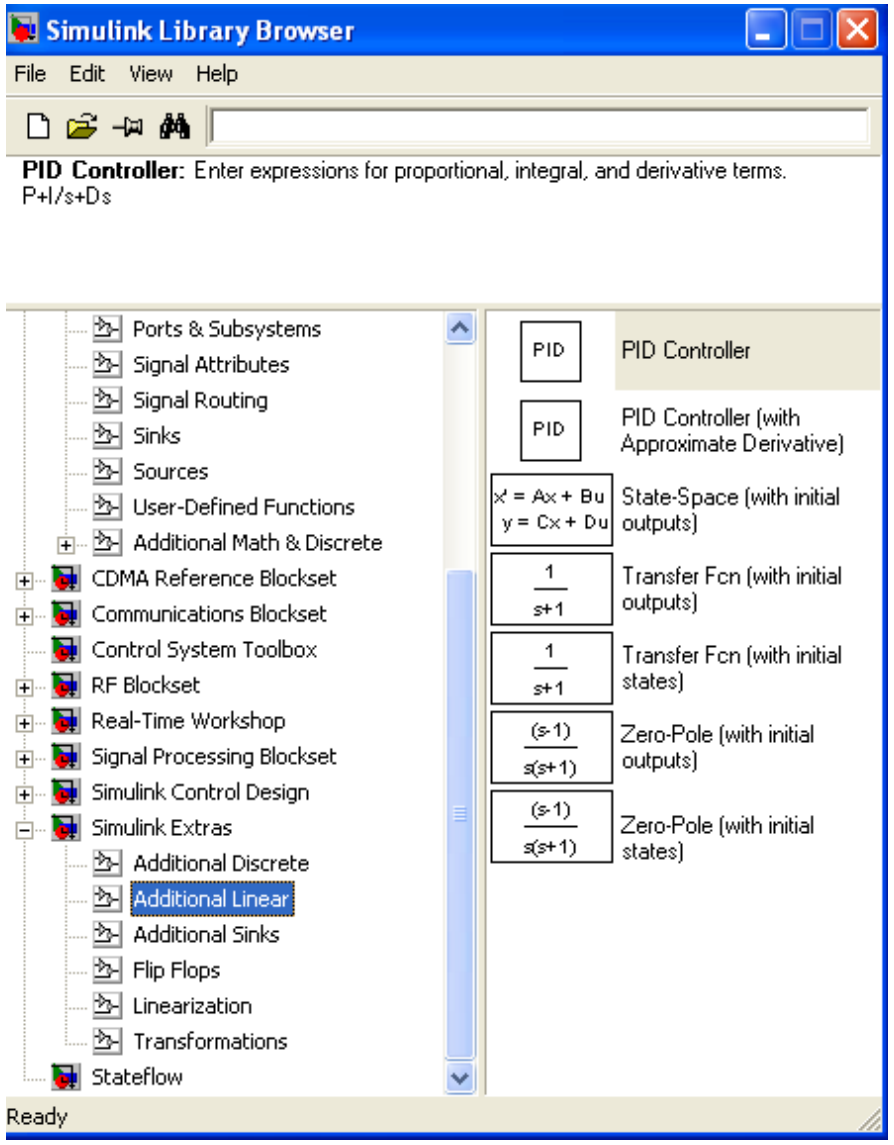
Simulink Library Browser

File Edit View Help

Digital Clock: Output current simulation time at the specified rate.

Signal Routing

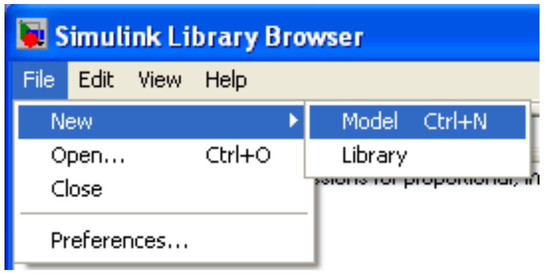
- Simulink Extras: additional linear provide added block diagram models. The two PID controller models are especially useful for PID controller simulation.

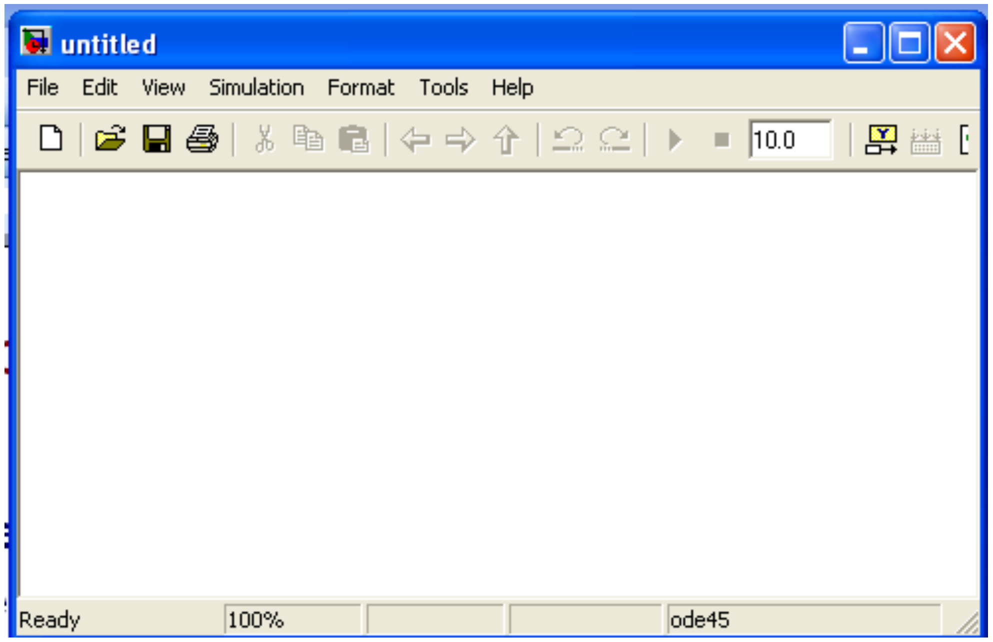


PID Controller: Enter expressions for proportional, integral, and derivative terms.
P+I/s+Ds

Building Models

- **Creating a New Model**
- To create a new model, click the **New** button on the Library Browser's toolbar
- This opens a new untitled model window.





Building Models (2)

- **Model elements** are added by selecting the appropriate elements from the Library Browser and dragging them into the Model window. Alternately, they may be copied from the Library Browser and pasted into the model window.
- To illustrate lets model the capacitor charging equation:

$$\frac{dv_C(t)}{dt} = -\frac{1}{RC}v_C(t) + \frac{1}{RC}u(t)$$

$u(t) = \text{The unit step function}$

Modeling the Capacitor System

To model the system we let $s = d/dt$ and rewrite the differential equation as

$$sV_C(s) = -\frac{1}{RC}V_C(s) + \frac{1}{RC}U(s)$$

Solving for $V_C(s)$ we get

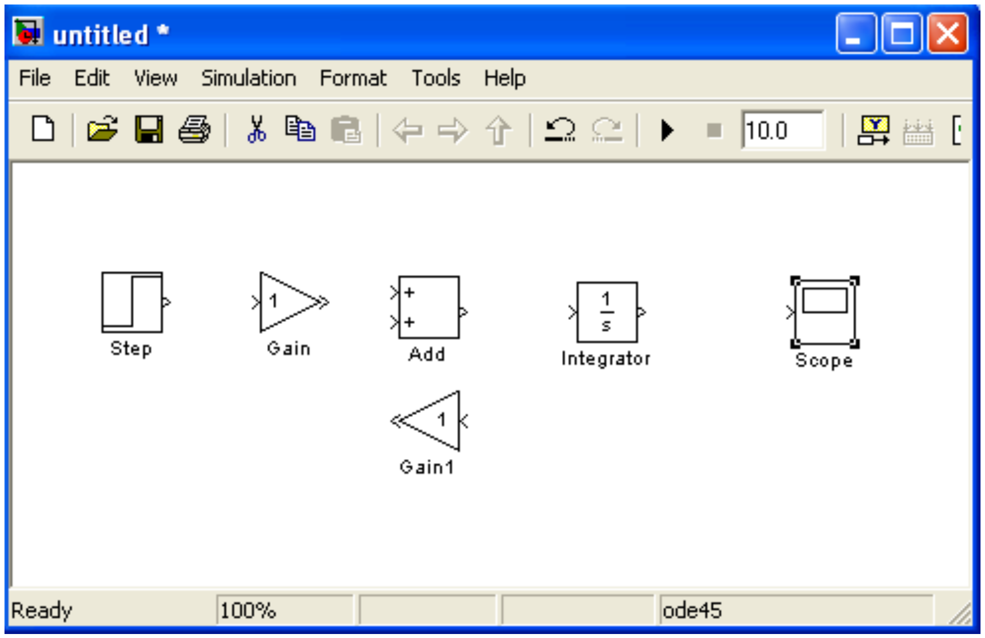
$$V_C(s) = \frac{1}{s}\left(\frac{1}{RC}U(s) - \frac{1}{RC}V_C(s)\right)$$

To model the system we need 1 integrator block, a summing junction block, 2 gain blocks, a step function block, and a sink block to display the results.

The next slide showes the model block diagram

Modeling the Capacitor System (2)

Unconnected system model



Connecting the System Blocks

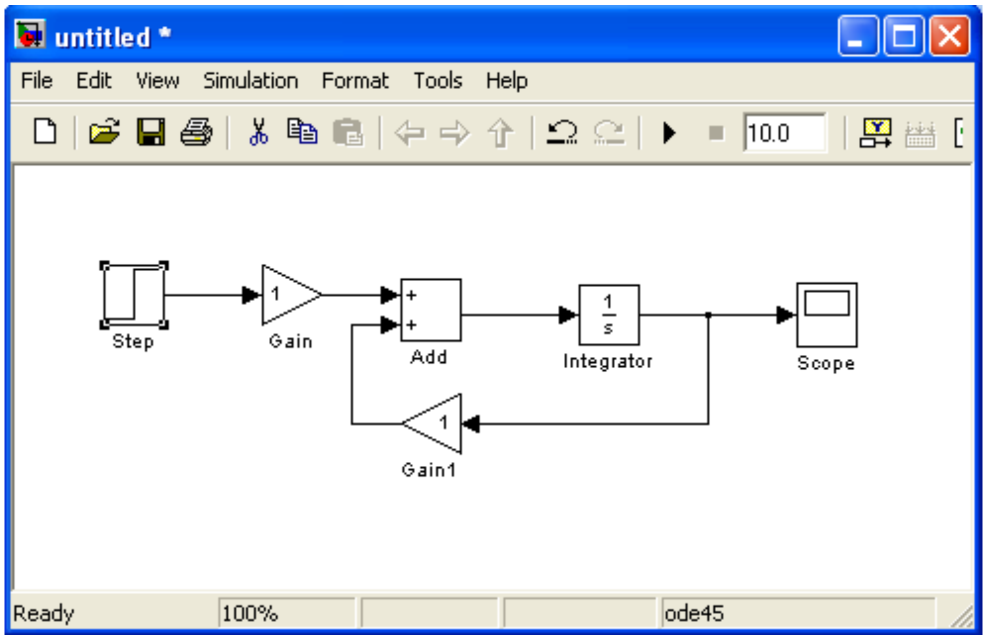
System blocks are connected in to ways; one way is the auto connect method which is done by 1st selecting the source block and holding down the Cntr key and selecting the destination block.

Simulink also allows you to draw lines manually between blocks or between lines and blocks. To connect the output port of one block to the input port of another block: Position the cursor over the first block's output port. The cursor shape changes to crosshairs. Right click and drag the crosshairs to the input of the destination block to make the connection.

The next slide shows the connected model

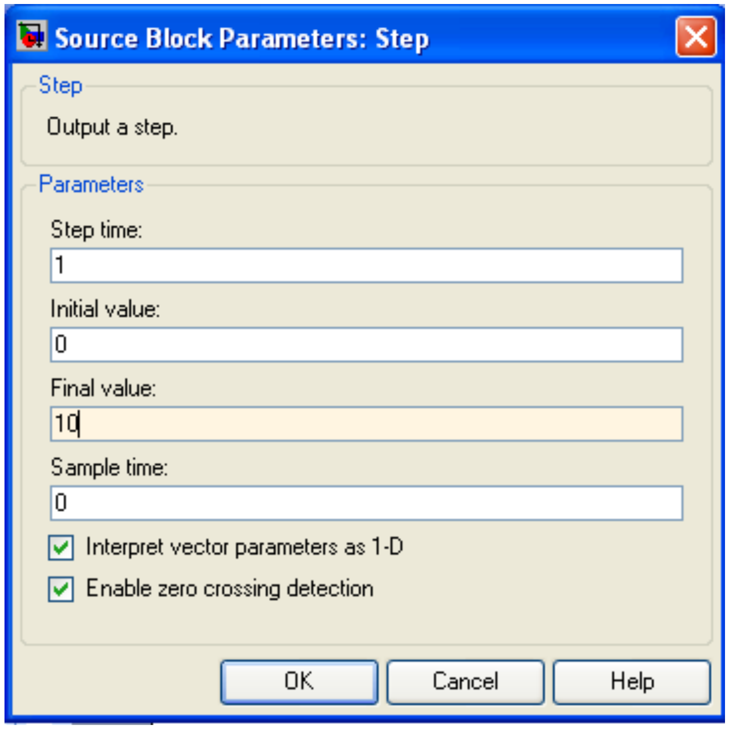
The Connected Capacitor Model

Connected system model



Editing the Capacitor Model

Now that we have connected the model, we need to edit it. This is done by selecting and double clicking to open the appropriate models and then editing. The blocks we need to edit are the step function, the two gain blocks, and the summing junction block.



Source Block Parameters: Step

Step

Output a step.

Parameters

Step time: 1

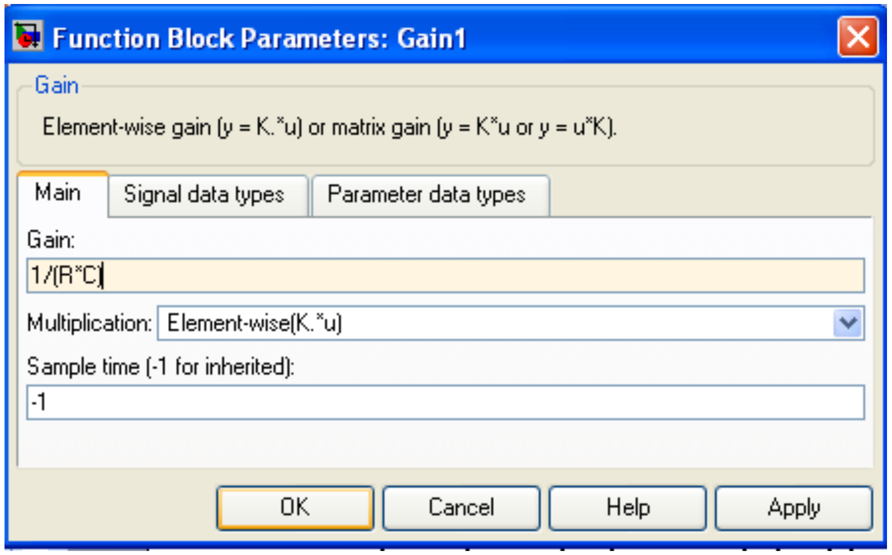
Initial value: 0

Final value: 10

Sample time: 0

[✓] Interpret vector parameters as 1-D

[✓] Enable zero crossing detection



Function Block Parameters: Gain1

Gain

Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).

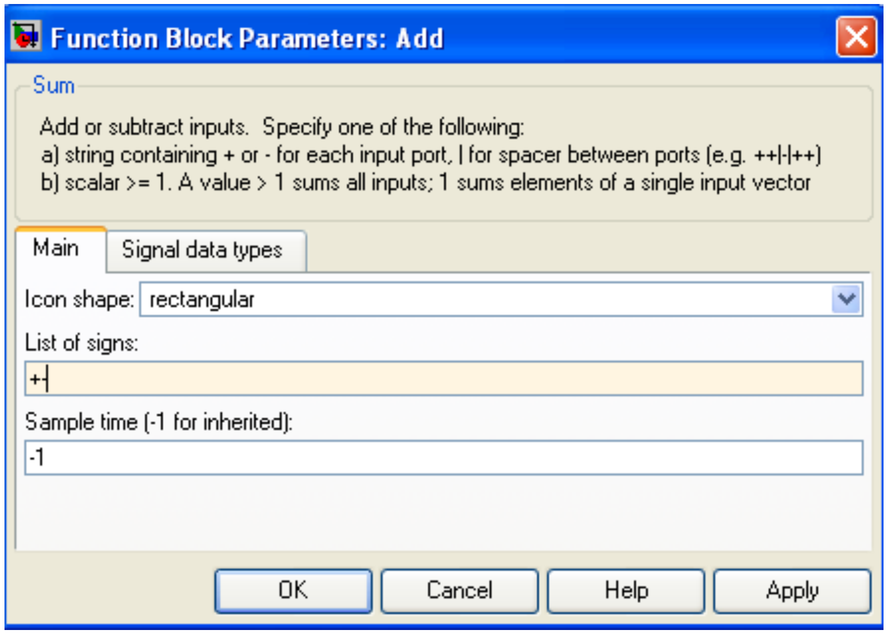
Gain: 1/(R*C)

Multiplication: Element-wise(K.*u)

Sample time (-1 for inherited): -1

Editing the Capacitor Model (2)

In editing the blocks note that we have set the height of the step function at 10, left R and C as variable (we will enter these from the workspace), and changed the sign of the summing junction input from + to -.



Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar >= 1. A value > 1 sums all inputs; 1 sums elements of a single input vector

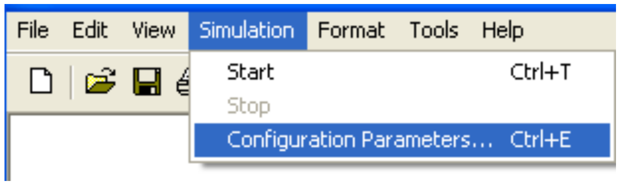
Icon shape: rectangular

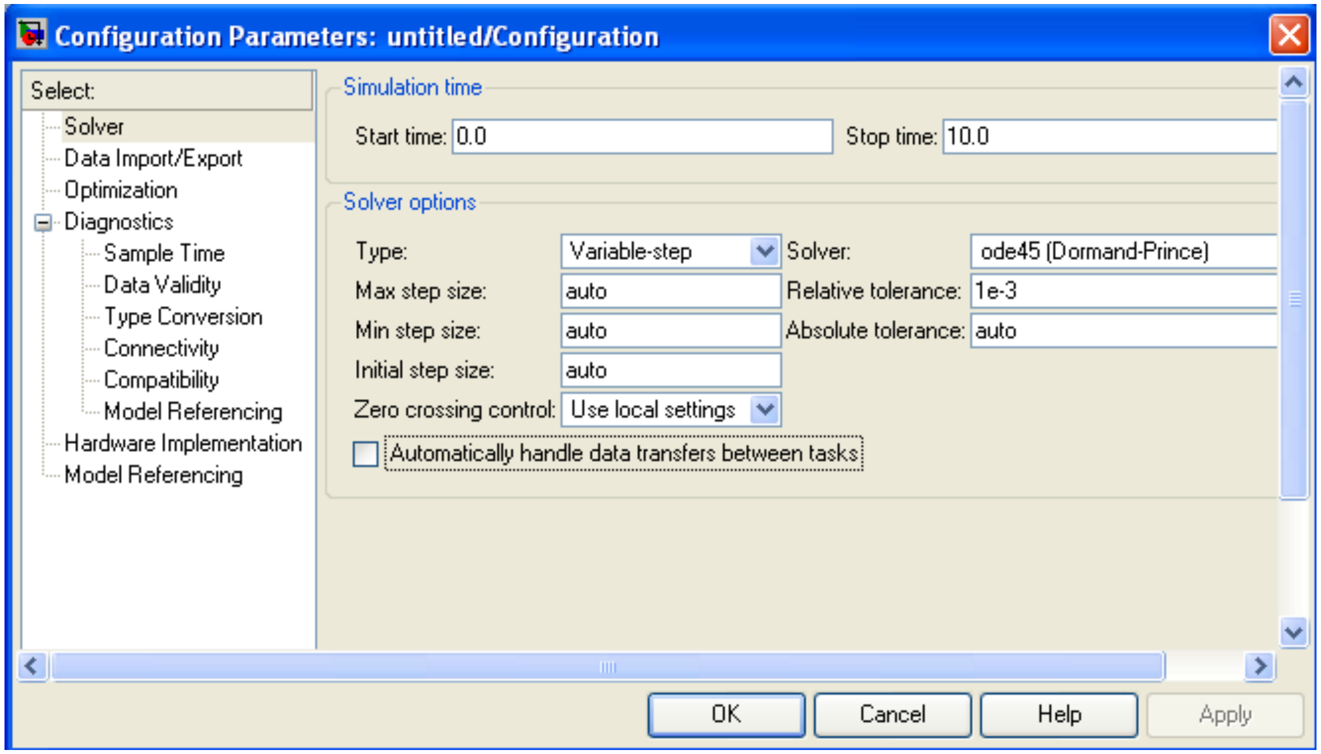
List of signs: +-

Sample time (-1 for inherited): -1

Controlling the Simulation

Simulation control is fixed by selecting Configuration Parameters




Configuration Parameters: untitled/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Max step size: auto Relative tolerance: 1e-3

Min step size: auto Absolute tolerance: auto

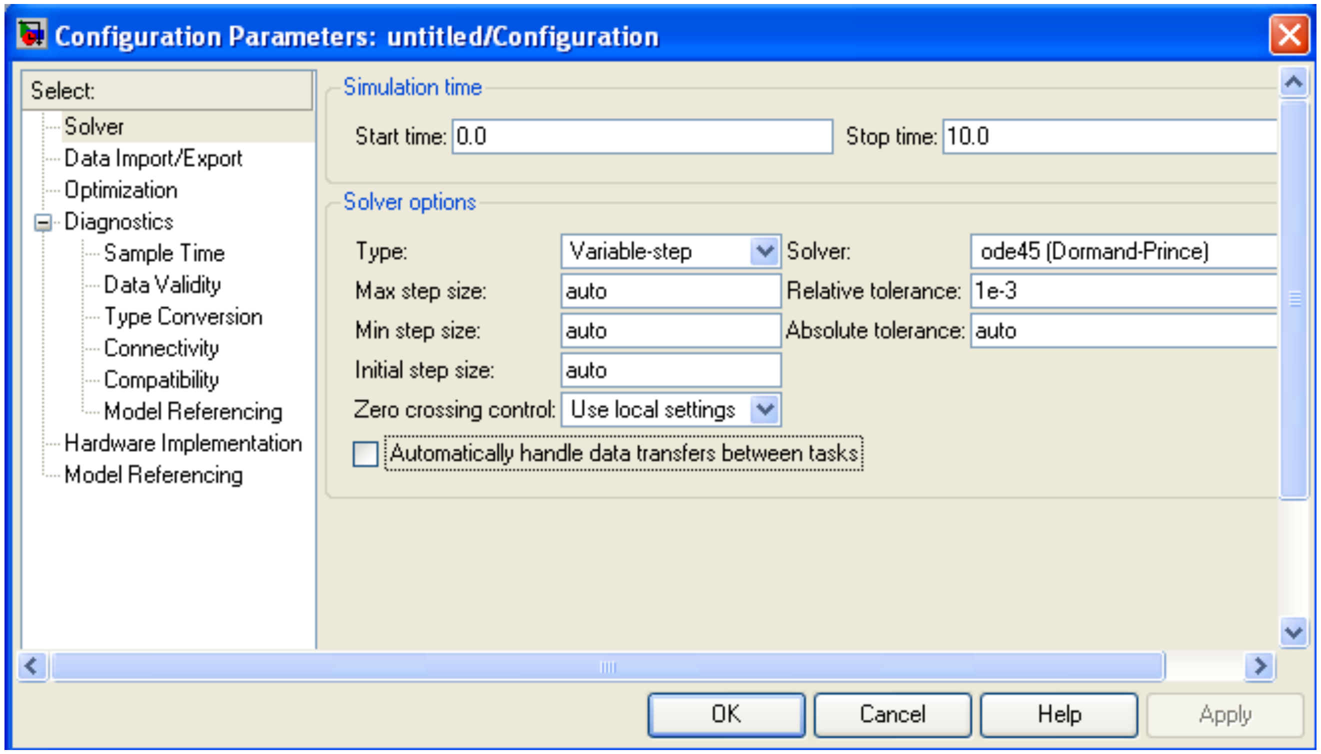
Initial step size: auto

Zero crossing control: Use local settings

☐ Automatically handle data transfers between tasks

Controlling the Simulation - Solver

Simulink uses numerical integration to solve dynamic system equations. **Solver** is the engine used for numerical integration. Key parameters are the Start and Stop times, and the Solver Type and methods.


Configuration Parameters: untitled/Configuration

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Max step size: auto Relative tolerance: 1e-3

Min step size: auto Absolute tolerance: auto

Initial step size: auto

Zero crossing control: Use local settings

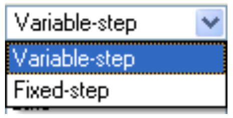
☐ Automatically handle data transfers between tasks

Solver Parameters

The Start and Stop times set the start and stop times for the simulation.

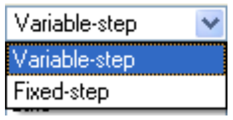
Solver Type sets the method of numerical integration as variable step or fixed step. Variable step continuously adjusts the integration step size so as to speed up the simulation time. I.E., variable step size reduces the step size when a model's states are changing rapidly to maintain accuracy and increases the step size when the system's states are changing slowly in order to avoid taking unnecessary steps.

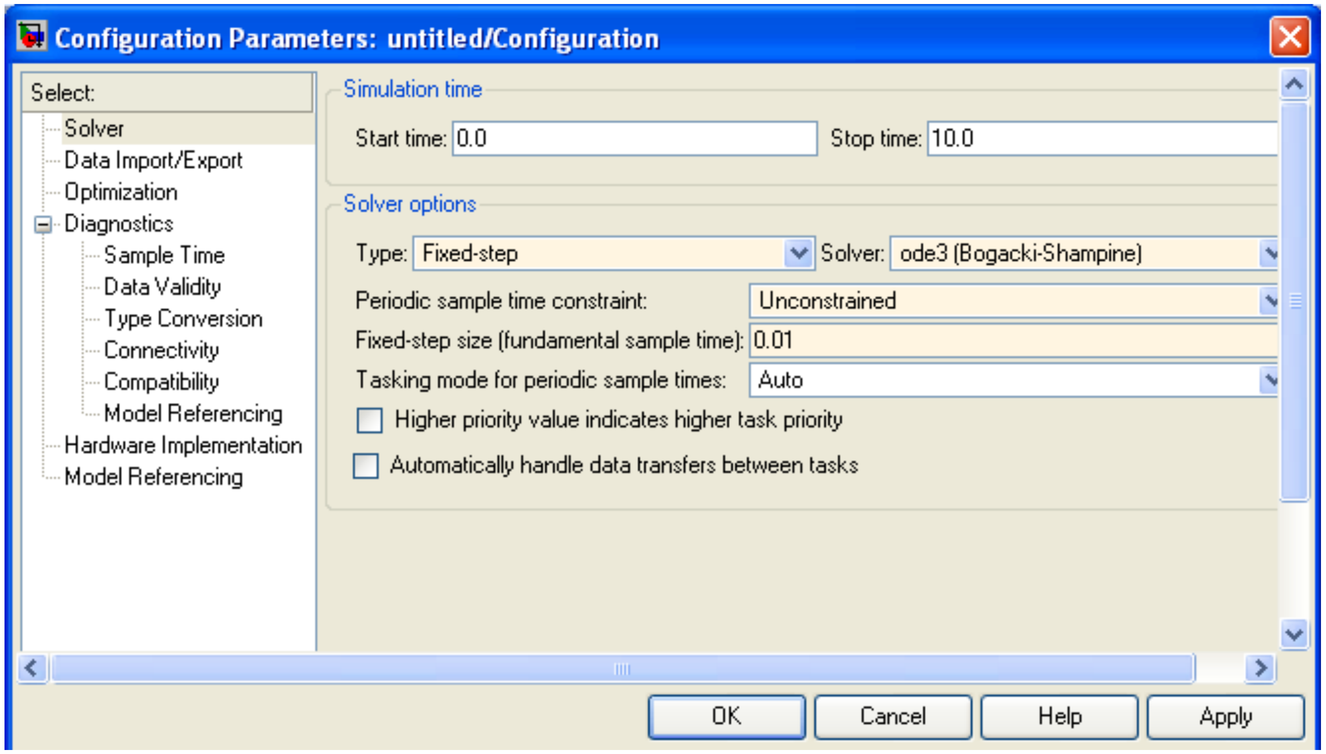
Fixed step uses the same fixed step size throughout the simulation. When choosing fixed step it is necessary to set the step size.



Setting Fixed Step Size

The following example illustrates setting the fixed step size to a value of 0.01 seconds.




Configuration Parameters: untitled/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Fixed-step Solver: ode3 (Bogacki-Shampine)

Periodic sample time constraint: Unconstrained

Fixed-step size (fundamental sample time): 0.01

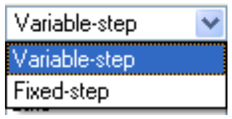
Tasking mode for periodic sample times: Auto

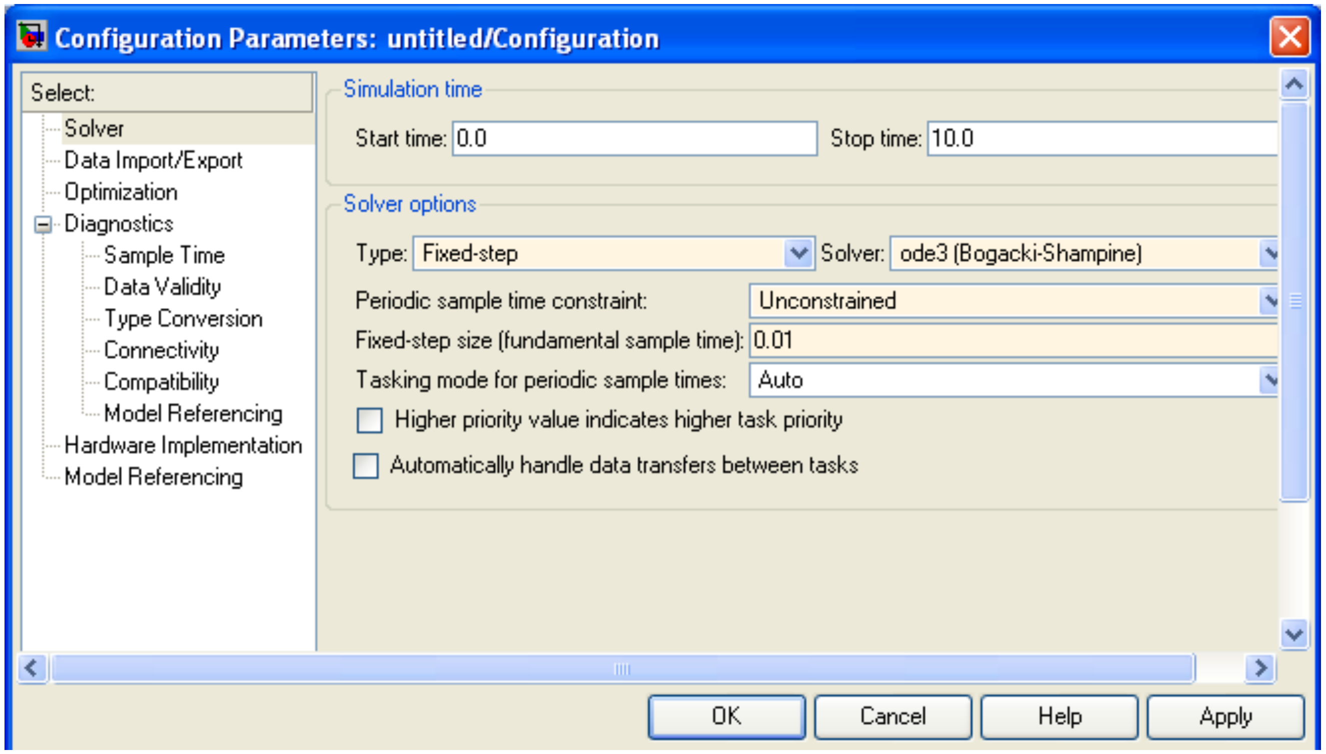
☐ Higher priority value indicates higher task priority

☐ Automatically handle data transfers between tasks

Setting Fixed Step Size

The following example illustrates setting the fixed step size to a value of 0.01 seconds.





Configuration Parameters: untitled/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Fixed-step Solver: ode3 (Bogacki-Shampine)

Periodic sample time constraint: Unconstrained

Fixed-step size (fundamental sample time): 0.01

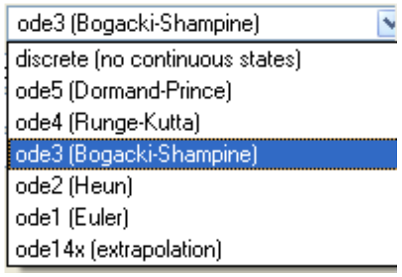
Tasking mode for periodic sample times: Auto

☐ Higher priority value indicates higher task priority

☐ Automatically handle data transfers between tasks

Setting the Integration Type

The list of integration type solvers are shown below. For more information on fixed and variable step methods and integration types consult the MATLAB Simulink tutorial.



ode3 (Bogacki-Shampine)

discrete (no continuous states)
ode5 (Dormand-Prince)
ode4 (Runge-Kutta)
ode3 (Bogacki-Shampine)
ode2 (Heun)
ode1 (Euler)
ode14x (extrapolation)

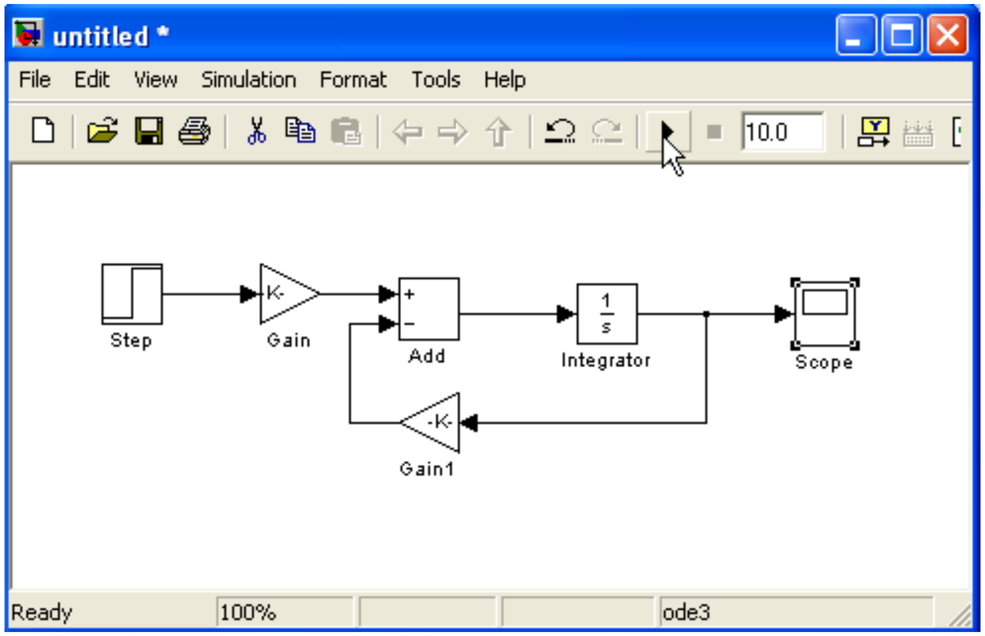
Running the Simulation

To run the **simulation** we 1st need to enter the values of R and C. Note we could have entered these directly in the gain blocks but we chose to enter these from the work space. To do this in the work space we type

>>R = 1; C = 1;

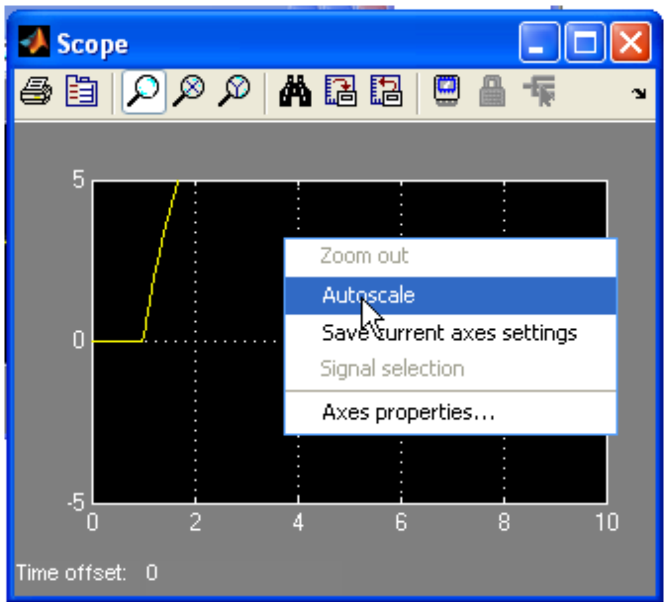
This will fix the gain block's as K = 1/(R*C)

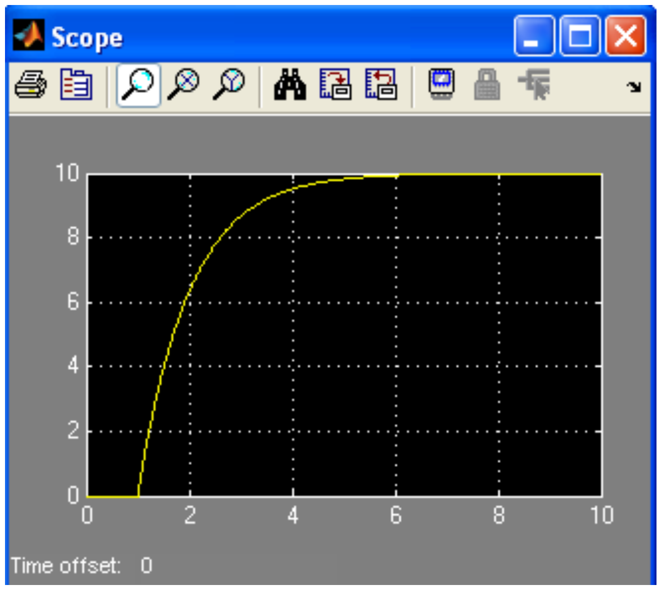
Now **select** and **click** the run button to run the simulation



Viewing the Simulation with the Scope

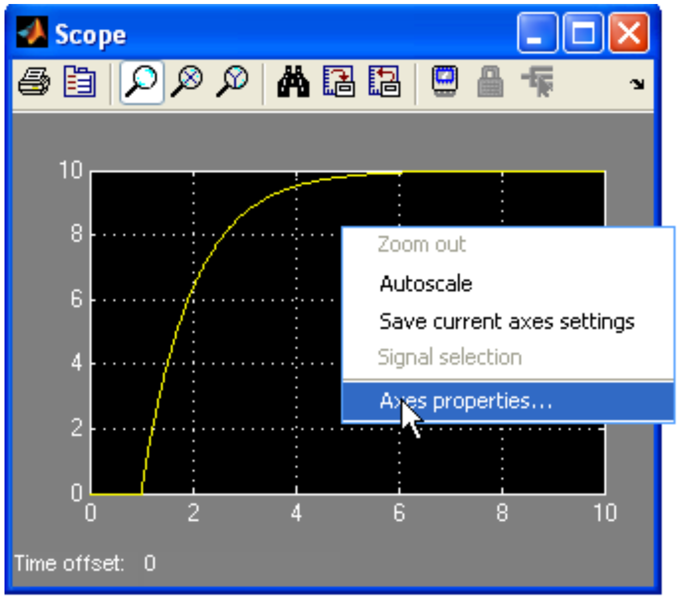
To view the simulation results double click on the Scope block to get. To get a better view left click and select Autoscale

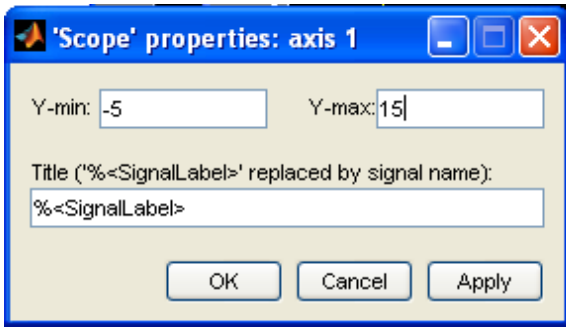




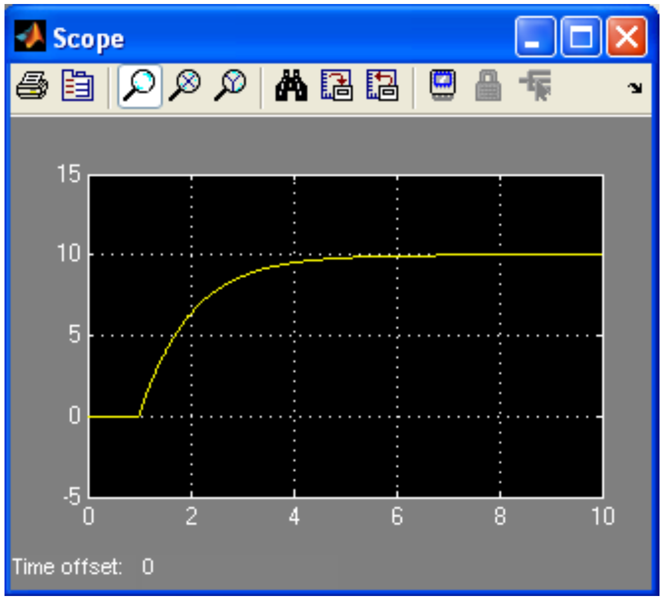
Setting the Scope Axis Properties

We can adjust the Scope axis properties to set the axis limits:



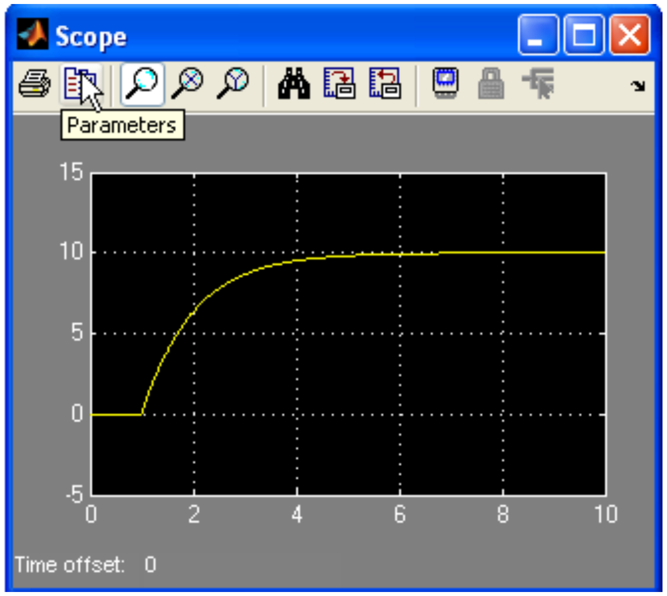

'Scope' properties: axis 1

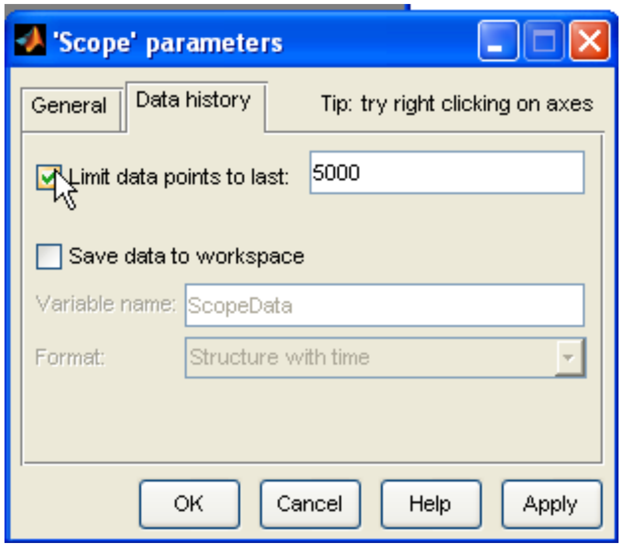
Y-min: -5
Y-max: 15
Title ('%<SignalLabel>' replaced by signal name): %<SignalLabel>



Setting the Scope Parameters

The default settings for the Scope is a Data History of 5000 data samples. By selecting Parameters →an un-clicking the Limit data points to last 5000 we can display an unlimited number of simulation samples. Especially useful for long simulations or simulations with very small time steps.





[✓] Limit data points to last: 5000

[ ] Save data to workspace

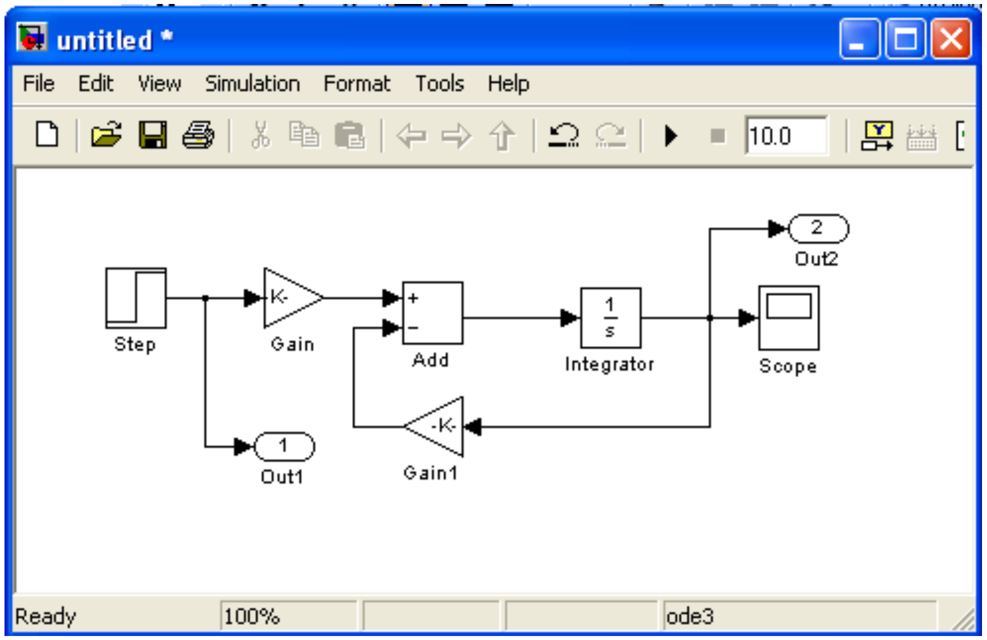
Variable name: ScopeData

Format: Structure with time

Saving Data to the Work Space

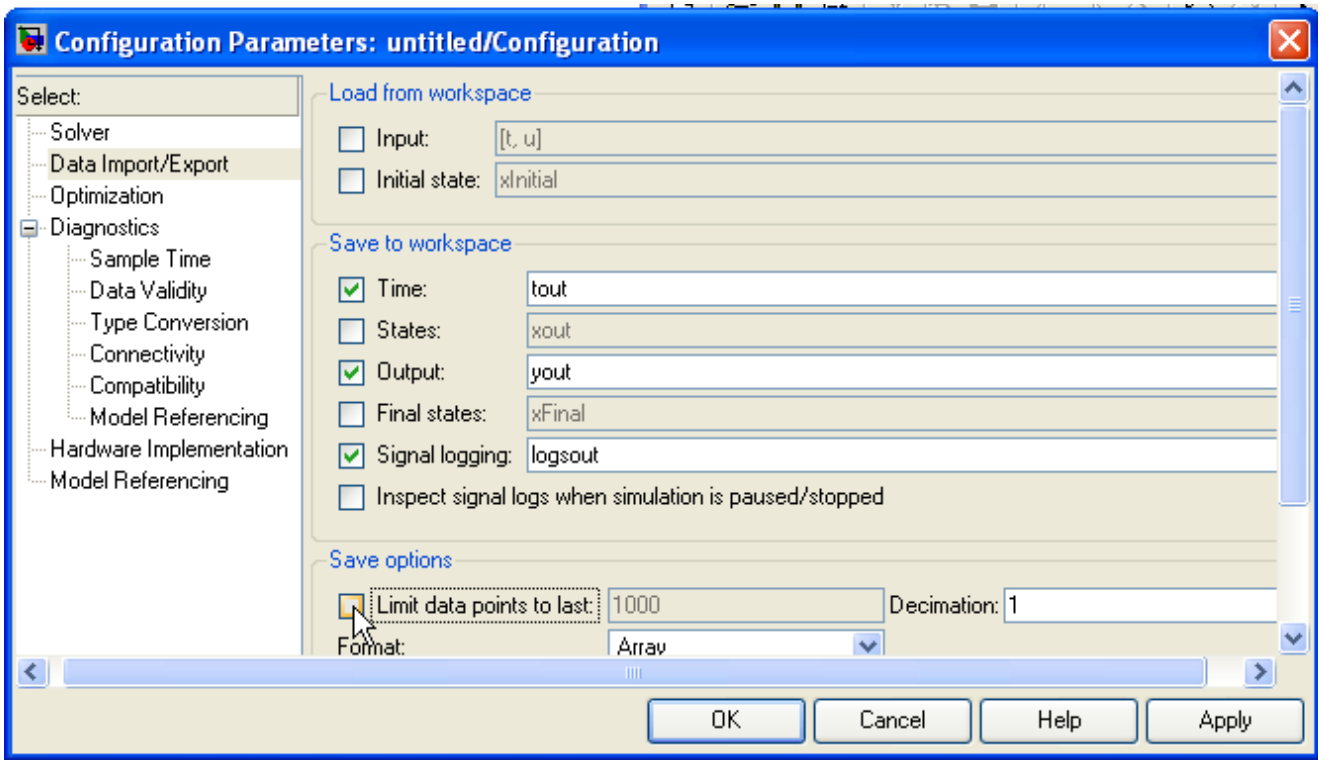
Often we wish to save data to the work space and use the MATLAB plot commands to display the data. To do this we can use the Sink Out1 block to capture the data we wish to display.

In this example we will connect Out blocks to both the input and the output as shown below:



Where does the Output Data get Saved

To see where the output data gets saved we open Solver and select Data Import/Export and unclick the Limit data points to last box. As shown time is stored as tout, and the outputs as yout.


Configuration Parameters: untitled/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing

Load from workspace

☐ Input:	[t, u]
☐ Initial state:	xInitial

Save to workspace

☑ Time:	tout
☐ States:	xout
☑ Output:	yout
☐ Final states:	xFinal
☑ Signal logging:	logsout

☐ Inspect signal logs when simulation is paused/stopped

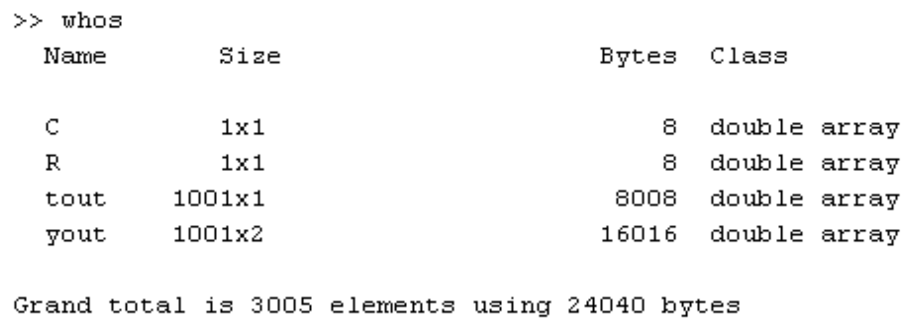
Save options

☐ Limit data points to last: 1000 Decimation: 1

Format: Array

Where does the Output Data get Saved

Now run the simulation and go to the workspace and type **whos**



```
>> whos
  Name        Size                    Bytes  Class

  C           1x1                         8  double array
  R           1x1                         8  double array
  tout     1001x1                      8008  double array
  yout     1001x2                     16016  double array

Grand total is 3005 elements using 24040 bytes
```

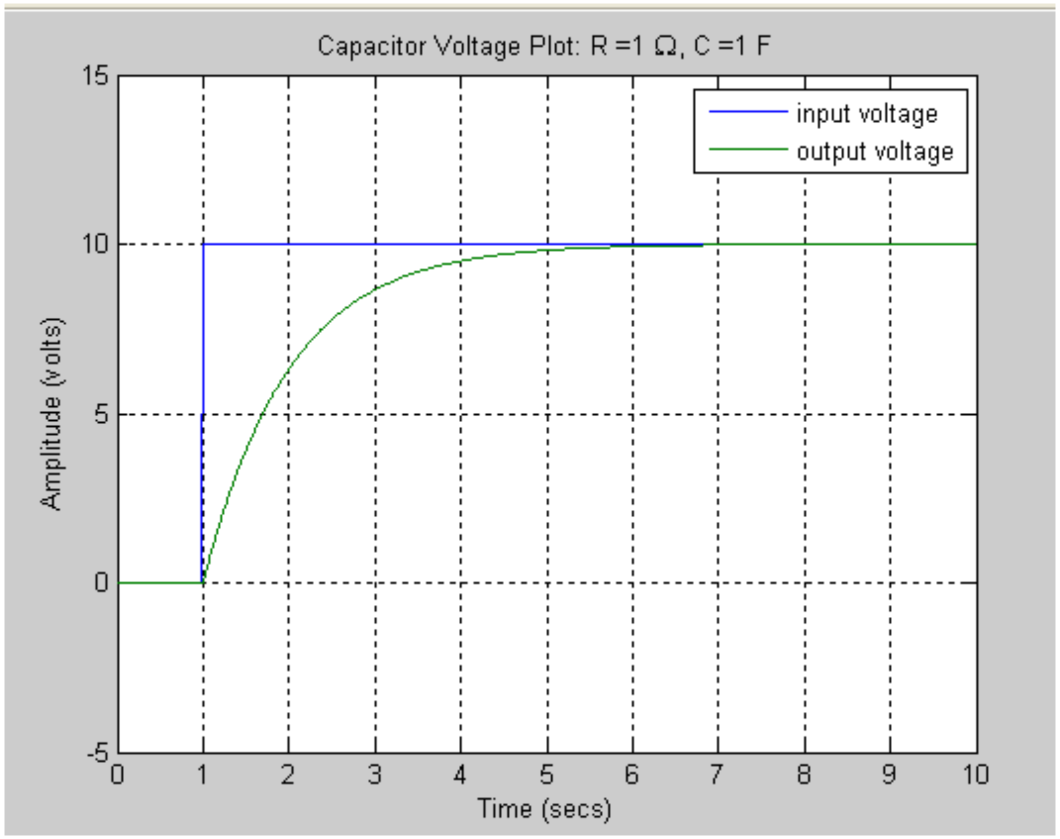
Note that yout is a 1001 by 2 vector, that means that yout(:, 1)=input samples, and yout(:,2)=output samples.

To plot the data execute the M-File script shown on the next slide.

M-File Script to Plot Simulink Data

```
' M-File script to plot simulation data '
plot(tout,yout(:,1),tout,yout(:,2)); % values to plot
xlabel('Time (secs)'); grid;
ylabel('Amplitude (volts)');
st1 = 'Capacitor Voltage Plot: R = '
st2 = num2str(R);    % convert R value to string
st3 = ' \Omega';     % Greek symbol for Ohm
st4 = ', C = ';
st5 = num2str(C);
st6 = ' F';
stitle = strcat(st1,st2,st3,st4,st5,st6);  % plot title
title(stitle)
legend('input voltage','output voltage')
axis([0 10 -5 15]); % set axis for voltage range of -5 to 15
```

Capacitor Input/Output Plot



The SIM Function

Suppose we wish to see how the capacitor voltage changes for several different values of C. To do this we will use the sim command.

To use the sim command we must 1st save the model. To do this select the File→ Save As menu and save the file as capacitor_charging_model. Make sure you save the model to a folder in your MATLAB path.

Next execute the M-File script on the next slide. The for loop with the sim command runs the model with a different capacitor value each time the sim command is executed. The output data is saved as vc(k,:)

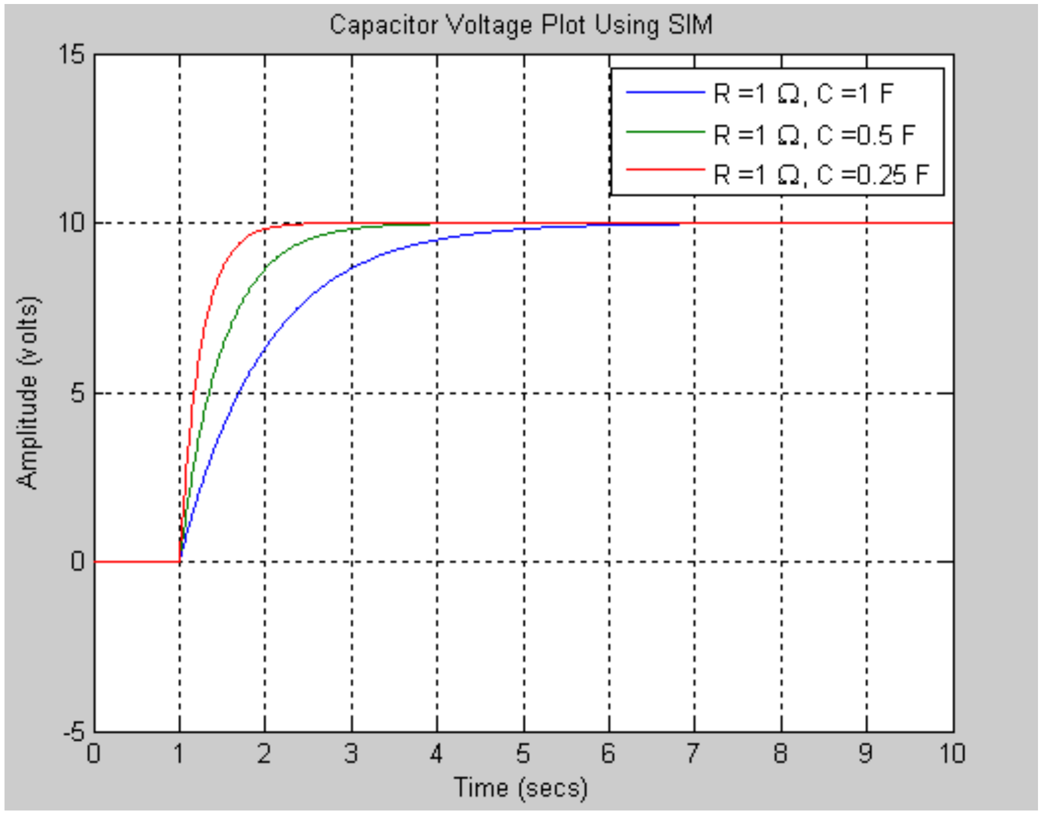
The plot command is used to plot the data for the three different capacitor values. Note the use of string's to build the plot legend.

M-Code to Run and Plot the Simulation data

```
' M-File script to plot simulation data '
R = 1; % set R value;
CAP = [1 1/2 1/4]; % set simulation values for C
for k = 1:3
    C = CAP(k); % capacitor value for simulation
    sim('capacitor_charging_model'); % run simulation
    vc(k,:) = yout(:,2); % save capacitor voltage data
end
plot(tout,vc(1,:),tout,vc(2,:),tout,vc(3,:)); % values to plot
xlabel('Time (secs)'); grid;
ylabel('Amplitude (volts)');
st1 = 'R = '
st2 = num2str(R);    % convert R value to string
st3 = ' \Omega';     % Greek symbol for Ohm
st4 = ', C = ';
st5 = num2str(CAP(1));
st6 = ' F';
sl1 = strcat(st1,st2,st3,st4,st5,st6);  % legend 1
st5 = num2str(CAP(2));
```

M-Code to Run and Plot the Simulation data

```
sl2 = strcat(st1,st2,st3,st4,st5,st6);  % legend 2
st5 = num2str(CAP(3));
sl3 = strcat(st1,st2,st3,st4,st5,st6);  % legend 3
title('Capacitor Voltage Plot Using SIM')
legend(sl1,sl2,sl3); % plot legend
axis([0 10 -5 15]); % set axis for voltage range of -5 to 15
```



Modeling a Series RLC Circuit

For a 2nd example lets model the voltage across the capacitor of a series RLC circuit. The differential equation for the capacitor voltage is :

$$\frac{d^2 v_C(t)}{dt^2} = -\frac{R}{L}\frac{dv_C(t)}{dt} - \frac{1}{LC}v_C(t) + \frac{1}{LC}u(t)$$

$u(t)$ = The unit step function

To model this system we let

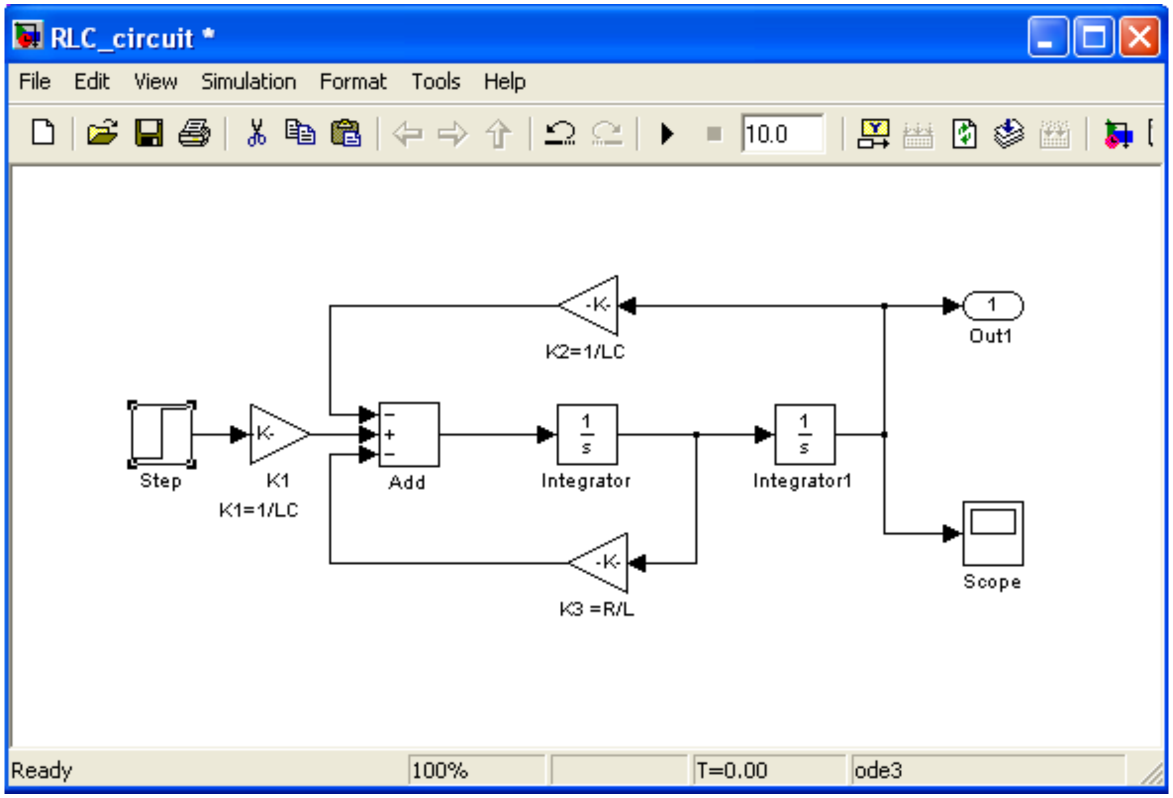
$$z_1 = v_C$$

$$\frac{dz_1}{dt} = z_2 = \frac{dv_C}{dt}$$

$$\frac{dz_2}{dt} = -\frac{R}{L}z_2 - \frac{1}{LC}z_1 + \frac{1}{LC}u(t)$$

The completed simulation diagram is shown on the next slide

RLC Circuit Simulink Model



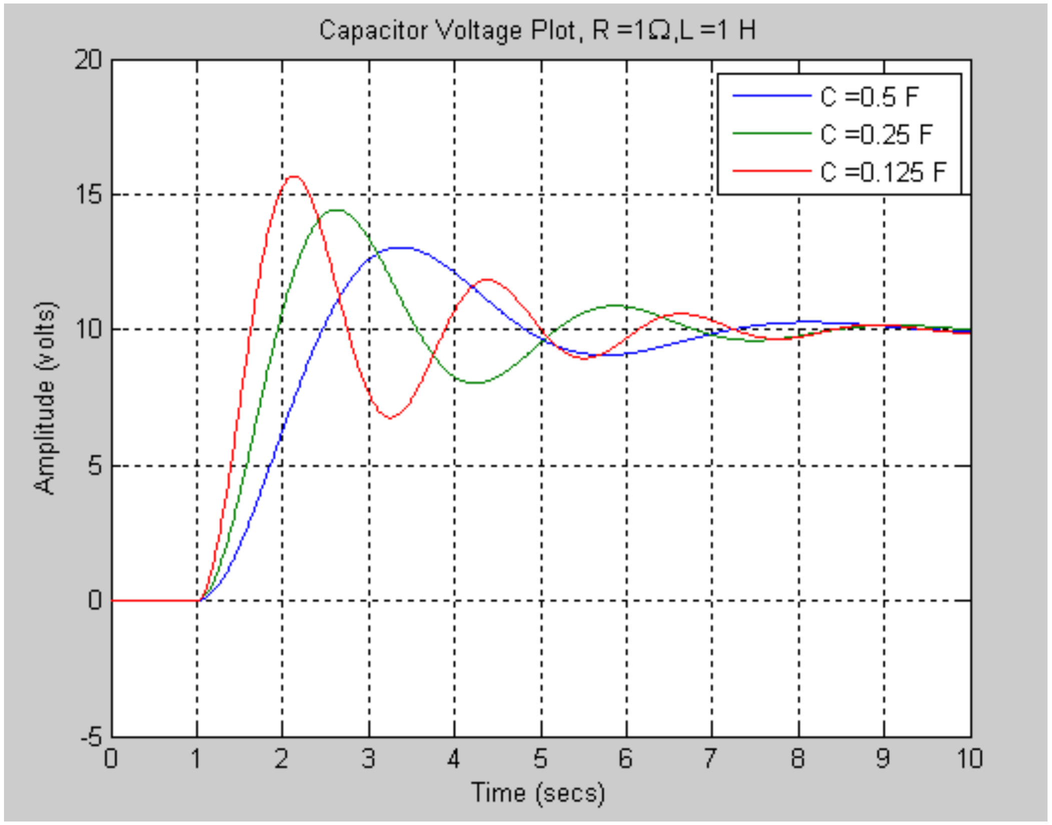
SIM script to run RLC Circuit Model

```
' M-File script to plot simulation data '
R = 1; % set R value;
L = 1; % set L
CAP = [1/2 1/4 1/8]; % set simulation values for C
for k = 1:3
    C = CAP(k); % capacitor value for simulation
    sim('RLC_circuit'); % run simulation
    vc(k,:) = yout; % save capacitor voltage data
end
plot(tout,vc(1,:),tout,vc(2,:),tout,vc(3,:)); % values to plot
xlabel('Time (secs)'); grid;
ylabel('Amplitude (volts)');
st1 = 'C = ';
st2 = num2str(CAP(1));
st3 = ' F';
sl1 = strcat(st1,st2,st3);  % legend 1
st2 = num2str(CAP(2));
sl2 = strcat(st1,st2,st3);  % legend 2
st2 = num2str(CAP(3));
sl3 = strcat(st1,st2,st3);  % legend 3
```

SIM script to run RLC Circuit Model

```
st1 ='Capacitor Voltage Plot, R = ';
st2 = num2str(R); st3 = '\Omega';
st4 = ',L = '; st5 = num2str(L);
st6 = ' H';
stitle = strcat(st1,st2,st3,st4,st5,st6);
title(stitle)
legend(sl1,sl2,sl3); % plot legend
axis([0 10 -5 20]); % set axis for voltage range of -5 to 20
```

RLC Circuit Model Plot



Modifying the Series RLC Circuit

For a 3rd example lets modify the series RLC circuit model to also plot the voltage across the inductor and the resistor. To do this we note that:

$$v_R(t) = Ri(t) = RC\frac{dv_C(t)}{dt}$$

From KVL we have

$$v_L(t) = u(t) - v_R(t) - v_C(t)$$

But in terms of the variables z_1 and z_2 we have

$$\frac{dv_C}{dt} = \frac{dz_1}{dt} = z_2$$

Making this substitution the voltage drops across the resistor becomes:

Modifying the Series RLC Circuit (2)

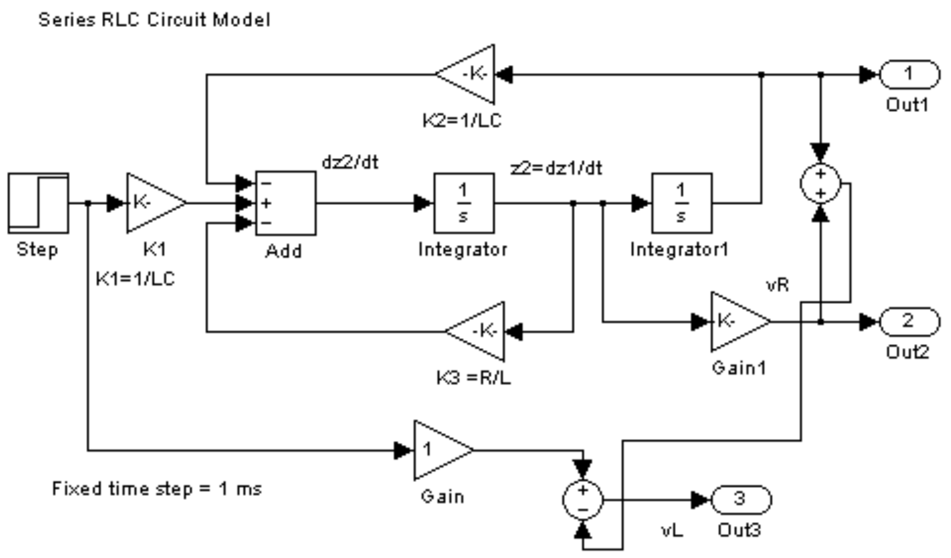
Resistor Voltage Drop :

$$v_R(t) = Ri(t) = RC\frac{dv_C(t)}{dt} = RCz_2$$

Inductor Voltage Drop :

$$v_L(t) = u(t) - v_R(t) - v_C(t)$$

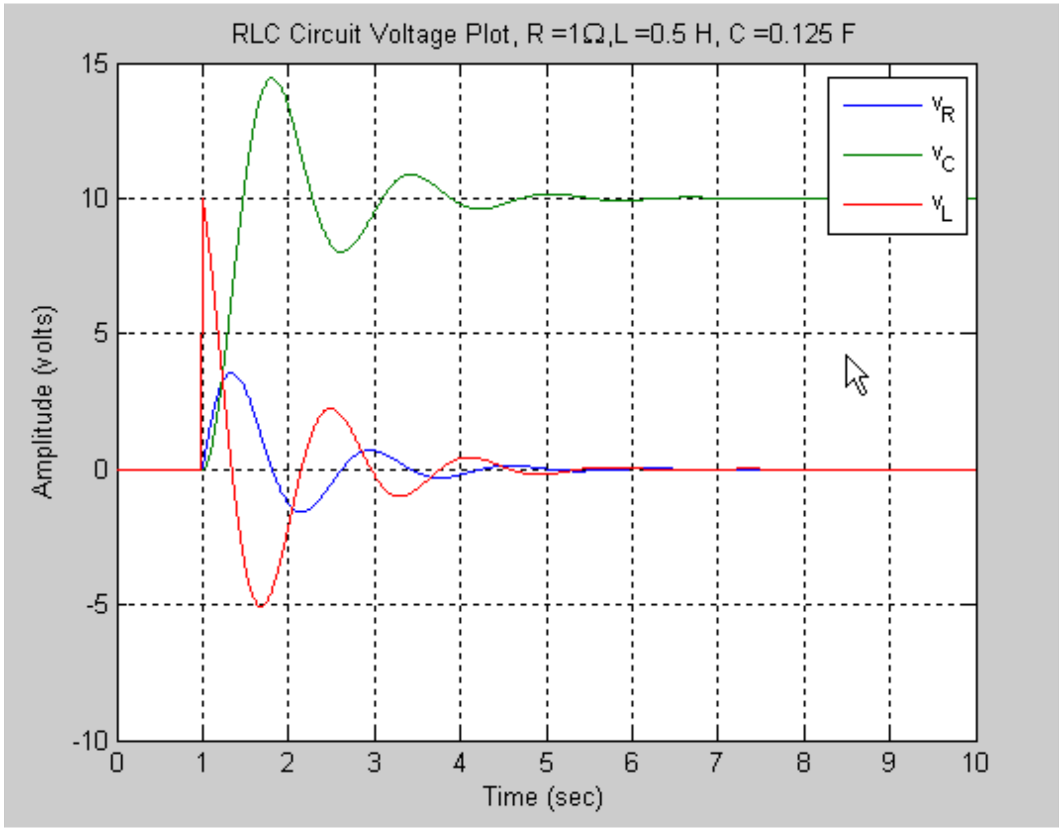
Simulink Model :



Running the Simulation

```
' MATLAB script for RLC simulation '
R = 1; C = 1/8; L = 1/2; % set circuit parameters
sim('RLC_circuit_Mod'); % run simulation
vC = yout(:,1); % capacitor voltage drop
vR = yout(:,2); % resistor voltage drop
vL = yout(:,3); % inductor voltage drop
plot(tout,vR,tout,vC,tout,vL); % plot individual voltage drops
legend('v_R','v_C','v_L'); % add plot legend
grid; % add grid to plot
xlabel('Time (sec)');  % add x and y labels
ylabel('Amplitude (volts)');
% Build plot title
st1 ='RLC Circuit Voltage Plot, R = ';
st2 = num2str(R); st3 = '\Omega';
st4 = ',L = '; st5 = num2str(L);
st6 = ' H'; st7 = ', C = ';
st8 = num2str(C); st9 = ' F';
stitle = strcat(st1,st2,st3,st4,st5,st6,st7,st8,st9);
title(stitle)
```

Simulation Results


RLC Circuit Voltage Plot, R =1Ω,L =0.5 H, C =0.125 F
Amplitude (volts)
Time (sec)
v_R
v_C
v_L