

Using the Latest Features of Simulink 2

>> 1997

>> **matlab** conference

Rick Spada

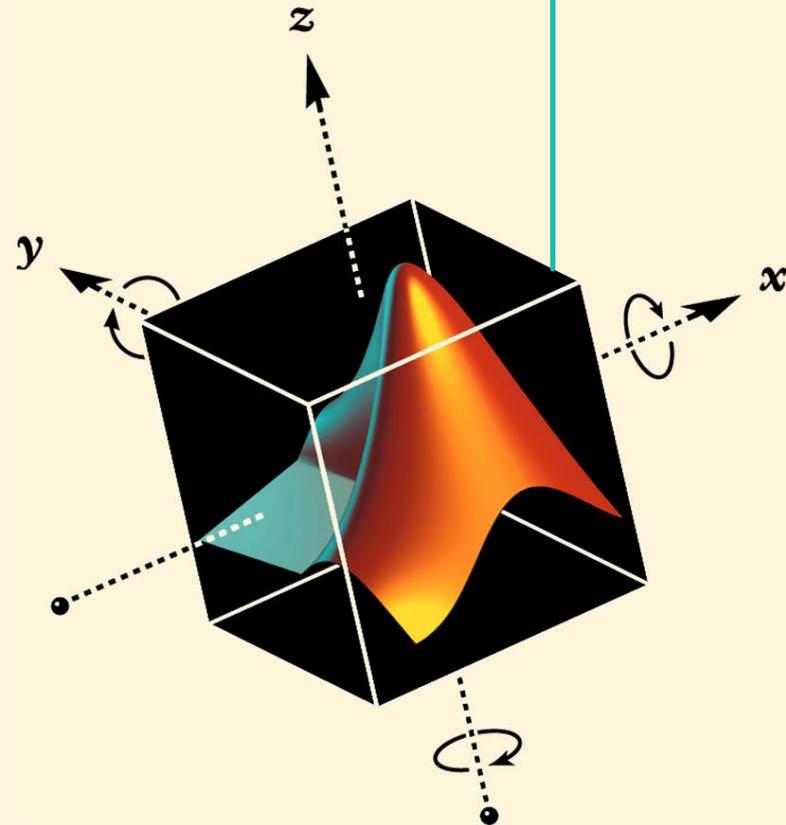
ricks@mathworks.com

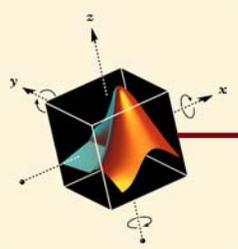
The MathWorks Inc.

24 Prime Park Way

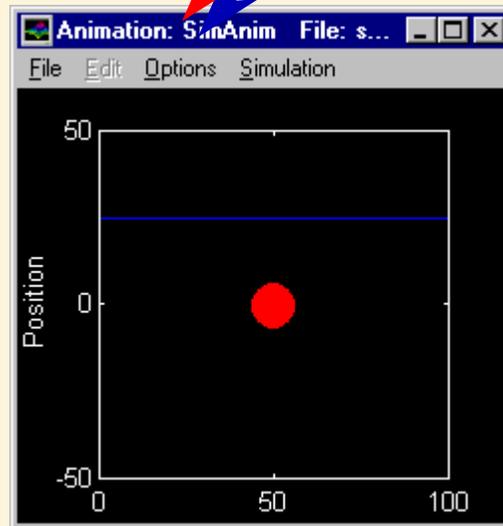
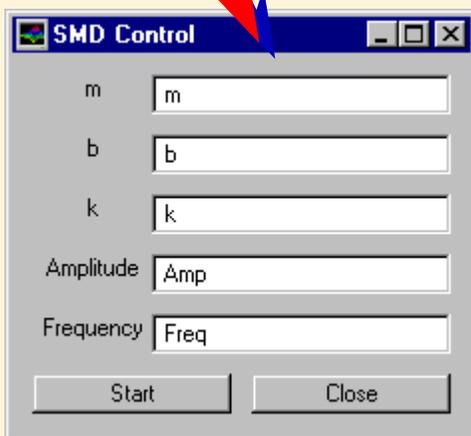
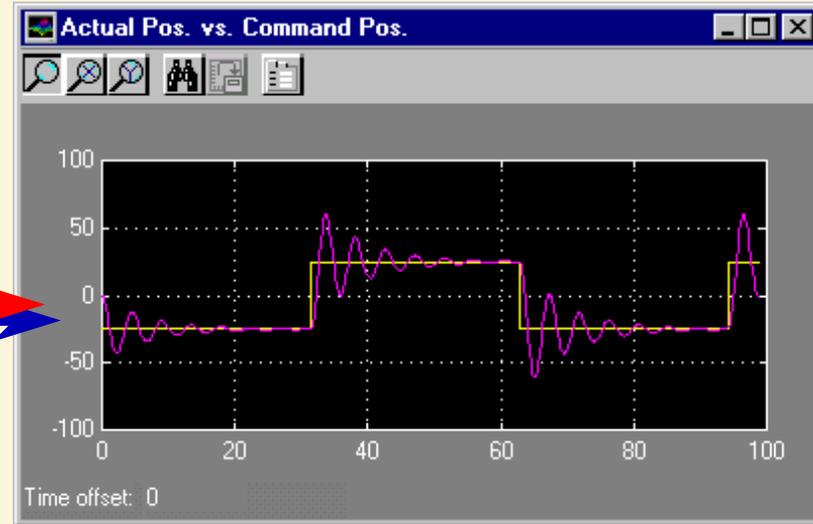
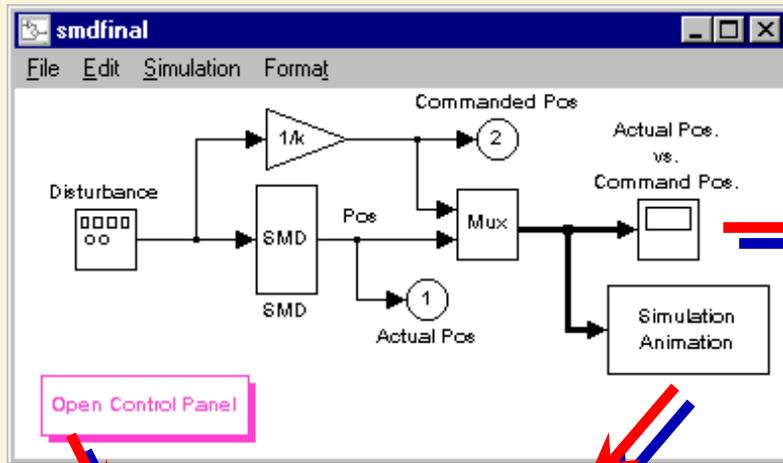
Natick, MA 01760-1500

USA

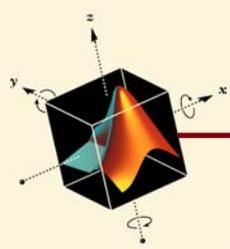




What you are going to learn about...

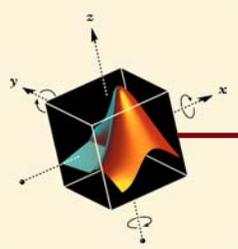


- ◆ Building a model
- ◆ Block libraries
- ◆ New blocks
- ◆ Running a simulation



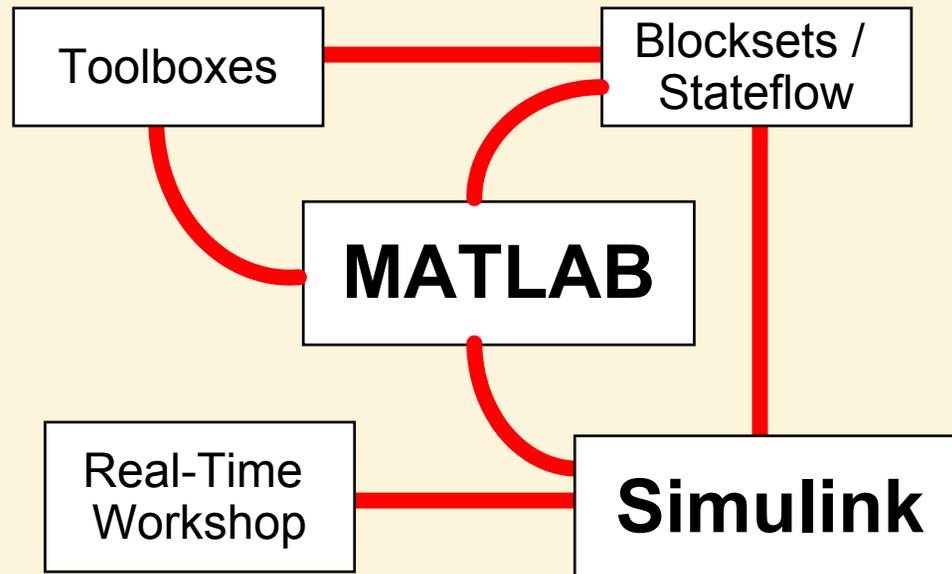
Course Outline

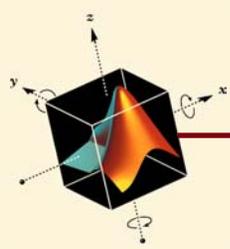
- ◆ Introduction
- ◆ Review of Simulink basics
- ◆ Building a model
- ◆ Using Block Libraries
- ◆ New Blocks
- ◆ Running a Simulation



What is Simulink?

Simulink is an interactive, block-diagram-based tool for modeling and analyzing dynamic systems. It is tightly coupled with MATLAB and supported by Blocksets and Extensions.



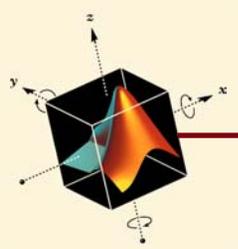


What is Simulink?

A simulation tool in MATLAB's overall Technical Computing Environment.

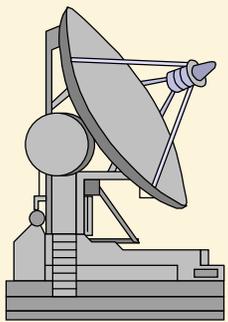
Among other things, this means:

- ◆ Block diagram editing
- ◆ Nonlinear simulation
- ◆ Continuous and discrete simulation
- ◆ Hybrid simulation
- ◆ Asynchronous (non-uniform sampling) simulation
- ◆ Integration with MATLAB, Extensions, Blocksets & Toolboxes



What can you model with Simulink?

Just about anything you can model mathematically

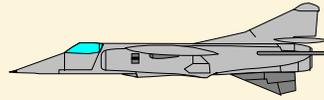


Communications and satellite systems

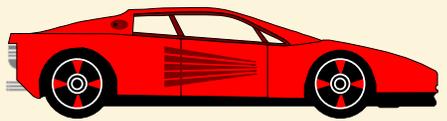
Ship systems



Aircraft and spacecraft dynamics and systems



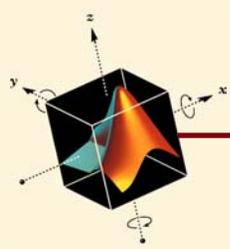
Biological systems



Automotive systems

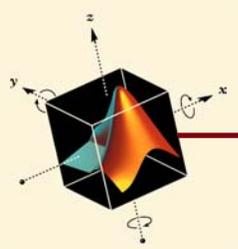


Monetary systems

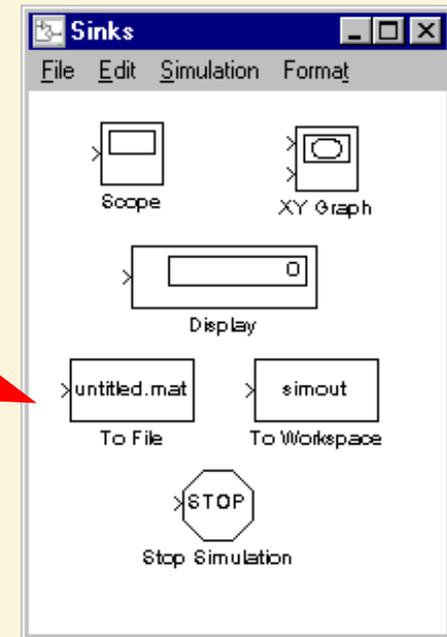
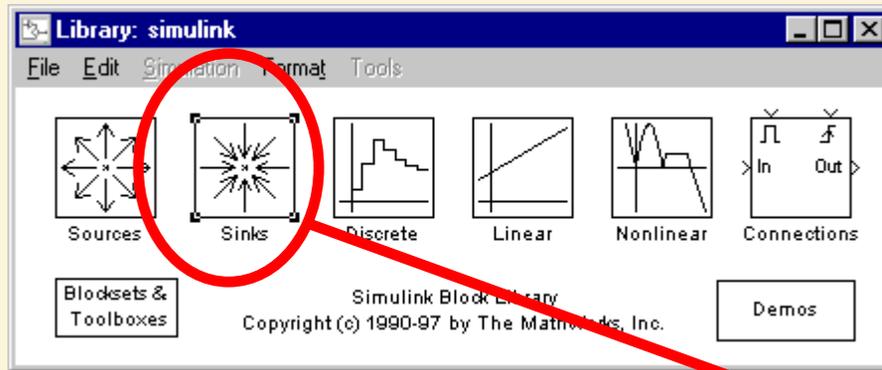


Simulink Basics

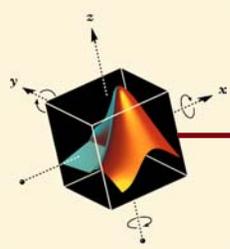
- ◆ It has its own special windows for block diagrams
- ◆ It works by setting up a dialogue between the system and the solver
- ◆ It shares MATLAB's workspace



The Simulink Block Library

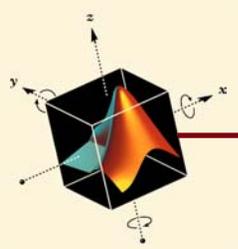


>> simulink

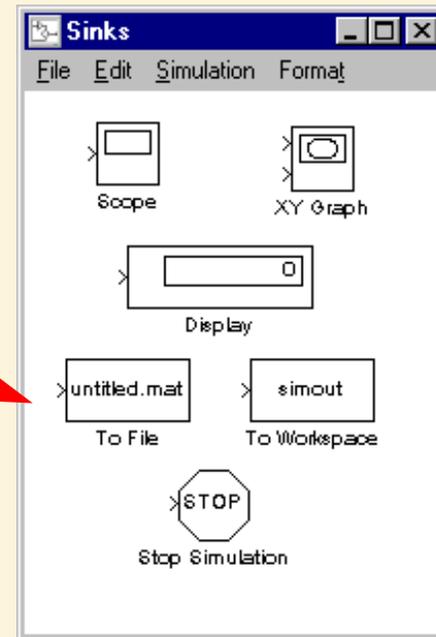
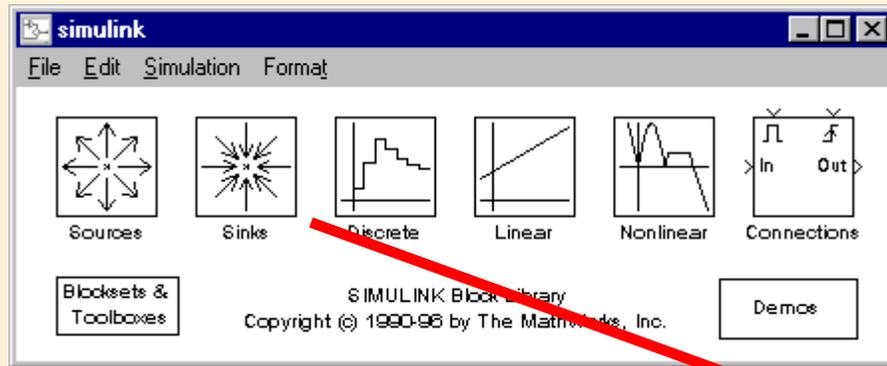


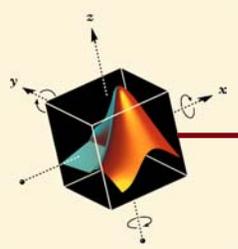
Building a Model

- ◆ **Starting Simulink**
- ◆ **Editing models**
 - ◆ Adding and connecting blocks
 - ◆ Annotations
 - ◆ Signal labels
 - ◆ Saving (MDL file format)
 - ◆ Undo
 - ◆ Printing
 - ◆ Browser
 - ◆ Help
 - ◆ Tips for building models

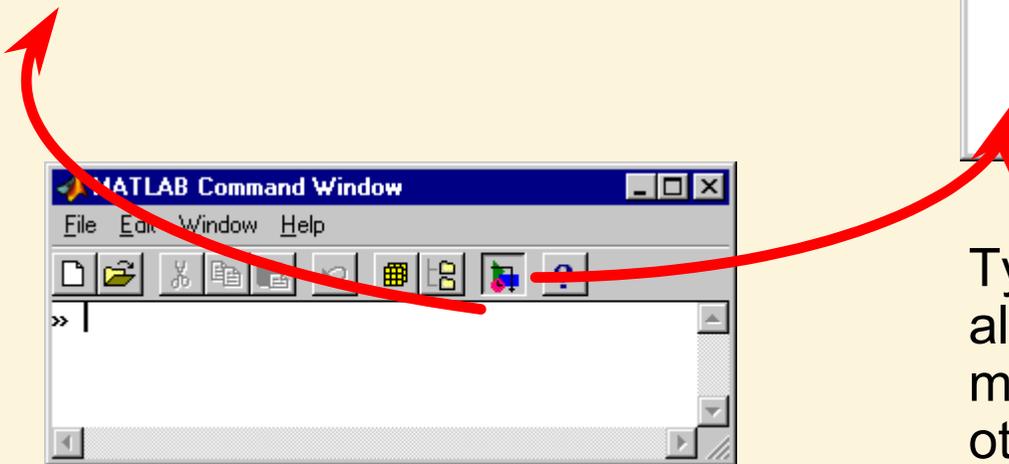
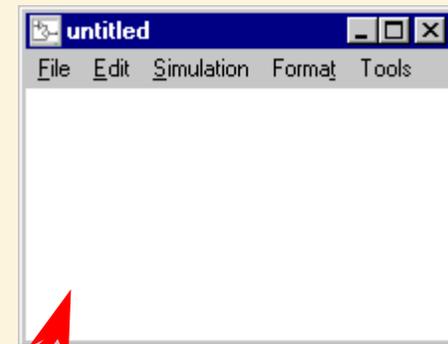
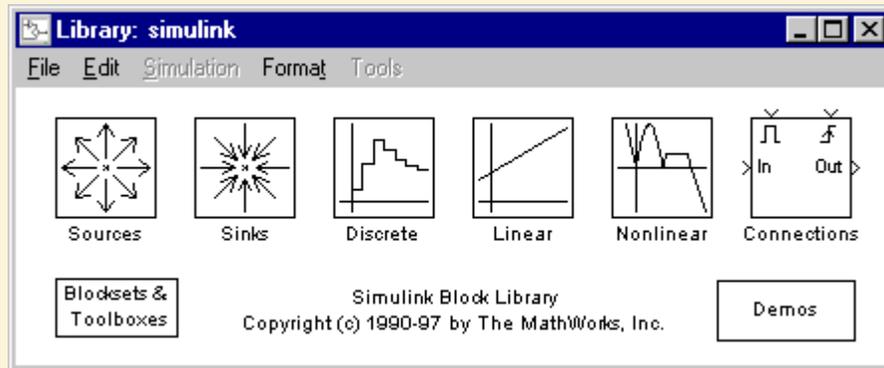


Starting Simulink



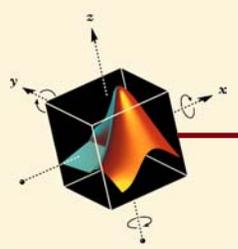


Open a new model window

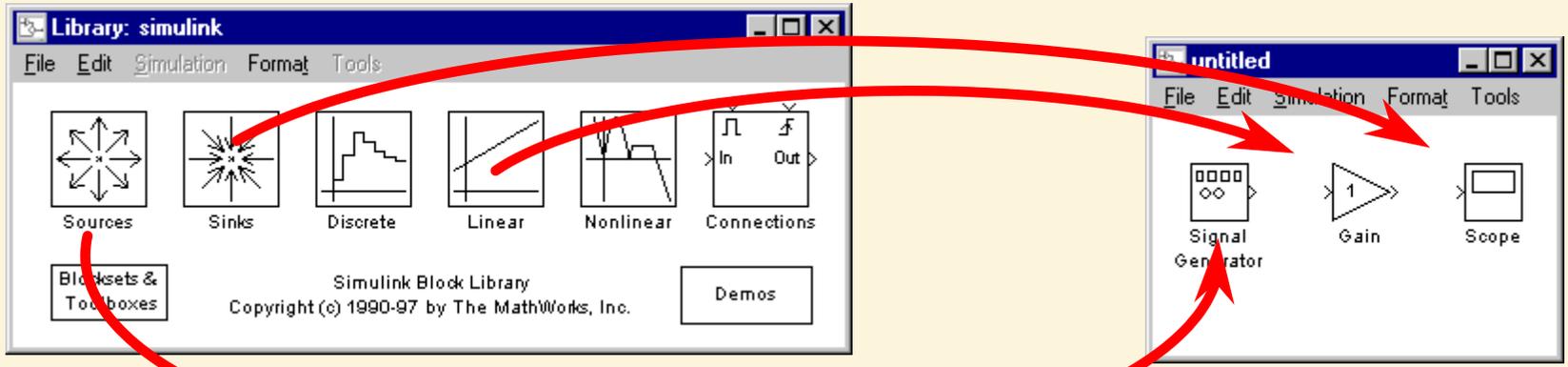


Typing `simulink` will also open a new model window if no other model is open

» `simulink`

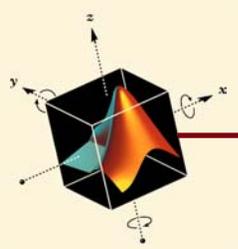


Adding Blocks



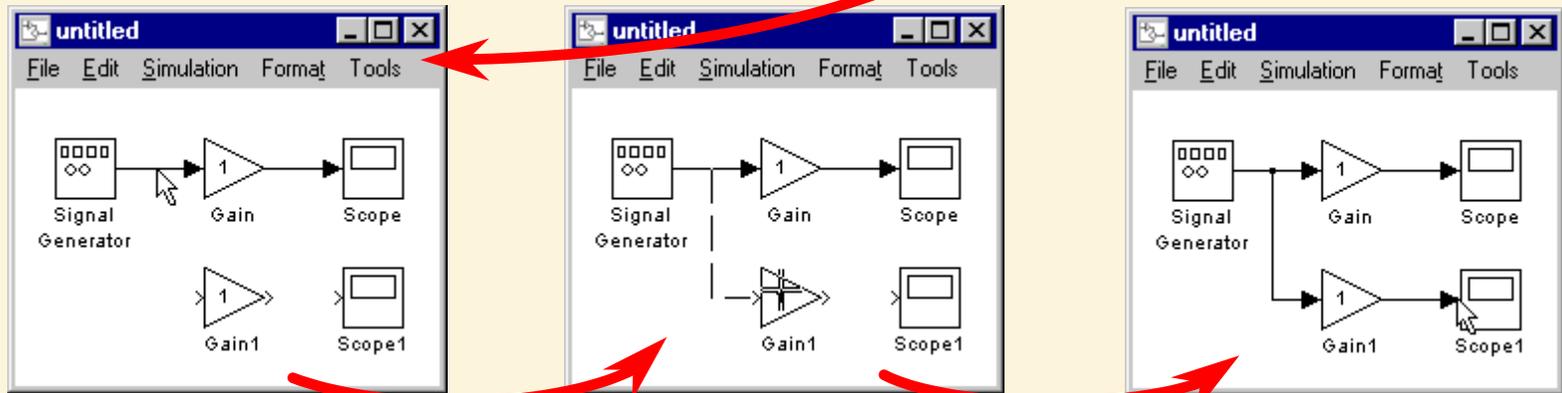
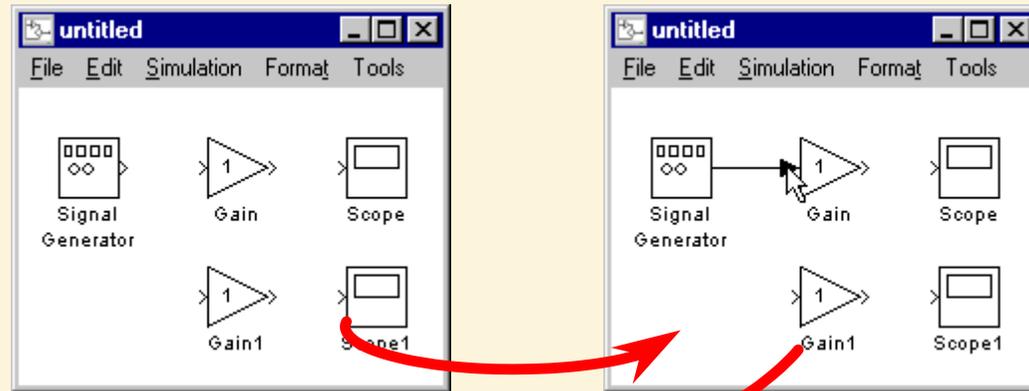
Pull the required blocks into a new Simulink block diagram window.

» twogain



Connecting Blocks

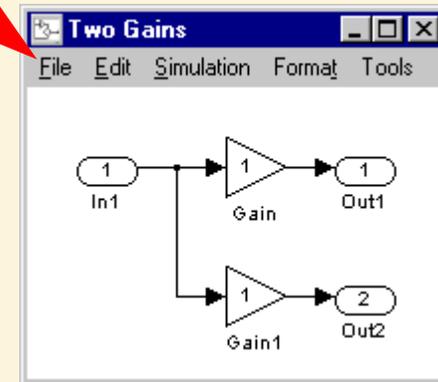
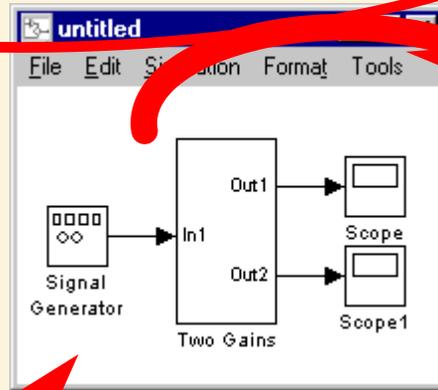
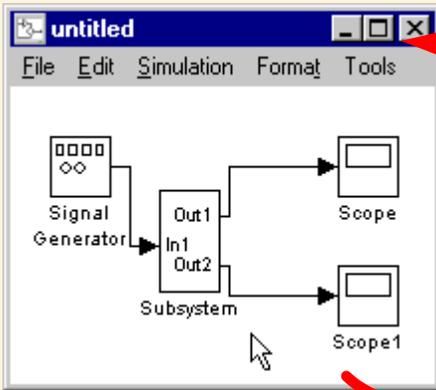
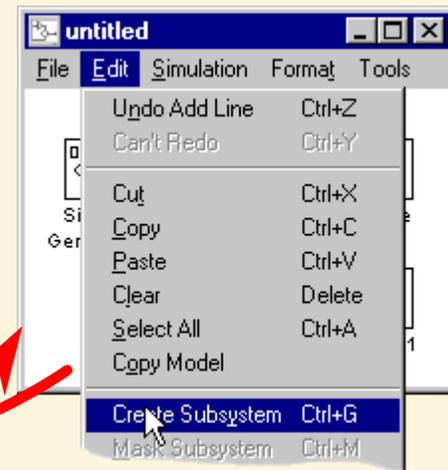
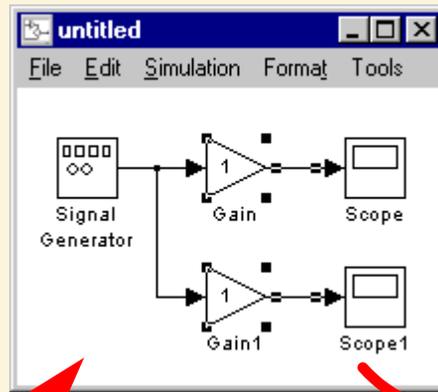
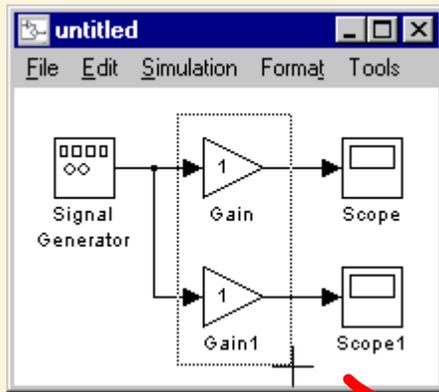
Left mouse drag from port to add a line.



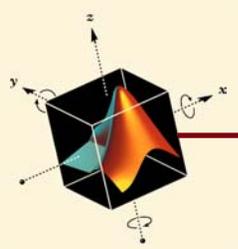
Right mouse drag to add a branch line.

» twogain

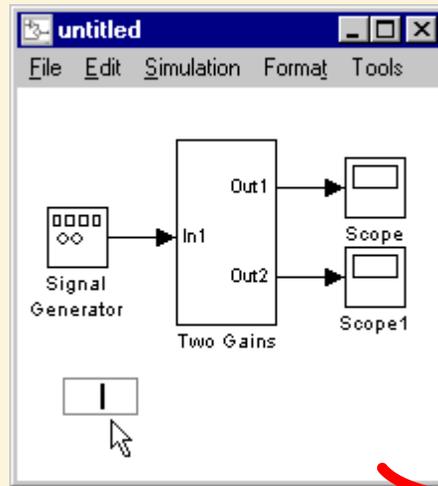
Creating Subsystems



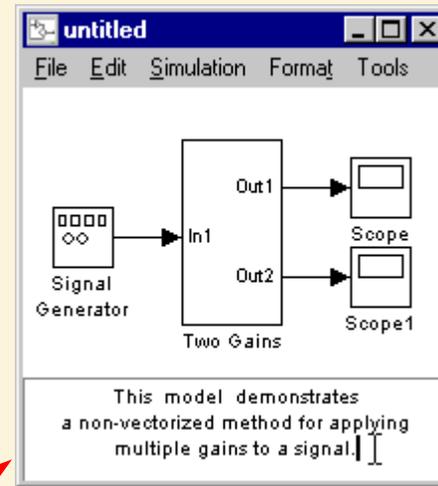
» twogain



Annotations

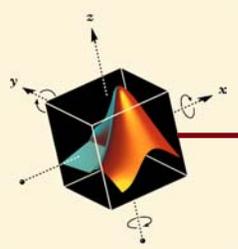


Double click on background

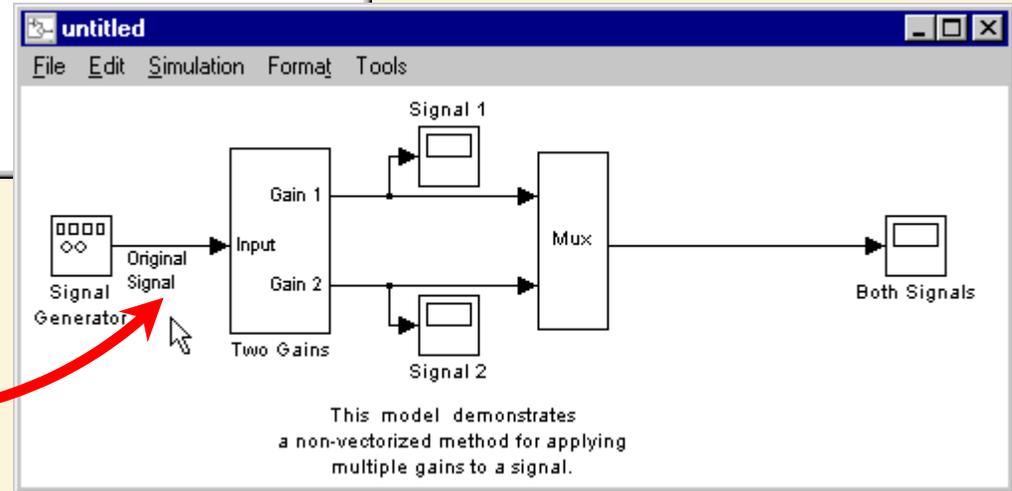
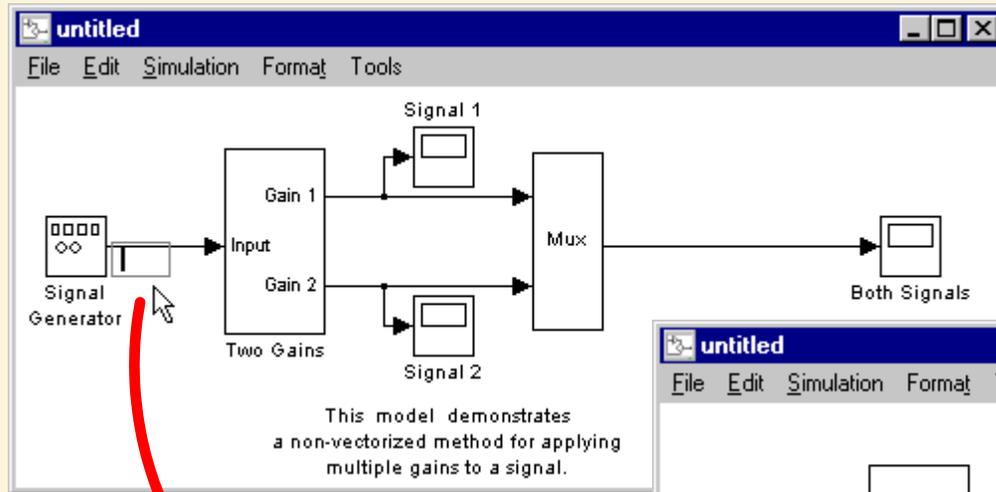


Type in the text

» twogain



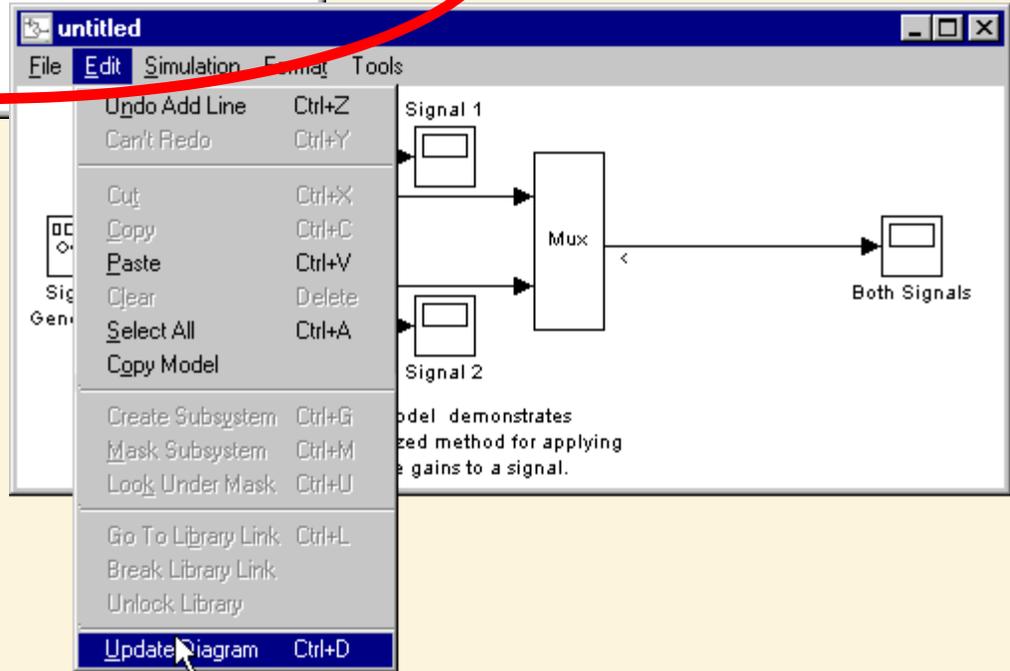
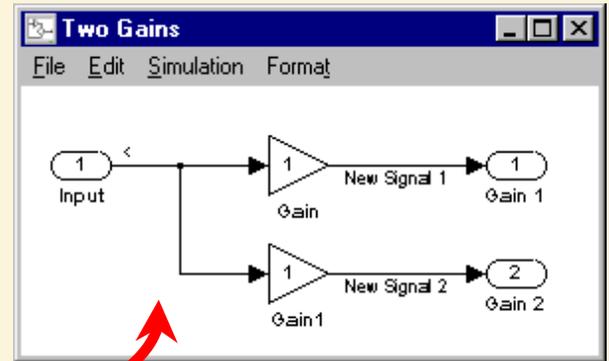
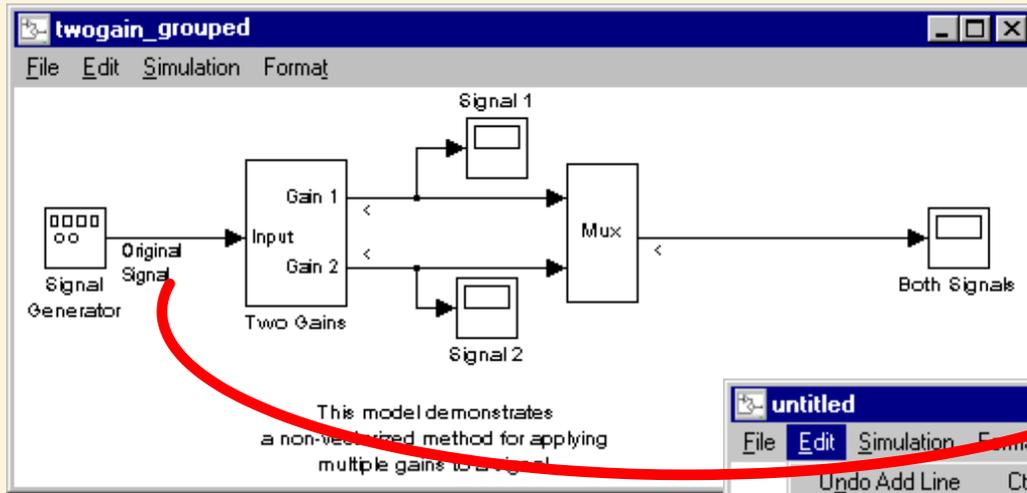
Signal Labels



Signal labels work just like annotations

» twogain

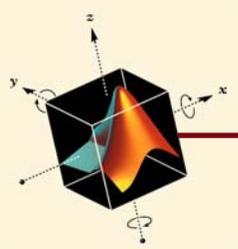
Signal Label Propagation



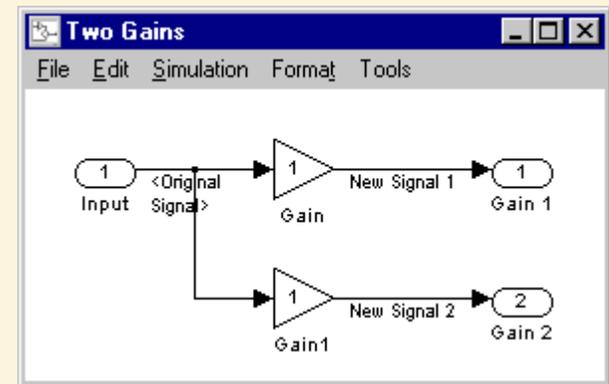
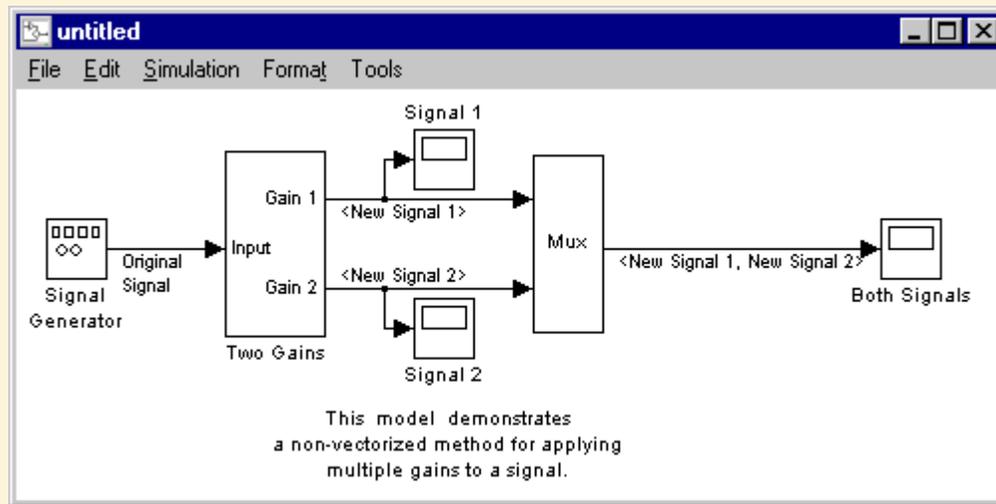
This model demonstrates a non-verified method for applying multiple gains to a signal.

Signal labels can pass through "virtual" blocks

» twogain

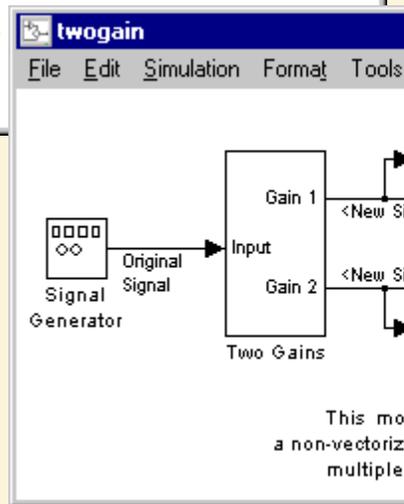
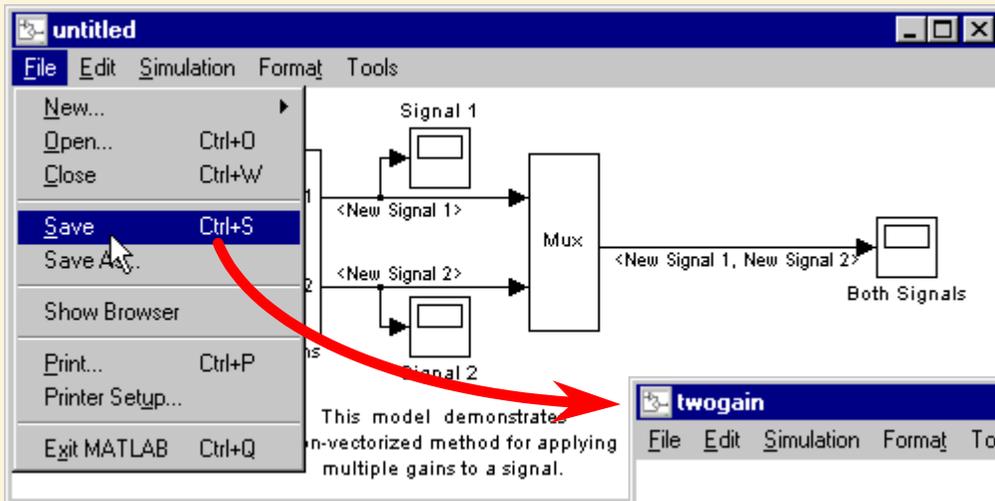


Signal Label Propagation



» twogain

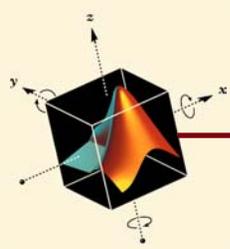
Saving (MDL file format)



```

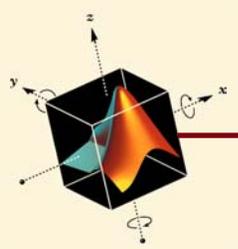
Model {
  BlockDefaults {
  }
  AnnotationDefaults {
  }
  System {
    Block
  }
  Line {
    Branch {
    }
  }
}
    
```

- » twogain
- » edit twogain.mdl



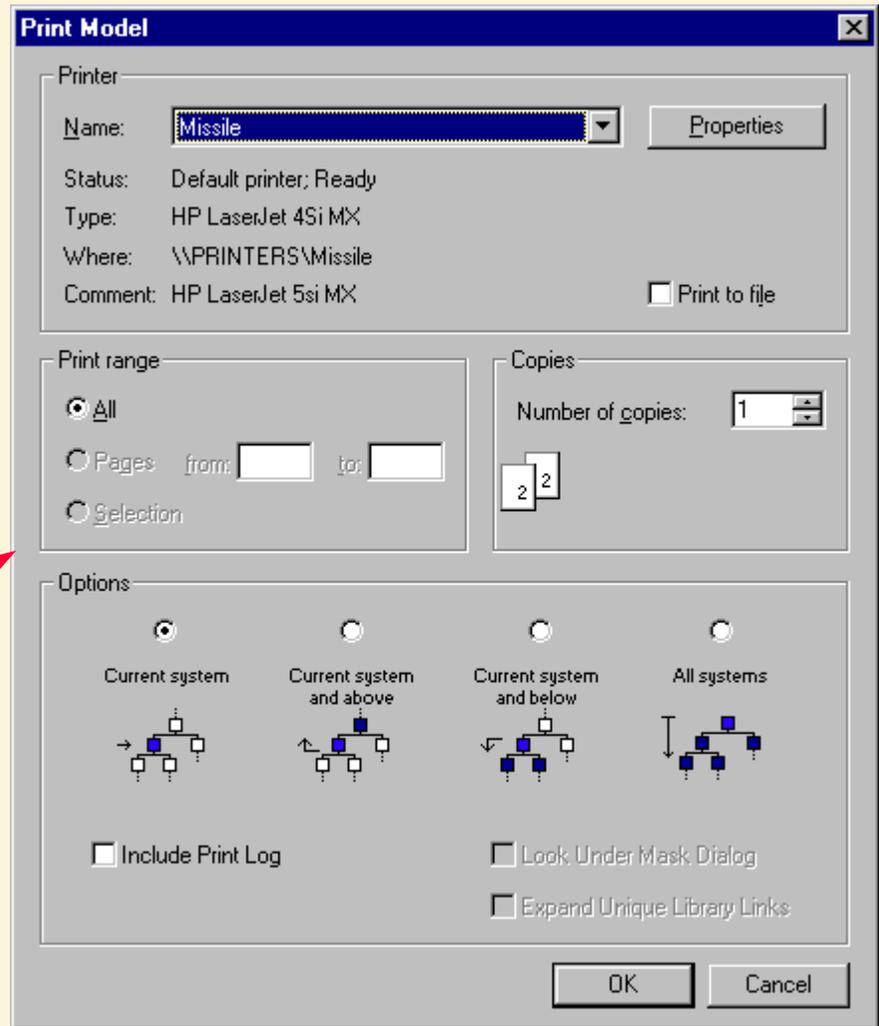
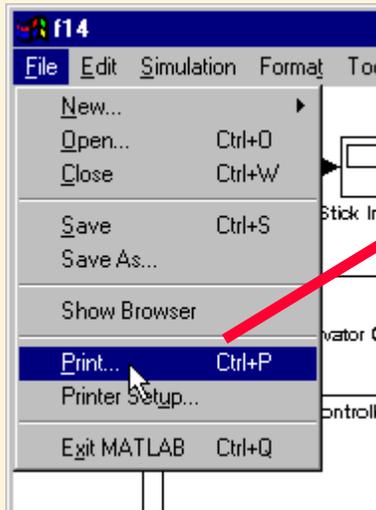
Undo / Redo

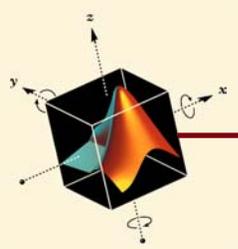
- ◆ **Select Undo or Redo from the Edit menu of the affected system.**
- ◆ **Up to 101 separate operations supported.**
- ◆ **Supported operations:**
 - ◆ Adding/deleting a block
 - ◆ Adding/deleting a line
 - ◆ Adding/deleting a model annotation
 - ◆ Editing a block name



Printing

Allows you to selectively print systems within your model.

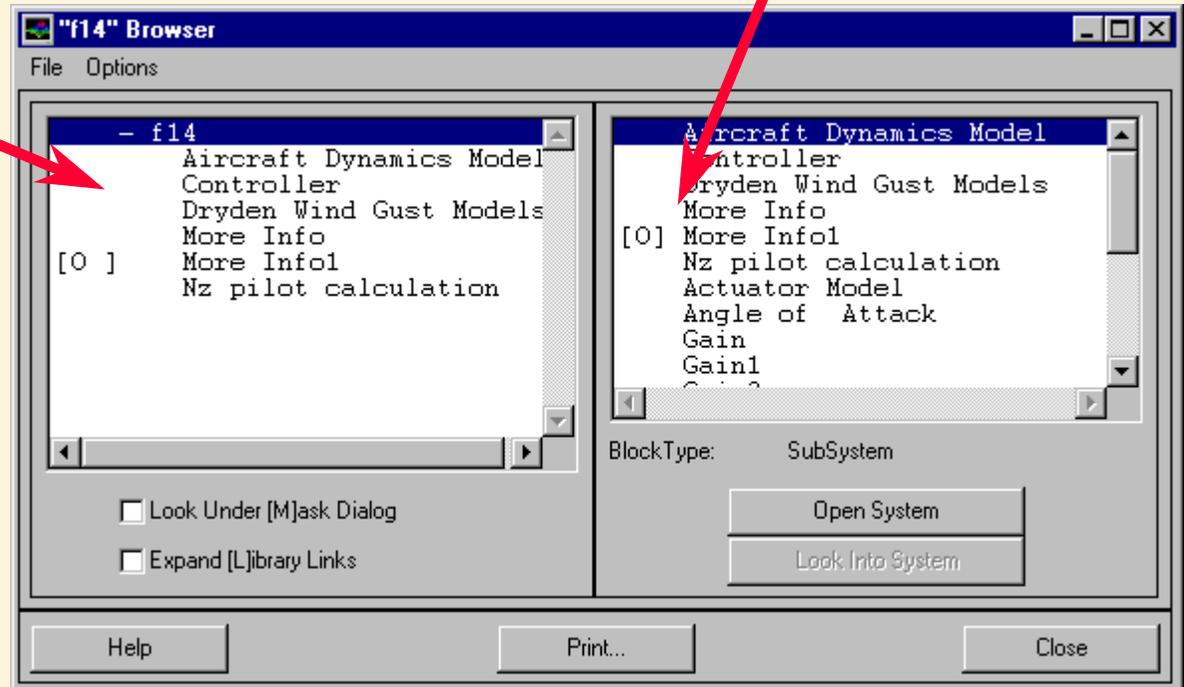
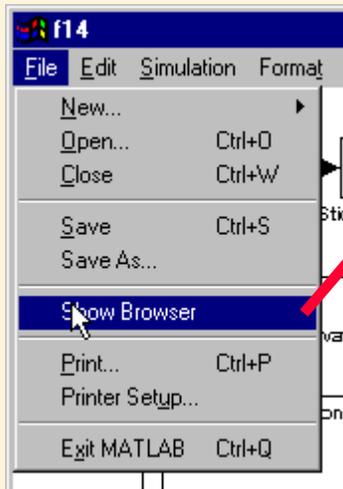




Simulink Model Browser

Current system and subsystems it contains

Blocks in the selected system



◆ **Enables you to:**

- Navigate a model hierarchically
- Open systems in a model directly
- Determine the blocks contained in a model

Simulink Help

untitled

File Edit Simulation Format Tools

Sine Wave Gain Scope

Gain

Gain

Scalar or vector gain. $y = k \cdot u$

Parameters

Gain:

1

Apply Revert **Help** Close

Pressing the Help button takes you to the Simulink Online Documentation

Simulink Linear Online Documentation - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Guide Print Security Stop

Simulink Blocks

Sources Sinks Discrete Linear Nonlinear Connections

Search

e.g., gain, switch

[About Searching](#)

Gain

Purpose

Multiply block input.

Parameters and Dialog Box

Gain

Scalar or vector gain. $y = k \cdot u$

Parameters

Gain:

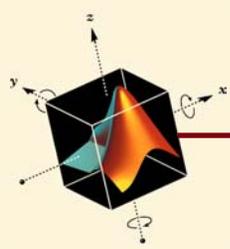
1

Apply Revert **Help** Close

Gain

The gain, specified as a scalar, vector, variable name, or expression. The default is 1.

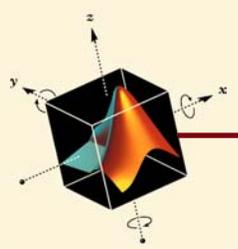
Document: Done



Tips for Building Models

“You should be able to know what is going on in a diagram by looking at the print out.”

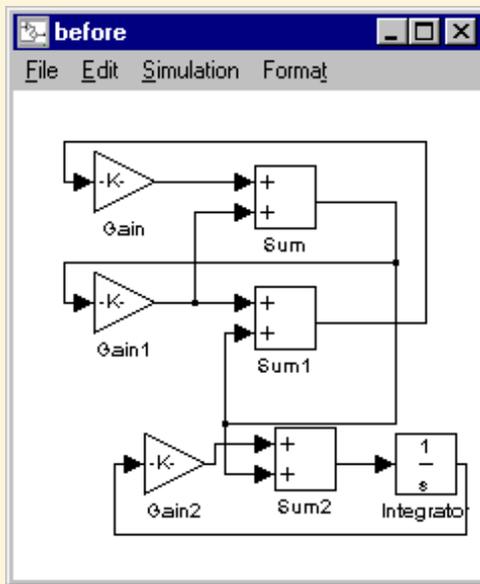
- Minimize line crossings
- Hide block names when not needed
- Show dialog parameters in block name
- Use Goto/From block wisely
- Size blocks appropriately
- Use grouping appropriately



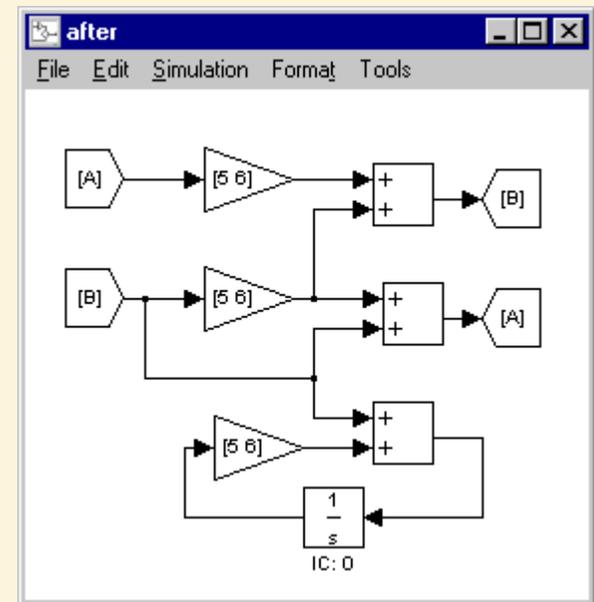
Tips for Building Models

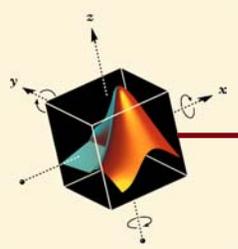
“You should be able to know what is going on in a diagram by looking at the print out.”

Before



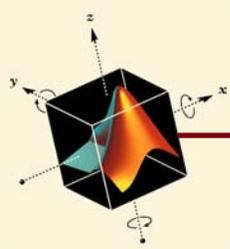
After





Libraries

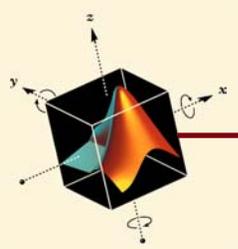
- **Enables you to copy blocks into your models from external libraries and automatically update the copied blocks when the source library changes.**



Libraries

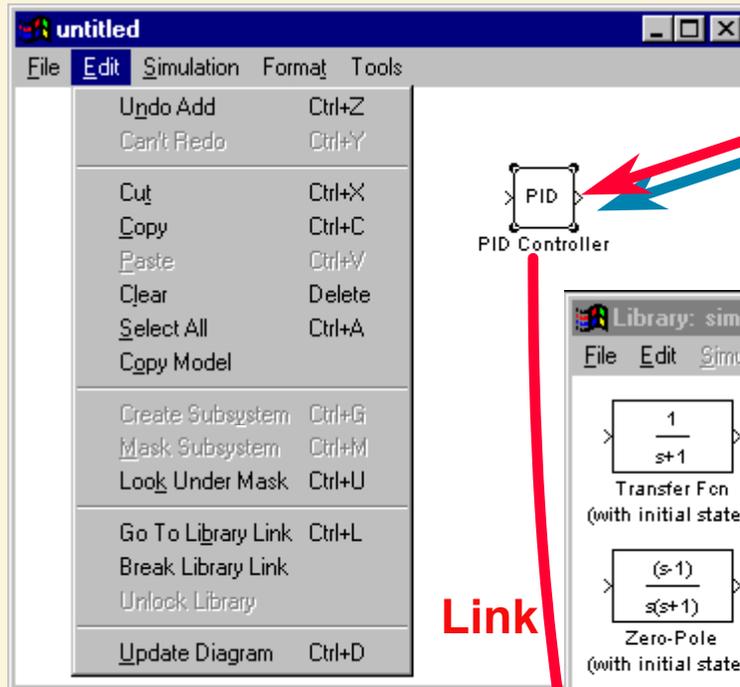
- **Terminology**

- ◆ Library - A collection of library blocks. A library must be explicitly created using 'New Library'.
- ◆ Library block - A block in a library.
- ◆ Reference block - A copy of a library block.
- ◆ Link - The connection between the reference block and its library block that allows Simulink to update the reference block when the library block changes.
- ◆ Copy - The operation that creates a reference block from either a library block or another reference block.



Libraries

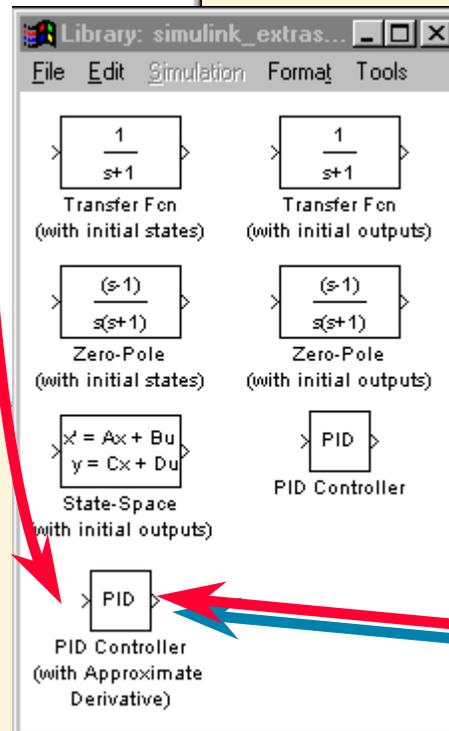
Model window



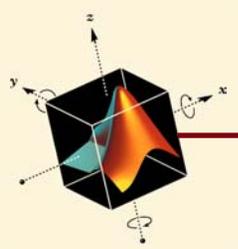
PID Controller
(Reference block)

Changes made to the Library block will affect all Reference blocks.

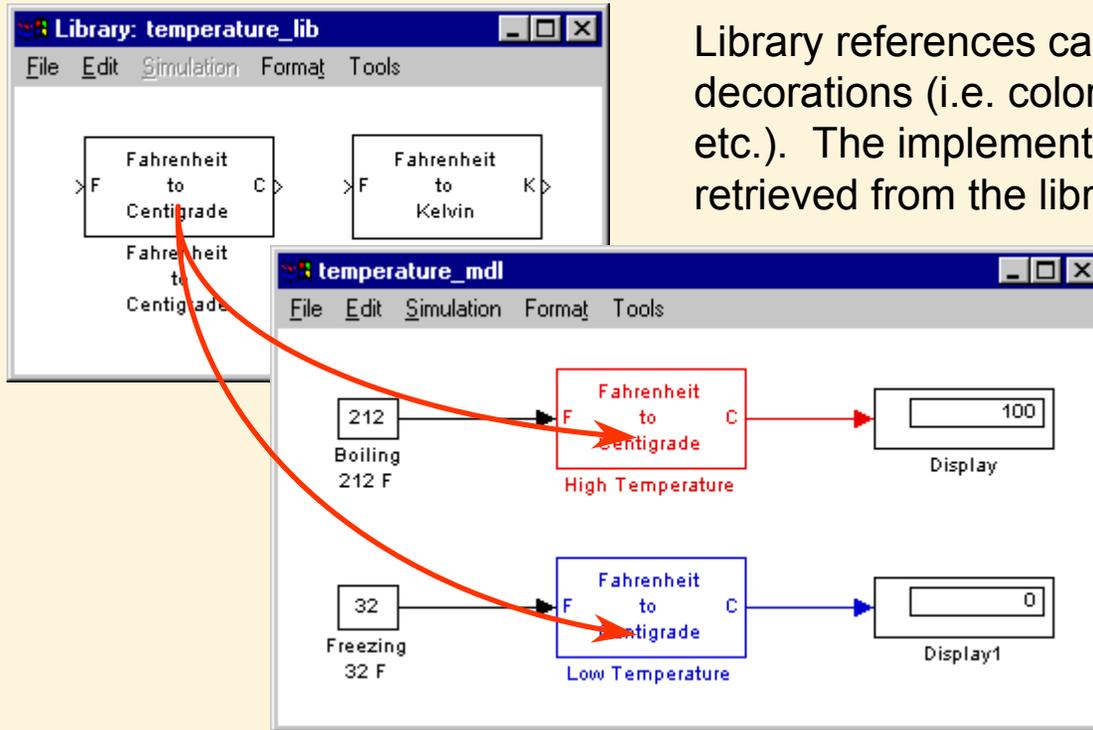
Library window



Library block

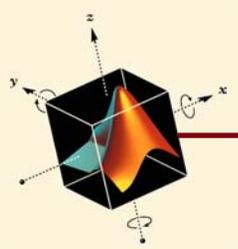


Example



Library references can have different decorations (i.e. colors, drop shadow, etc.). The implementation is always retrieved from the library.

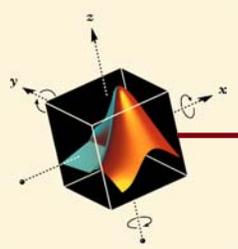
>> temperature_lib



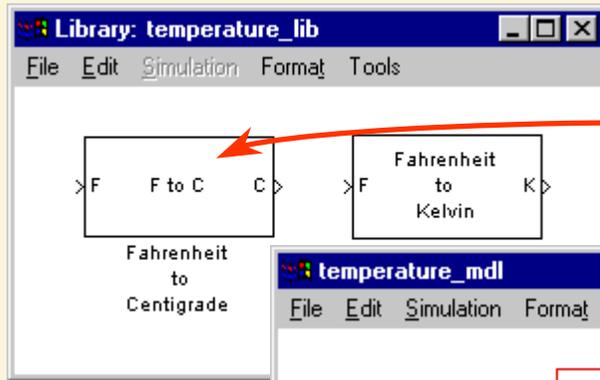
Breaking a Library Link

The screenshot shows the Simulink environment for a model named 'temperature_md1'. The 'Edit' menu is open, and the 'Break Library Link' option is highlighted. The diagram contains two 'Display' blocks. The top block, labeled 'Display', shows the value '100' and is connected to a library source (indicated by a red arrow and the label 'C'). The bottom block, labeled 'Display1', shows the value '0' and is also connected to the same library source (indicated by a blue arrow and the label 'C'). The menu items include: Undo Block Name (Ctrl+Z), Can't Redo (Ctrl+Y), Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Clear (Delete), Select All (Ctrl+A), Copy Model, Create Subsystem (Ctrl+G), Mask Subsystem (Ctrl+M), Look Under Mask (Ctrl+U), Go To Library Link (Ctrl+L), Break Library Link (highlighted), Unlock Library, and Update Diagram (Ctrl+D).

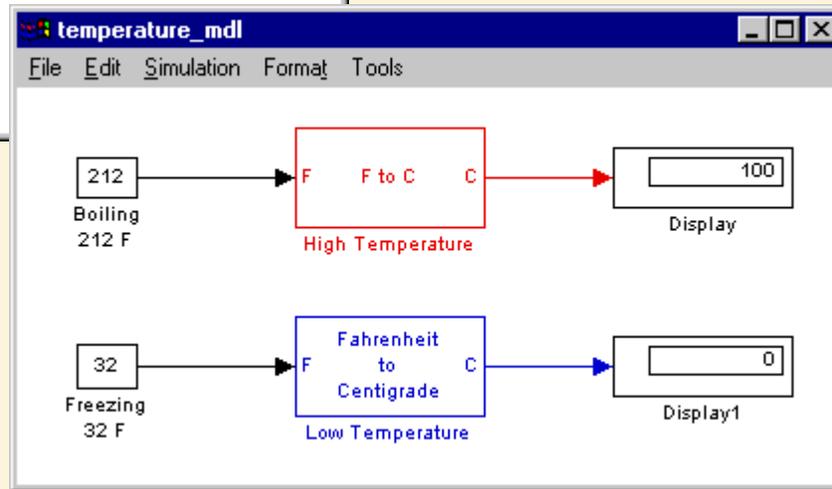
A library link can be broken, making it independent of changes to the library source.



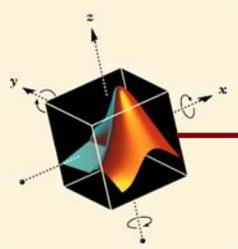
Breaking a Library Link



The Fahrenheit to Centigrade icon has been changed.

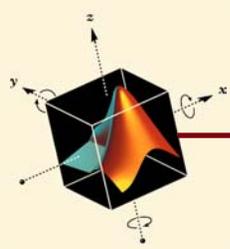


The **Low Temperature** block does not inherit the icon change made in the library.



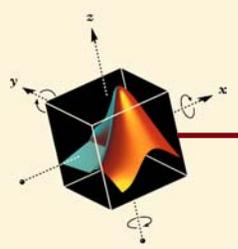
Libraries

- **All blocks have a LinkStatus property**
 - ◆ Possible values of LinkStatus are:
 - 'none' - the block is not a reference block
 - 'resolved' - the block is a reference block and it refers to a valid library
 - 'unresolved' - the block is a reference block and it does not refer to a valid library



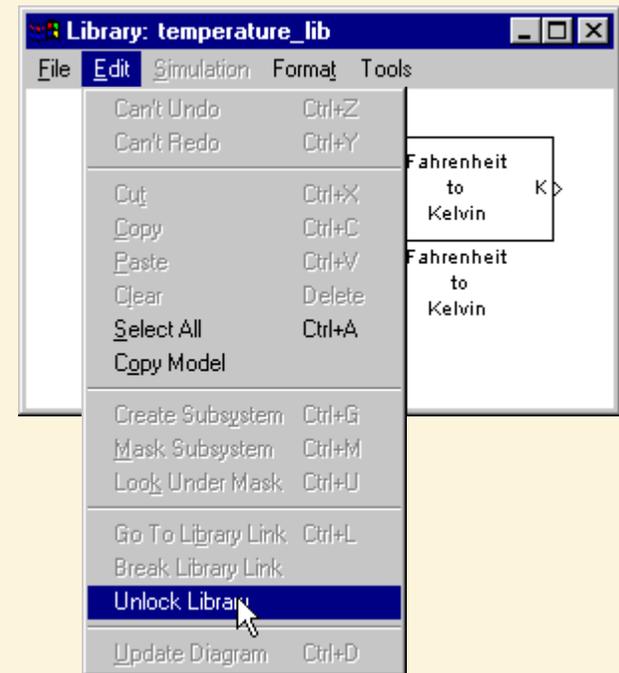
Libraries

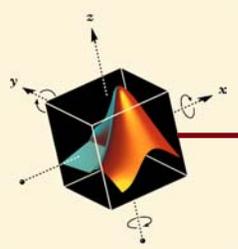
- **Model library information may be obtained using the libinfo command.**
 - ◆ returns a data structure with fields
 - Block - the block path
 - Library - the library name
 - ReferenceBlock - the reference block path
 - LinkStatus - the link status of the block



Libraries

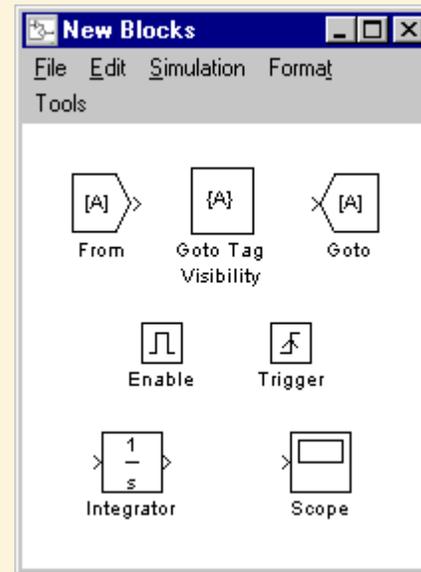
- A library may be locked which means that no further changes can be made until it is unlocked.
- Links to locked libraries are only instantiated once.
- Links to unlocked libraries are re-instantiated each time you 'Update Diagram' or start a simulation.

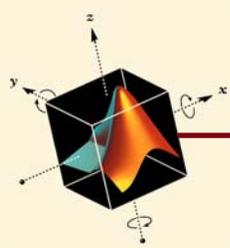




New Blocks

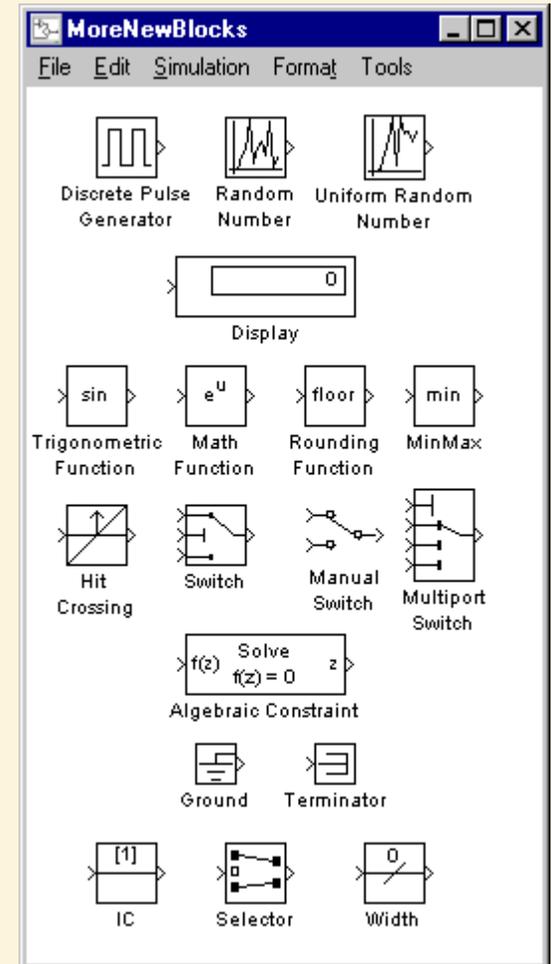
- ◆ **Goto/From**
- ◆ **Enable/Trigger**
- ◆ **Integrator**
- ◆ **Scope**
- ◆ **Others...**

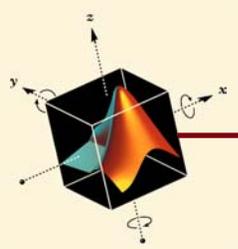




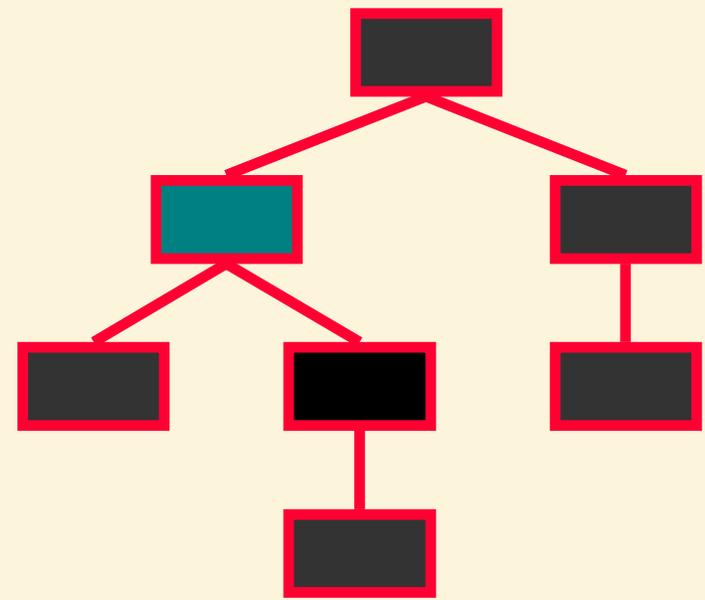
New Blocks

- ◆ Sources - Discrete Pulse Generator, Random Number, Uniform Random Number
- ◆ Sinks - Display
- ◆ Nonlinear - Trigonometric Function, Math Function, Rounding Function, MinMax, Hit Crossing, Manual Switch, Multiport Switch, Algebraic Constraint
- ◆ Connections - Data Store Read, Data Store Memory, Data Store Write, Ground, Terminator, Initial Condition, Selector, Width

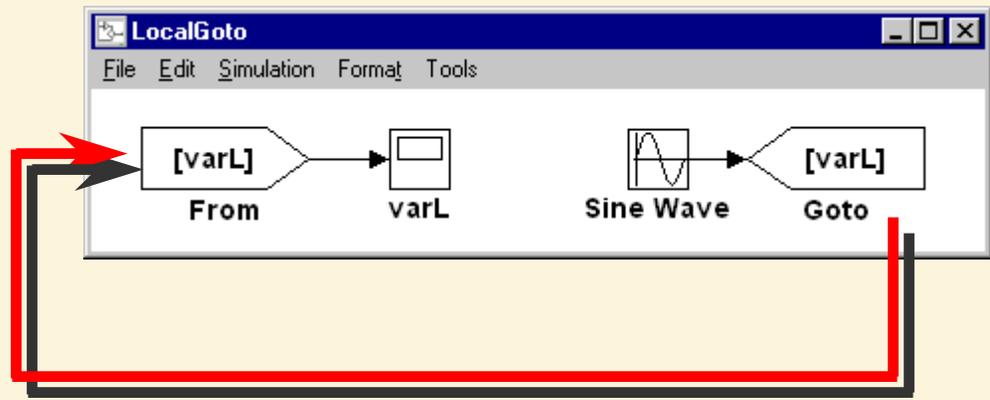




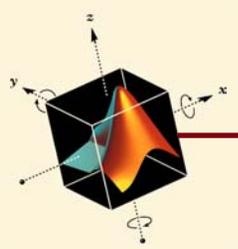
Goto / From Blocks



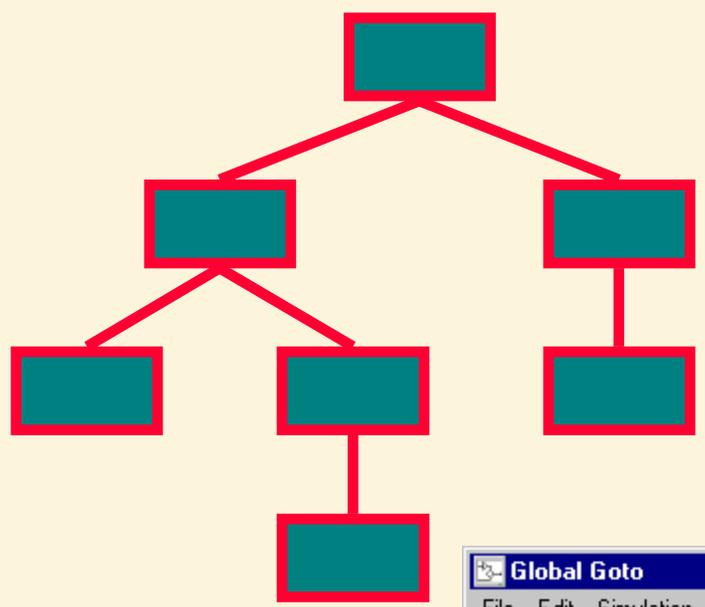
Local [] is only on the current window



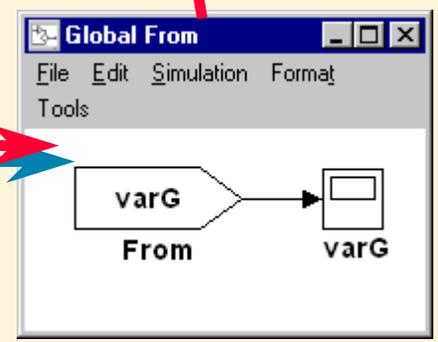
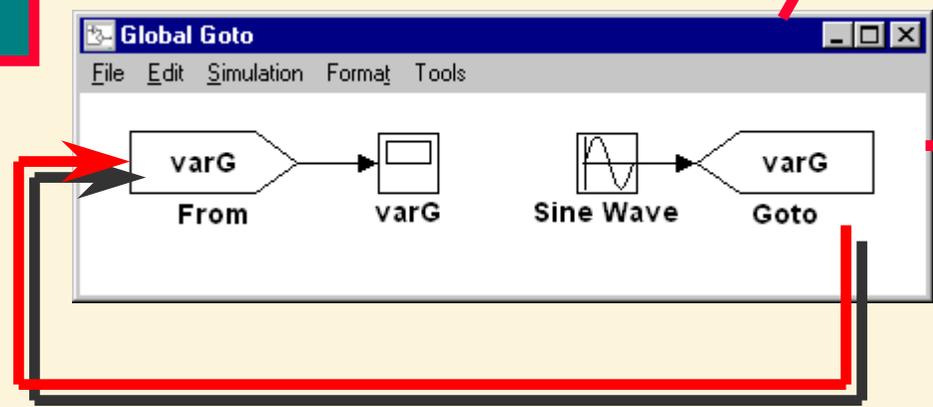
>> LocalGoto



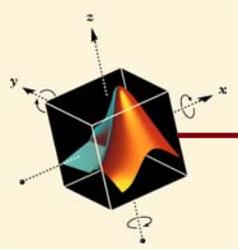
Goto / From Blocks



Global is all systems

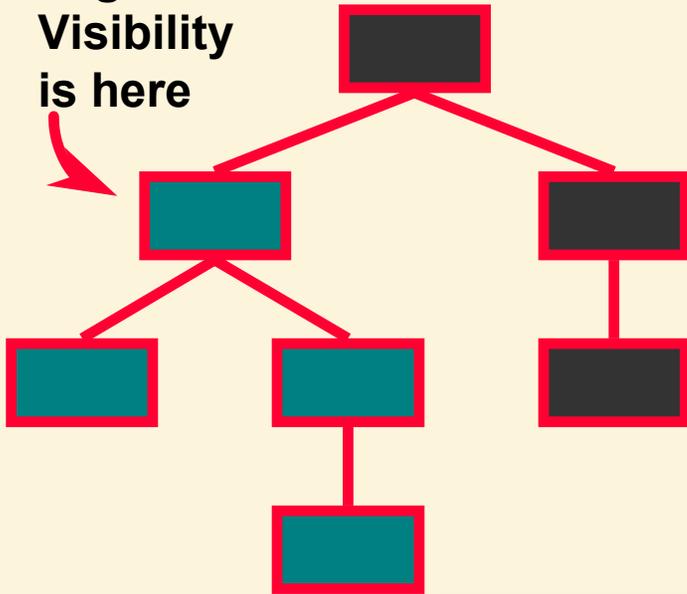


>> GlobalGoto

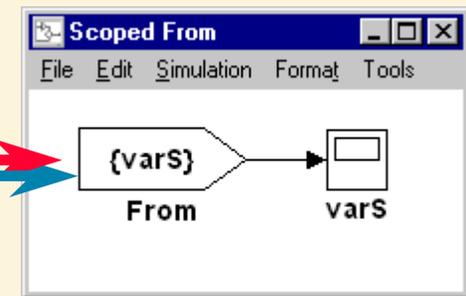
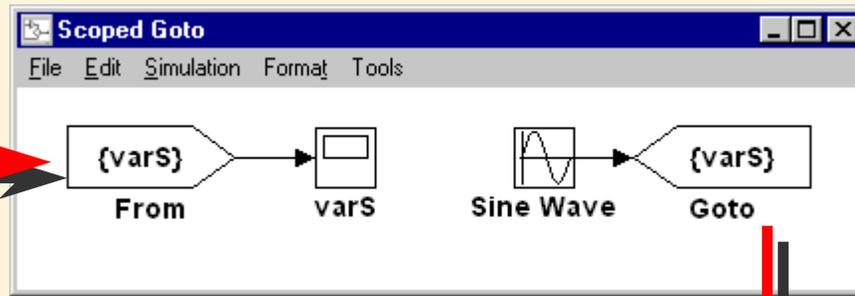
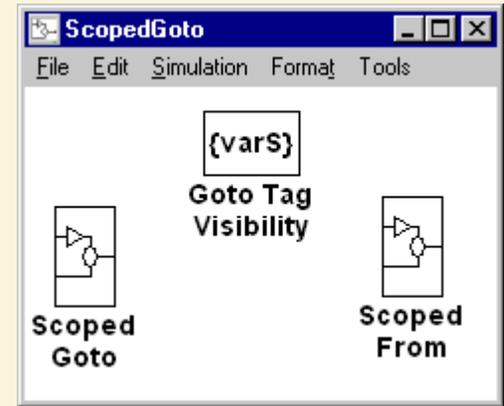


Goto / From Blocks

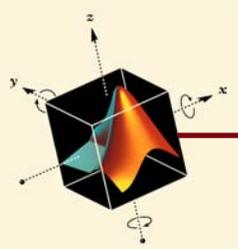
Tag
Visibility
is here



Scoped {} is all systems beneath the Tag Visibility block

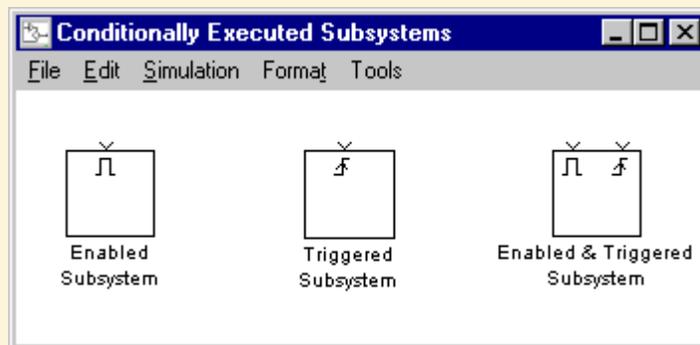


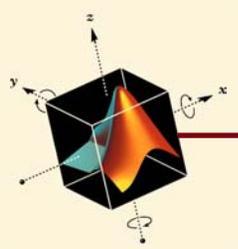
» ScopedGoto



Enable / Trigger

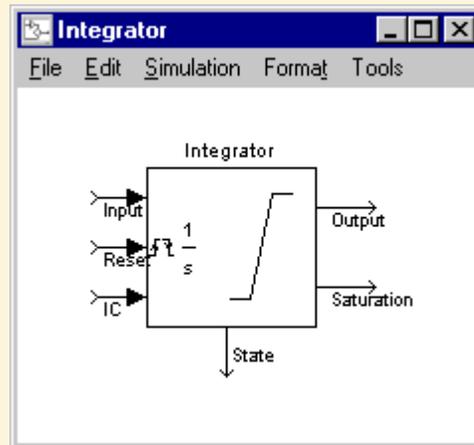
- ◆ An Enable block, when added to a subsystem, causes the subsystem to be conditionally executed at each simulation step for which the control signal to the subsystem's enable port is positive.
- ◆ A Trigger block, when added to a subsystem, causes the subsystem to execute whenever a trigger event occurs. The execution is for one time step only.
- ◆ Both Enable and Trigger blocks can be added to a subsystem, combining the affects of both.

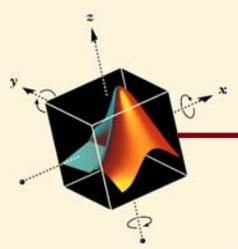




Integrator

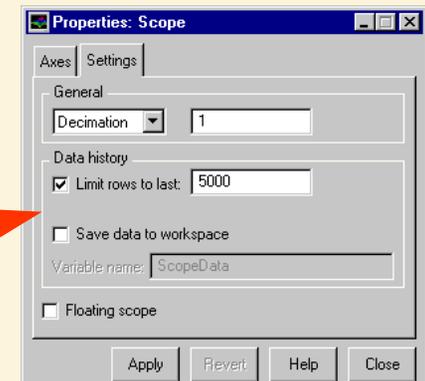
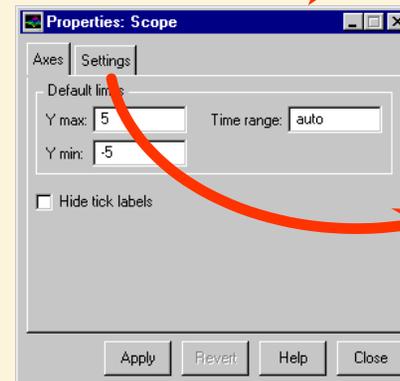
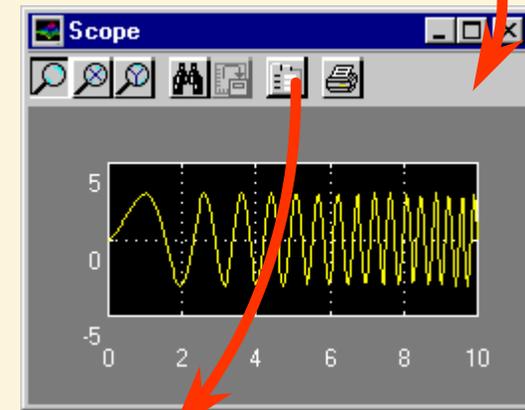
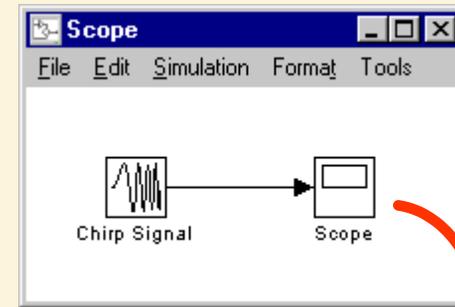
- ◆ **External reset (rising, falling, either)**
- ◆ **External initial condition**
- ◆ **Output limiting with saturation port**
- ◆ **State port**
 - ◆ useful for passing states between conditionally executed subsystems
 - ◆ useful for feedback without causing algebraic loops

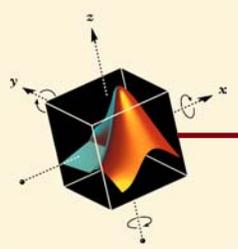




Scope

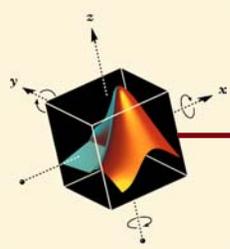
- ◆ Zoom in on signal
- ◆ Locate signal
- ◆ Save signal to workspace or file
- ◆ Print signal





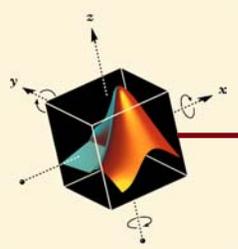
Running a Simulation

- ◆ **No states**
- ◆ **Continuous**
- ◆ **Discrete**
- ◆ **Hybrid**



Running a Simulation

- ◆ **Setting parameters**
- ◆ **Getting data into models**
- ◆ **Continuous systems**
- ◆ **Discrete systems**
- ◆ **Discontinuities (zero crossing events)**
- ◆ **Algebraic Loops (implicit equations)**
- ◆ **From the command line**
- ◆ **Improving performance and accuracy**



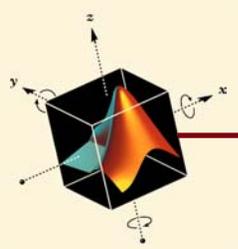
Setting Parameters

The image shows a Simulink model window titled 'twogain'. The 'Simulation' menu is open, and the 'Parameters...' option is selected. A red arrow points from this menu item to the 'Simulation parameters: twogain' dialog box. The dialog box has several tabs: 'Solver', 'Workspace I/O', 'Diagnostics', 'RTW', and 'RTW External'. The 'Solver' tab is active, showing the following settings:

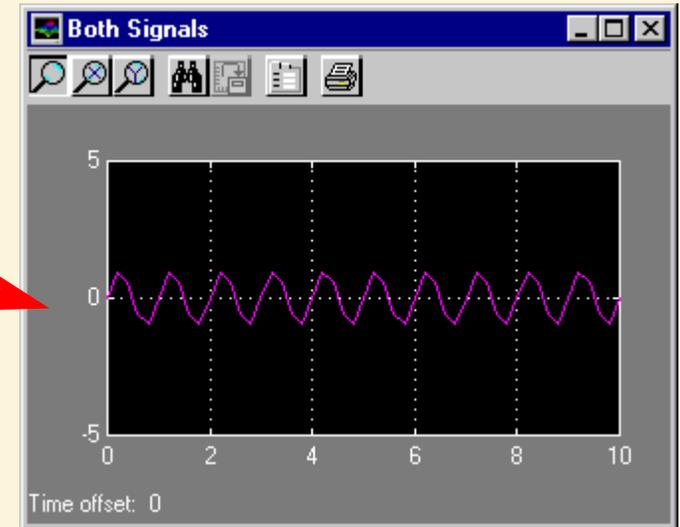
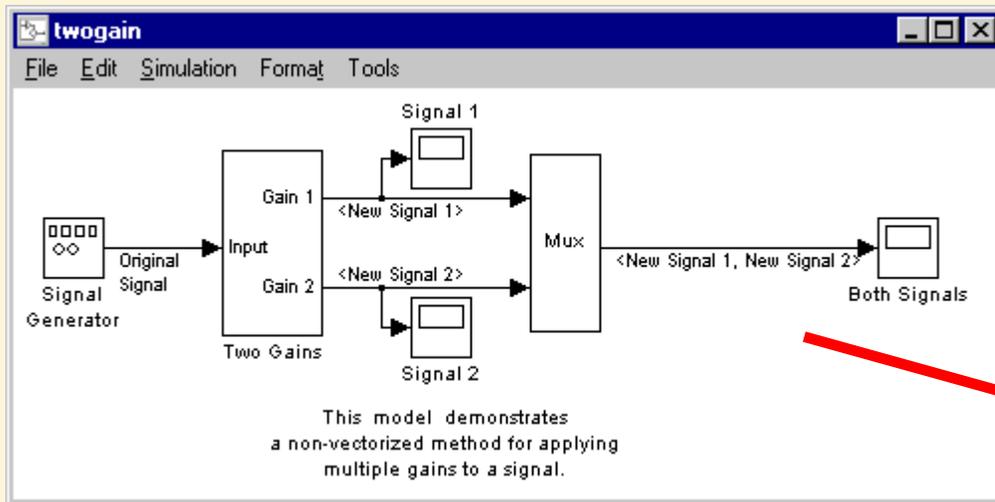
- Simulation time: Start time: 0.0, Stop time: 10.0
- Solver options: Type: Variable-step, Solver: ode45 (Dormand-Prince)
- Max step size: auto, Relative tolerance: 1e-3
- Initial step size: auto, Absolute tolerance: 1e-6
- Output options: Refine output, Refine factor: 1

Buttons at the bottom of the dialog include 'Apply', 'Revert', 'Help', and 'Close'. The background model shows a Signal Generator connected to a Mux block, which is connected to two Gain blocks (Gain 1 and Gain 2), which are then connected to a Scope block. A text box at the bottom of the model window reads: 'This model demonstrates a non-vectorized method for applying multiple gains to a signal.'

» twogain

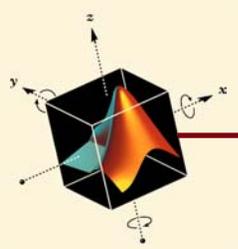


Simulating

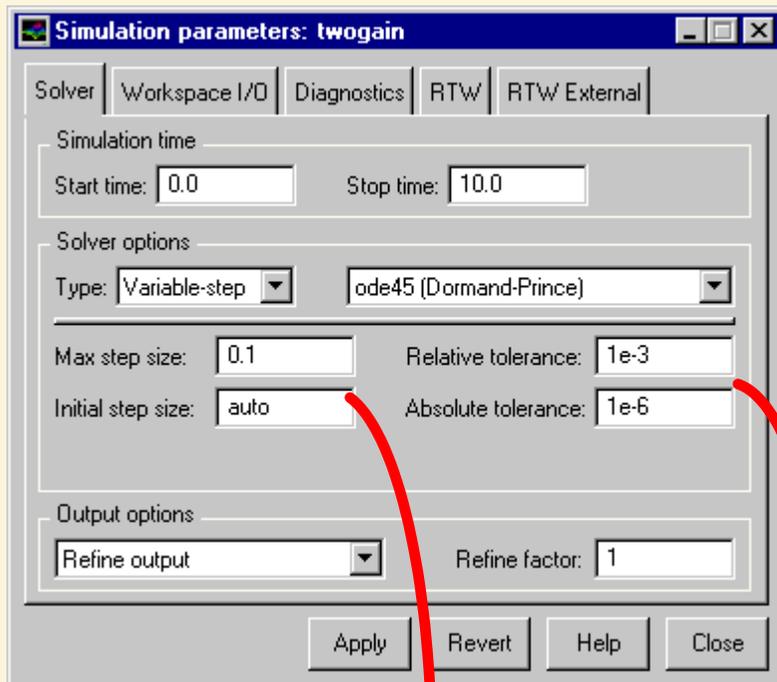


Use:
Variable-step discrete time solver
Max. Step Size = 0.1

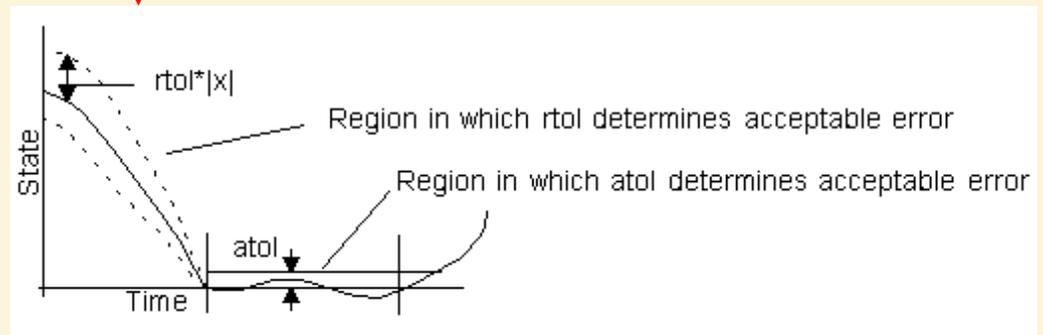
» twogain



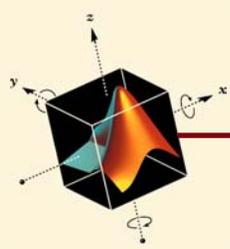
Setting Parameters



$$h_{\max} = \frac{t_{\text{stop}} - t_{\text{start}}}{50}$$



» twogain



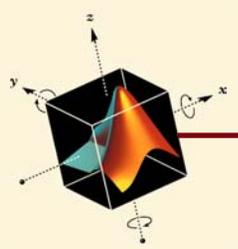
Available Solvers

- ◆ **Variable-step solvers**

- ◆ Modify step size during simulation
- ◆ Error control and zero crossing detection
- ◆ ode45, ode23, ode113, ode15s, ode23s, discrete

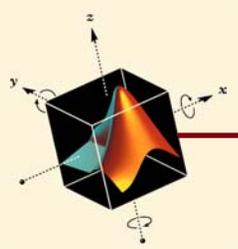
- ◆ **Fixed-step solvers**

- ◆ Same step size throughout simulation
- ◆ No error control or zero crossing detection
- ◆ ode5, ode4, ode3, ode2, ode1, discrete



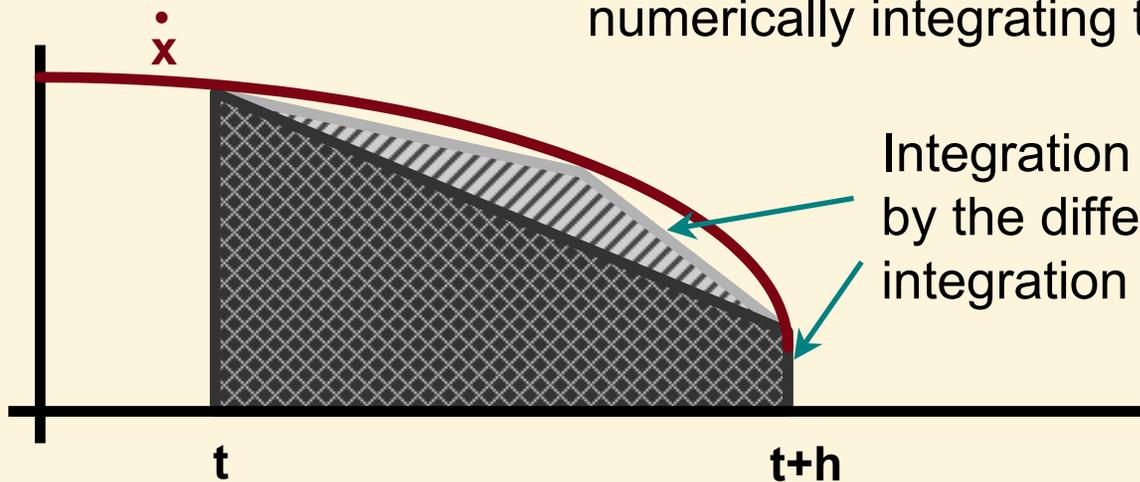
Available Solvers

- ◆ **Simulink selects a default solver**
 - ◆ ode45 solver used for models with continuous states
 - ◆ Variable-step discrete-time solver used for models with no continuous states
- ◆ **See supplied paper by Reichelt and Shampine**



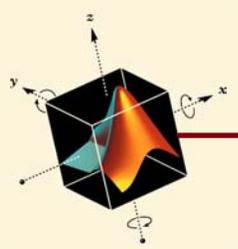
Step Size Calculation (variable step)

Simulink calculates continuous states by numerically integrating their derivatives.

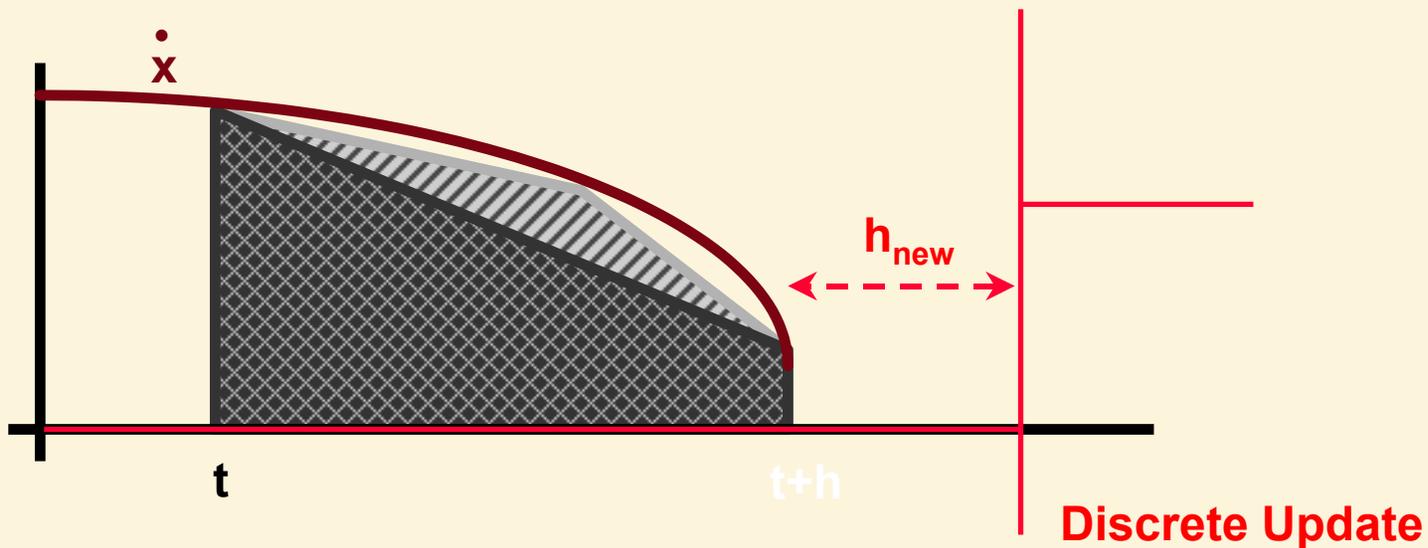


Integration error is approximated by the difference between two integration orders.

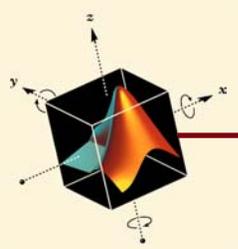
- If the error is acceptable, the simulation proceeds. Otherwise, step size is reduced and the integration is repeated.



Step Size Calculation (variable step)

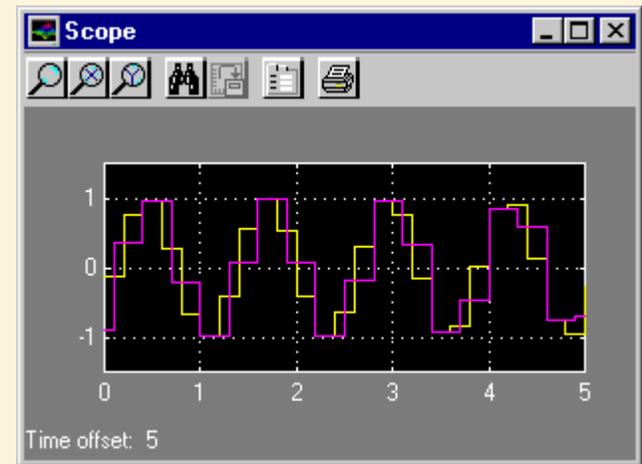
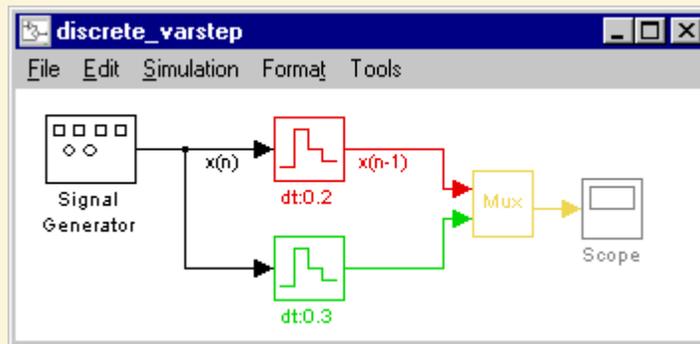


- If the error is acceptable, the simulation proceeds. Otherwise, step size is reduced and the integration is repeated.
- Step size is adjusted to coincide with updates of discrete blocks.

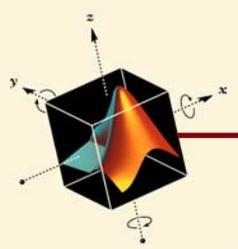


Step Size Calculation (variable step)

- ◆ Purely discrete systems
 - ◆ The step size is adjusted to coincide with the sample times of the blocks.

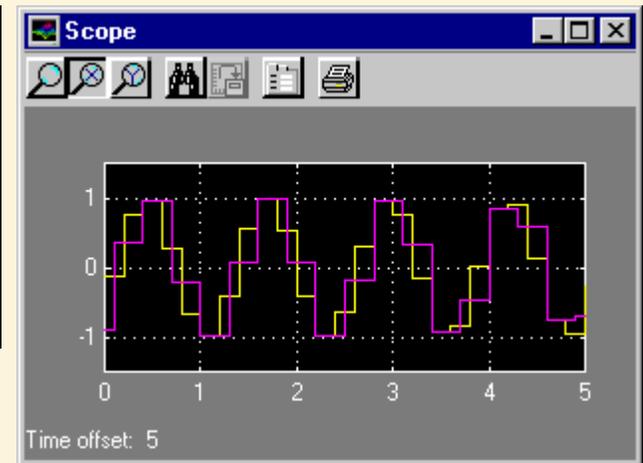
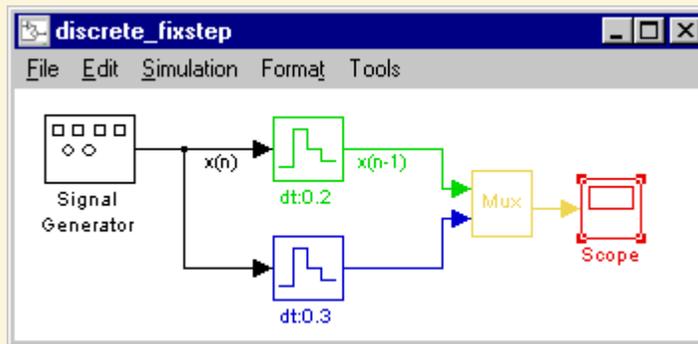


» discrete_varstep

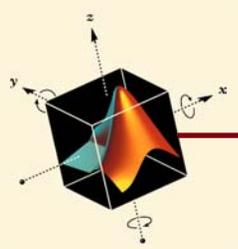


Step Size Calculation (fixed step)

- ◆ **Purely discrete systems**
 - ◆ The step size is determined by taking the greatest common divisor (GCD) of the sample times of the blocks:
 $T_{s1} = 0.2, T_{s2} = 0.3$
 $LCM = 0.1$



» discrete_fixstep



Step Size Calculation (fixed step)

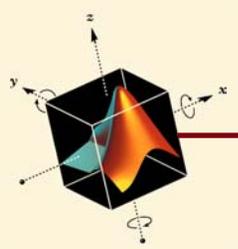
- ◆ **Purely discrete systems**

- ◆ The actual number of simulation steps may be more than expected if the sample times do not have a common multiple:

Numbers in red are those times that do not correspond to a sample time.

```
tout =  
0  
0.1000  
0.2000  
0.3000  
0.4000  
0.5000  
0.6000  
0.7000  
0.8000
```

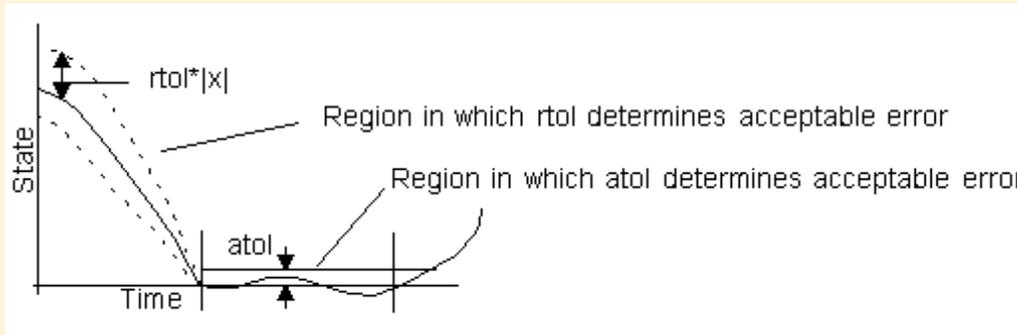
» discrete_fixstep



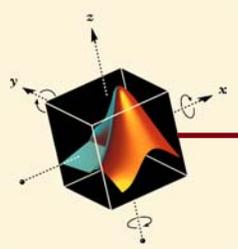
Error Tolerance

Integration step size is chosen to satisfy error constraint:

$$\text{error}_i < \max\{ \text{absolute_tol}, \text{relative_tol} * \text{abs}(x_i) \}$$



- Relative tolerance scales the acceptable error with a states magnitude, which is useful for systems with widely varying state values. (e.g.. 100,000 RPM vs 1.0 ATM)
- Absolute tolerance prevents the step size from becoming infinitesimal when a state crosses zero. It is used to determine tolerances around zero (i.e.. $|x| < \text{absolute_tol} / \text{relative_tol}$).
- Simulink 2 allows different absolute tolerances for each state.



Getting Data Into and Out of Models

Simulation parameters: Datal0

Solver | **Workspace I/O** | Diagnostics | RTW | RTW External

Load from workspace

Input: [t, u]

Save to workspace

Time: tout

States: xout

Output: yout

States

Load initial: xInitial

Save final: xFinal

Save options

Limit rows to last: 1000

Decimation: 1

Apply Revert Help Close

Datal0

File Edit Simulation Format

1 Constant Scope

untitled.mat From File untitled.mat To File

[T,U] From Workspace simout To Workspace

1 In 1 Out

Properties: Scope

Axes Settings

General

Decimation 1

Data history

Limit rows to last: 5000

Save data to workspace

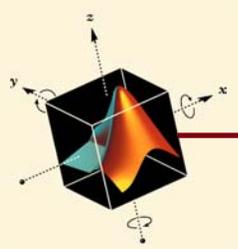
Variable name: ScopeData

Floating scope

Apply Revert Help Close

Applies to Inports and Outports on top level only

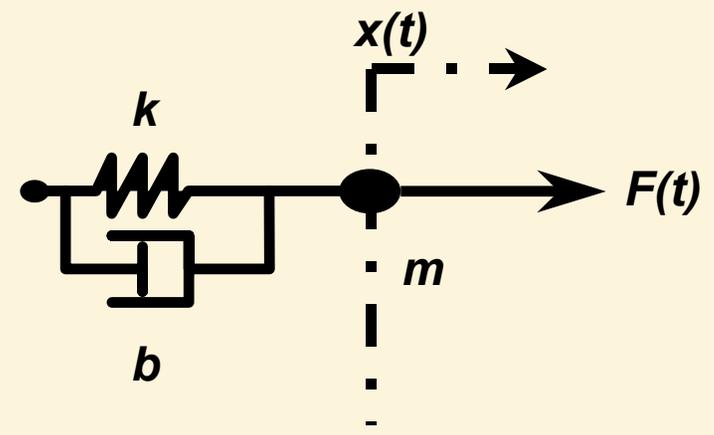
» dataio



Continuous models

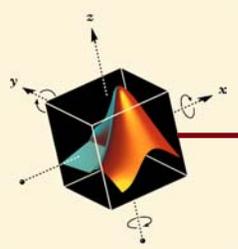
Ordinary differential equations

$$F = m\ddot{x} + b\dot{x} + kx$$

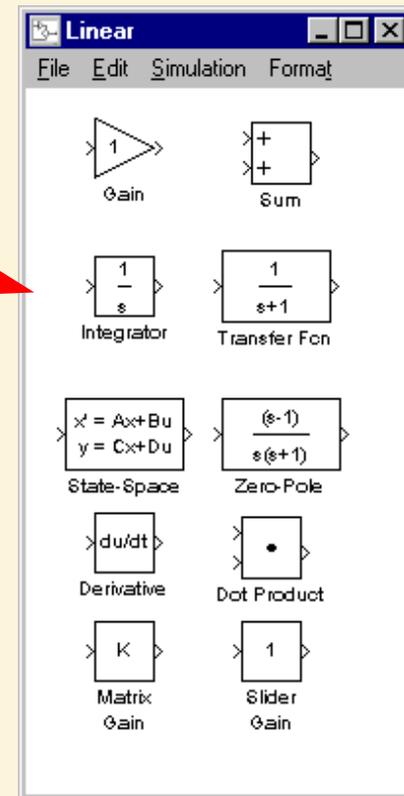
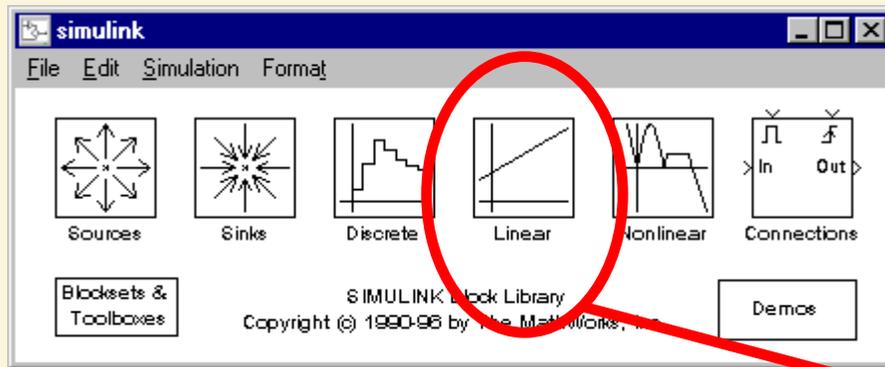


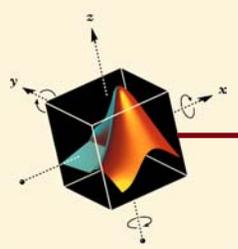
$$\frac{x}{F} = \frac{y}{u} = \frac{1}{ms^2 + bs + k}$$

$$[m \ b \ k]$$

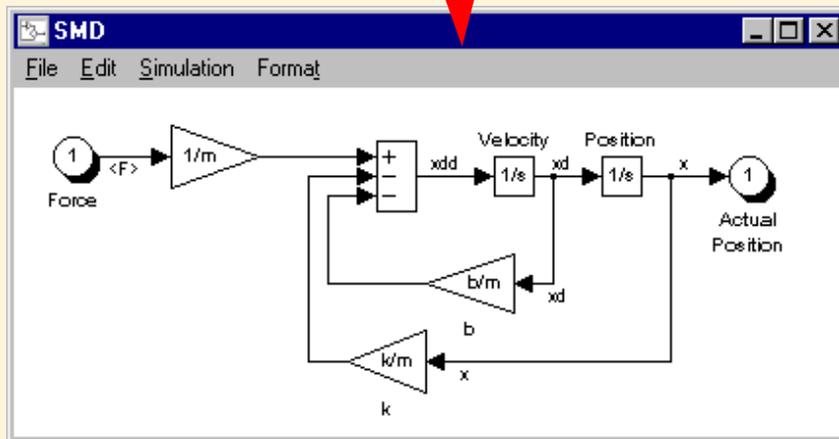
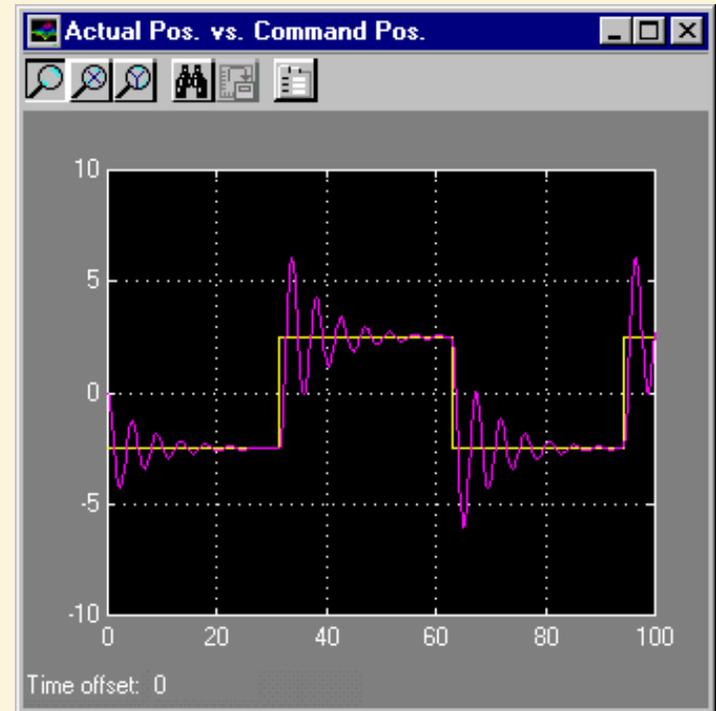
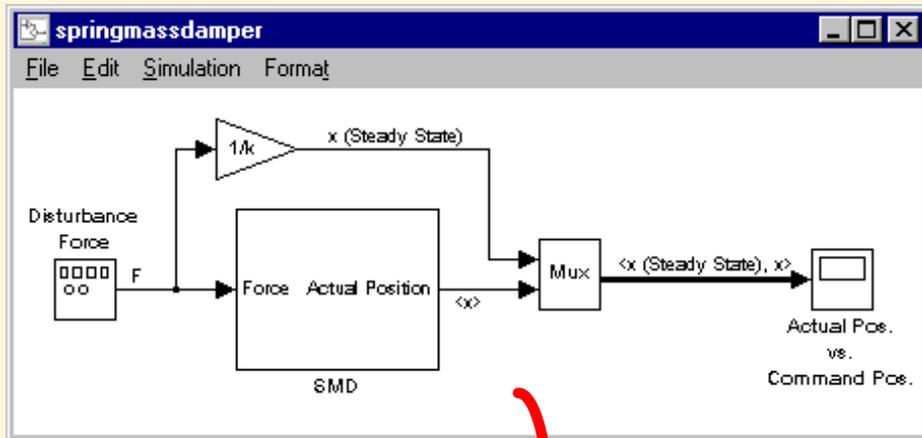


The linear block library

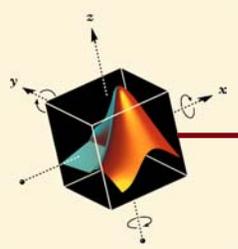




Spring-Mass-Damper



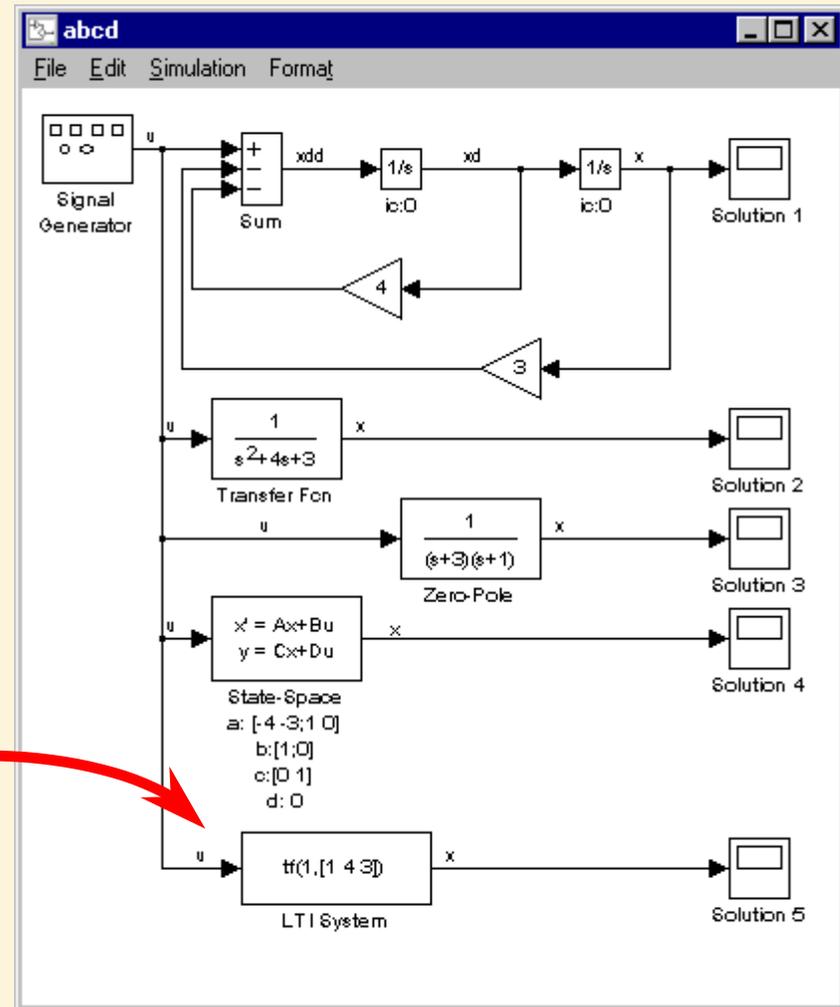
» springmassdamper



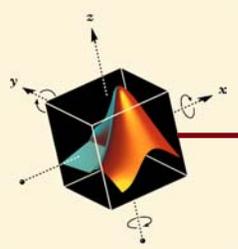
Modelling continuous systems

$$\ddot{x} = -4\dot{x} - 3 \cdot x + u$$

**Controls Toolbox
required for this solution**



» abcd



Exercise: Foxes and Bunnies

Lotka-Volterra predator-prey model

Foxes:

$$\dot{y}_1 = (1 - \alpha \cdot y_2) y_1$$

Bunnies:

$$\dot{y}_2 = (-1 + \beta \cdot y_1) y_2$$

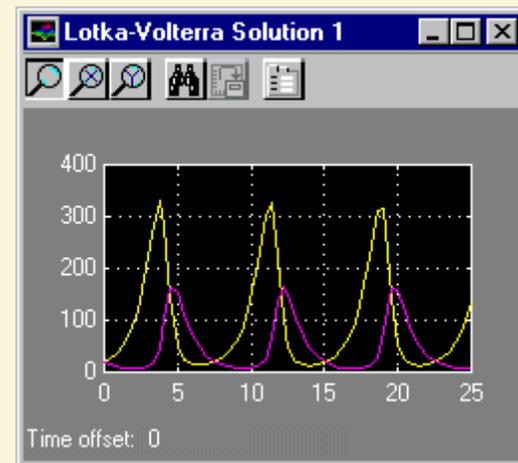
Initial conditions:

$$y_1(0) = 20$$

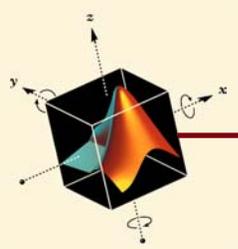
$$y_2(0) = 20$$

$$\alpha = 0.02$$

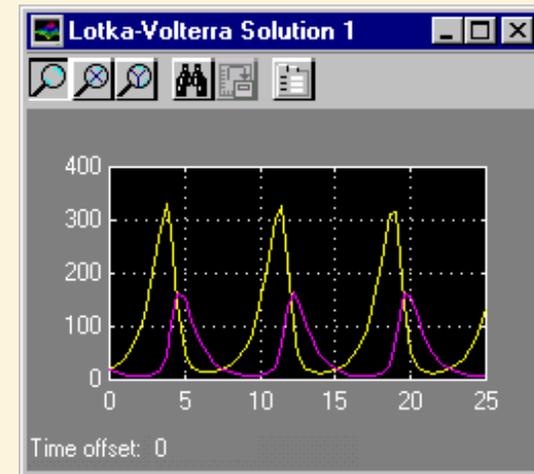
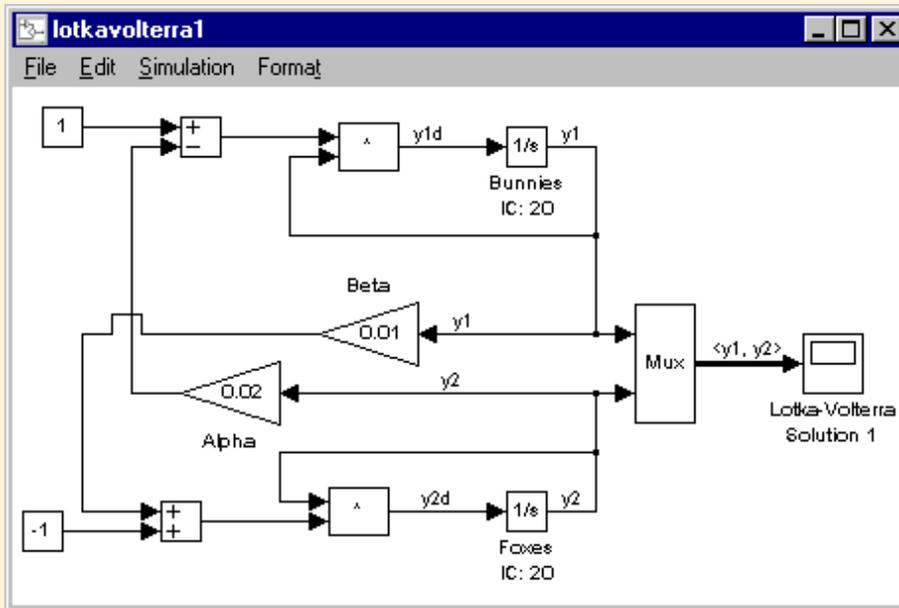
$$\beta = 0.01$$



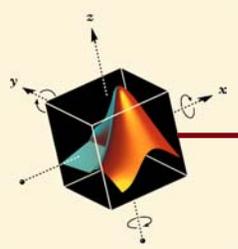
Build two different representations of the equations above and simulate the models for 25 seconds



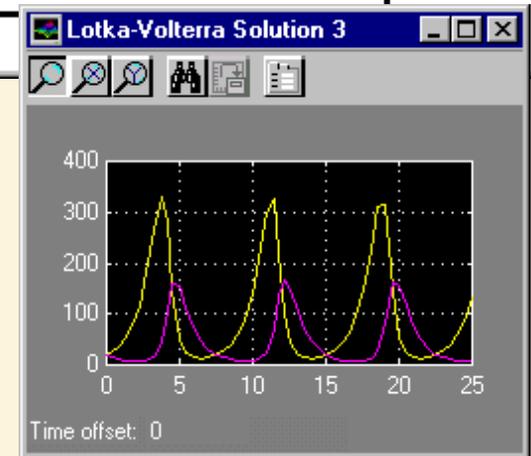
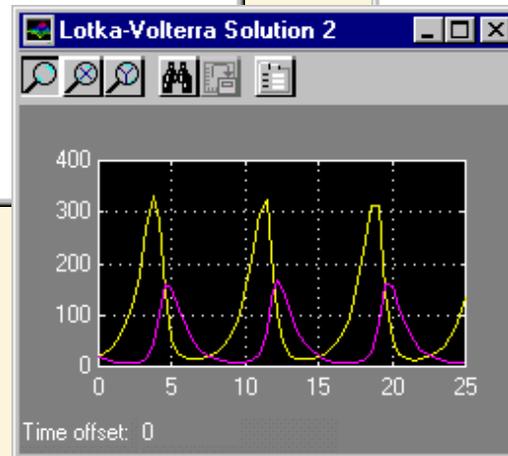
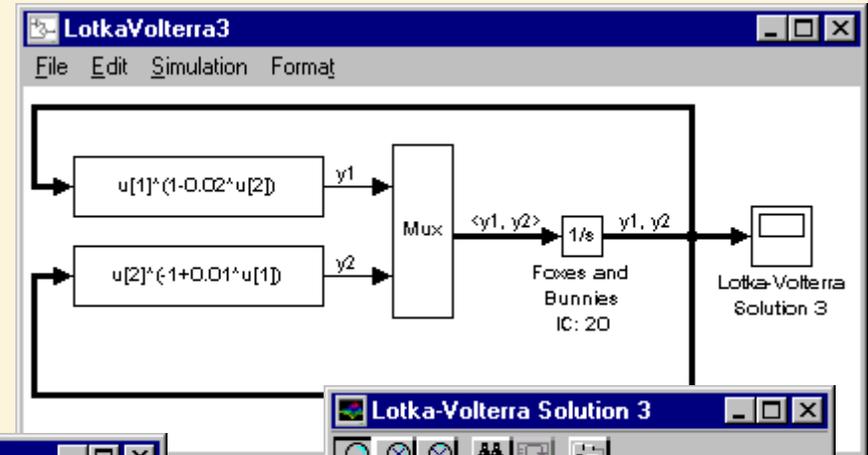
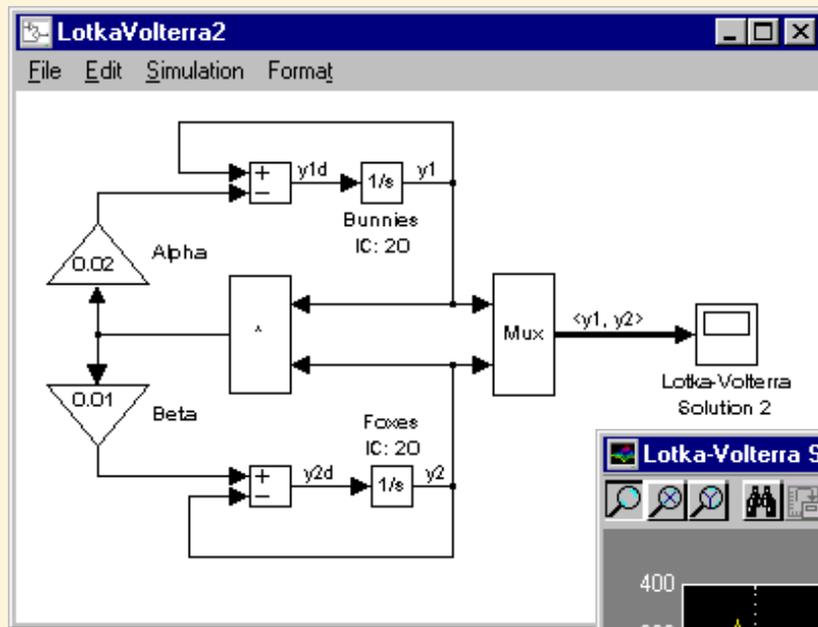
Solution: Foxes and Bunnies



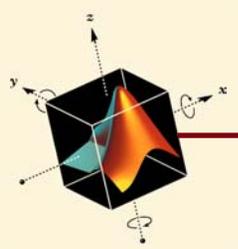
» lotkavolterra1



Additional Solutions: Foxes and Bunnies



- » lotkavolterra2
- » lotkavolterra3



Exercise: Lorenz Attractor

Lorenz attractor equations

$$\begin{aligned}\dot{x}_1 &= \sigma(-x_1 + x_2) \\ \dot{x}_2 &= -x_1x_3 + \rho x_1 - x_2 \\ \dot{x}_3 &= x_1x_2 - \beta x_3\end{aligned}$$

Initial conditions:

$$x_1 = 1.7$$

$$x_2 = 1.2$$

$$x_3 = 20$$

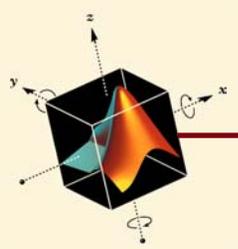
$$\sigma = 10$$

$$\beta = 8/3$$

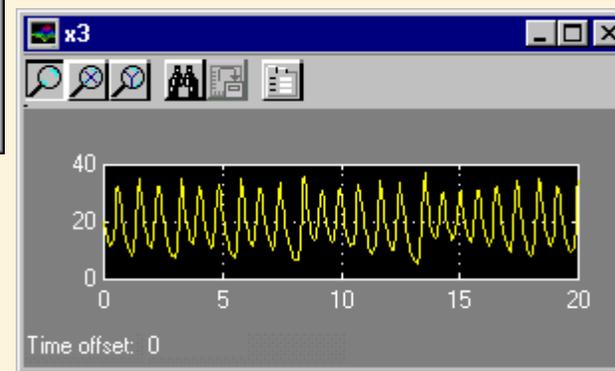
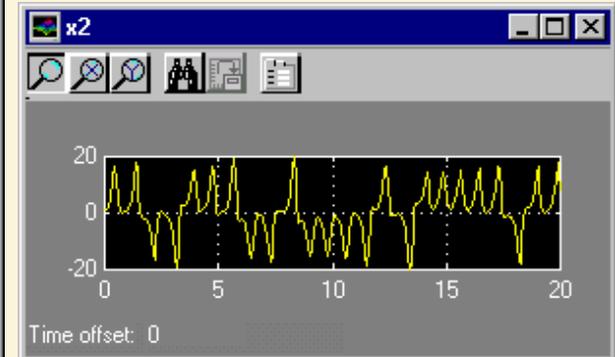
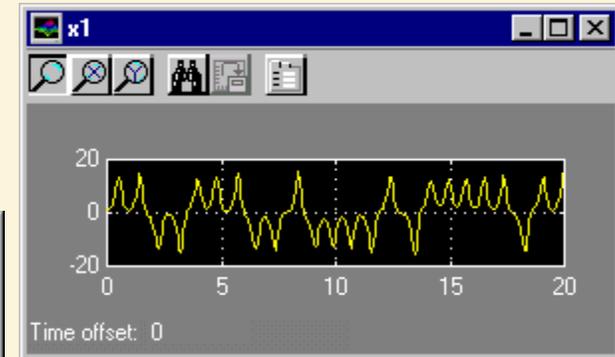
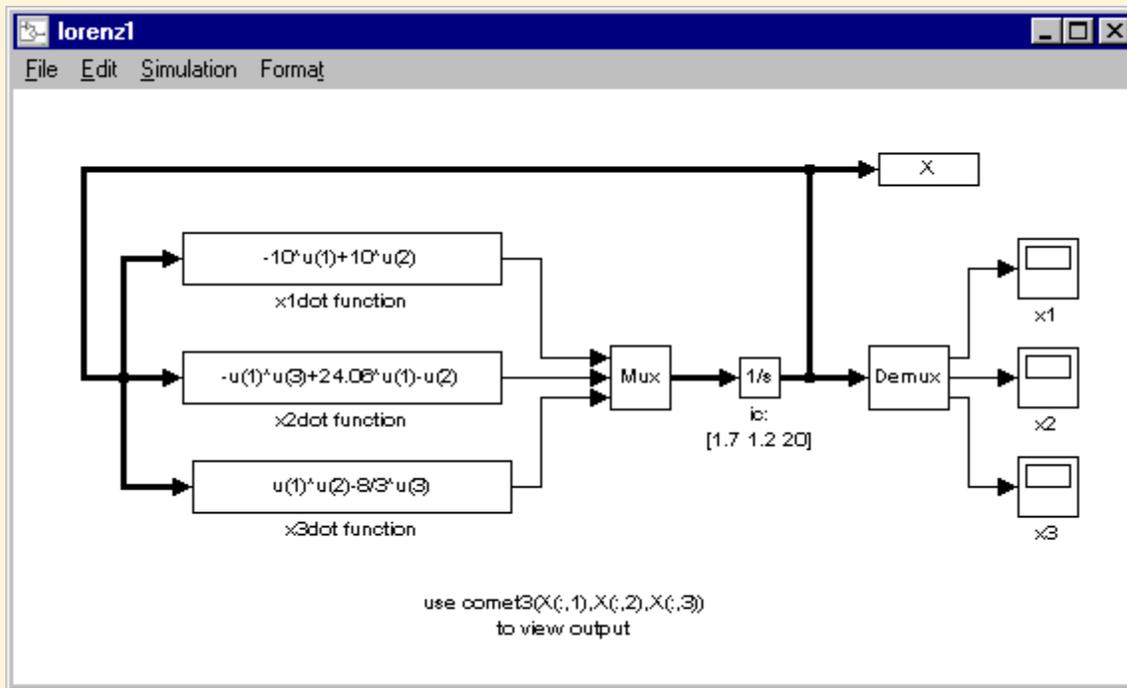
$$\rho = 24.06$$

- 1) Build two different representations of the equations above and simulate the models for 25 seconds
- 2) Use different integrators to show what happens to the solution x_1 , x_2 , x_3 . Remember...the Lorenz Attractor equations represent a chaotic system.

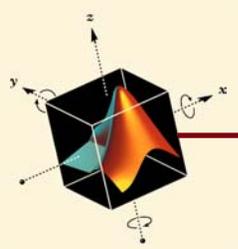
Bonus: Send the output (x_1, x_2, x_3) to the workspace and use the `comet3` command to view the trajectory.



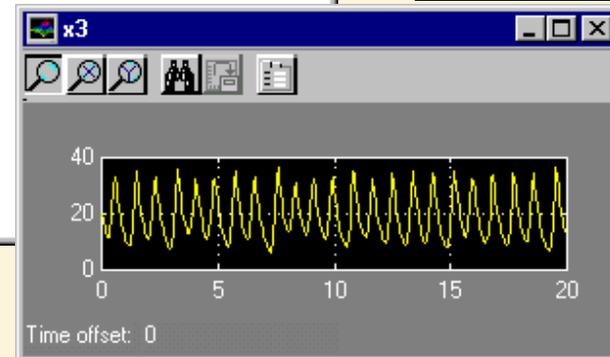
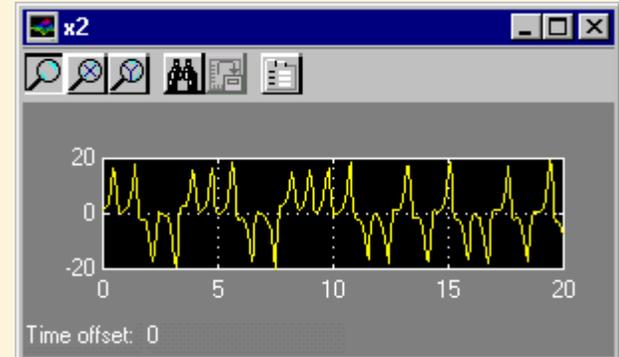
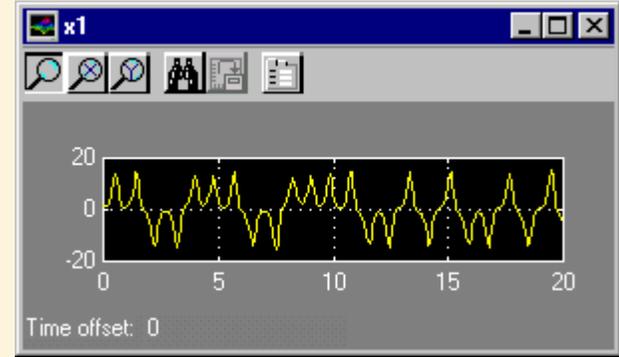
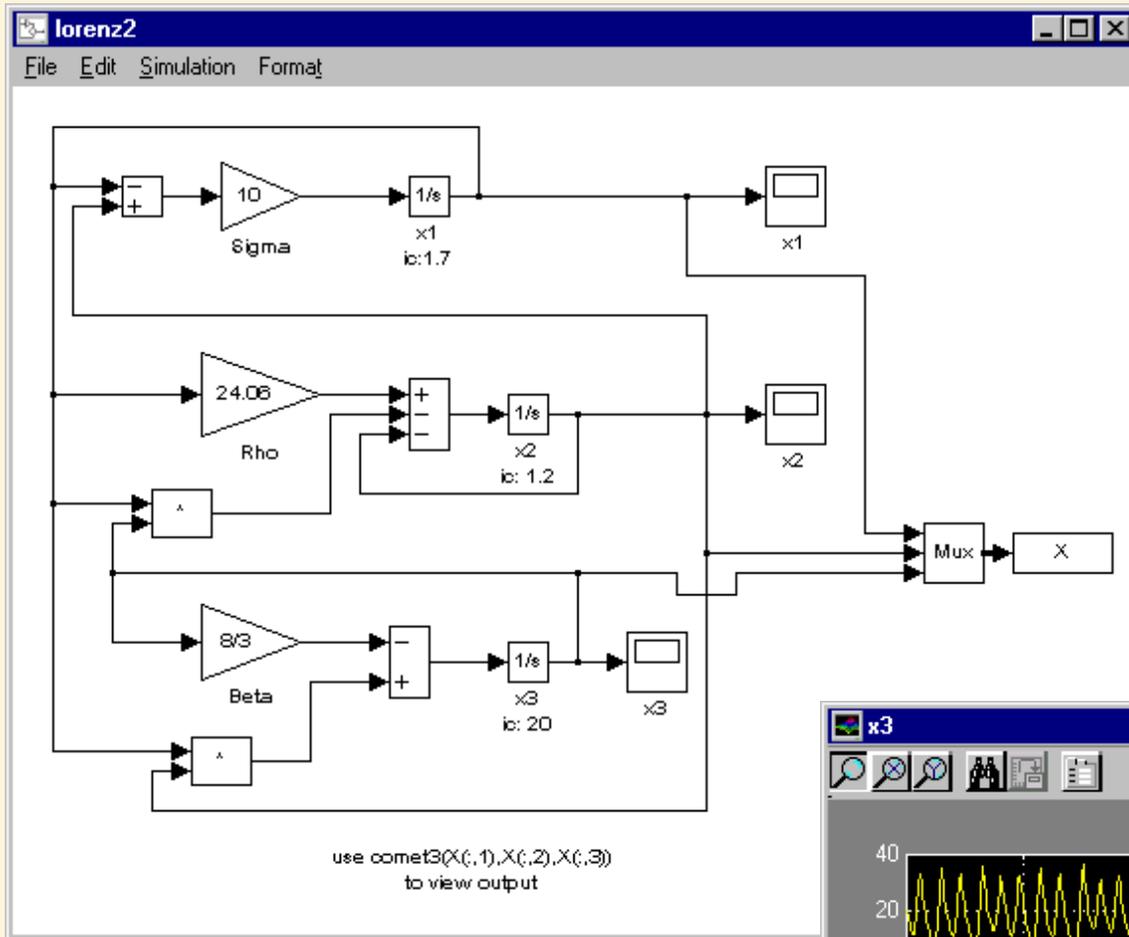
Solution: Lorenz Attractor



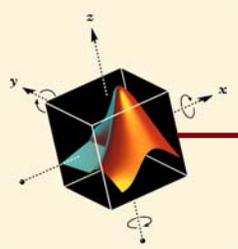
» `lorenz1`



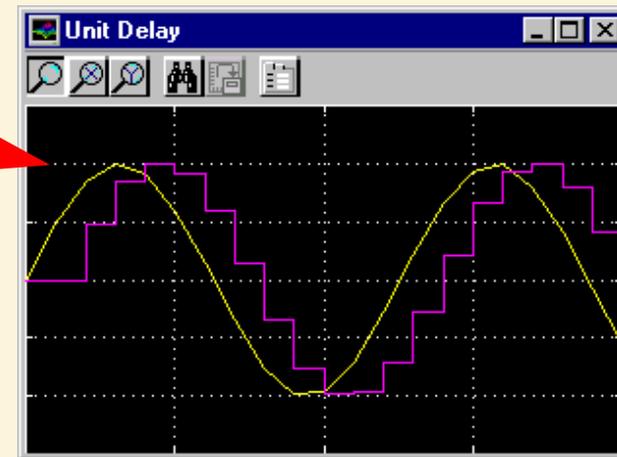
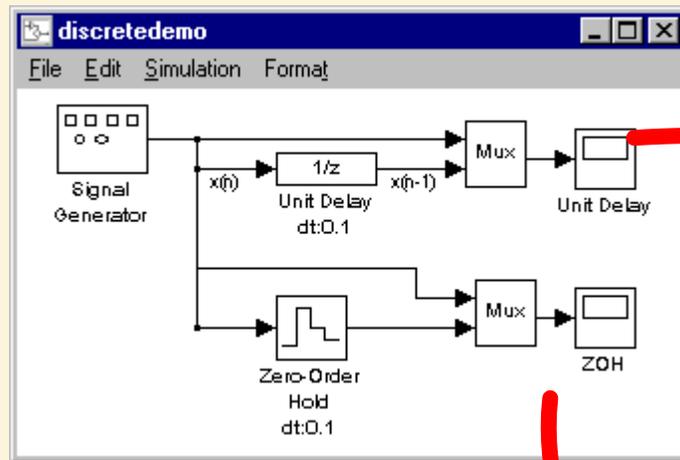
Additional Solution: Lorenz Attractor



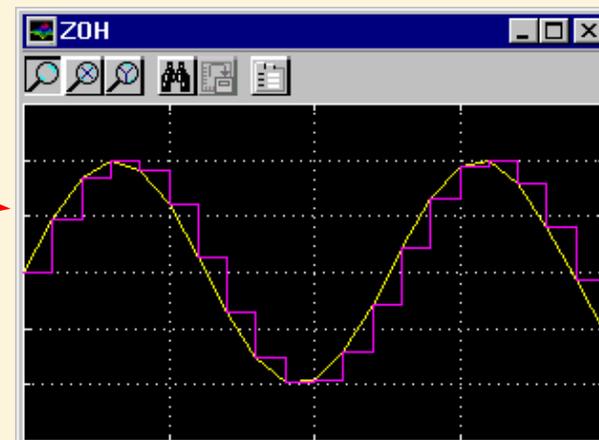
» **lorenz2**



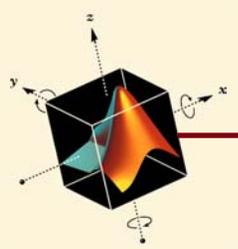
Discrete models



For discrete models, use a fixed-step integrator



» discretedemo

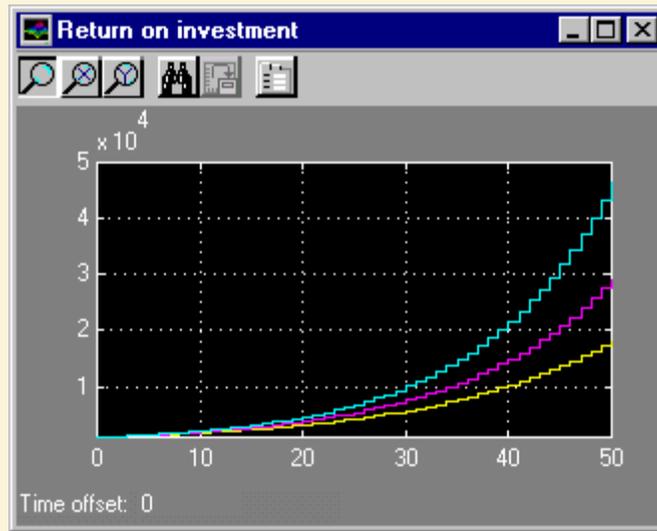
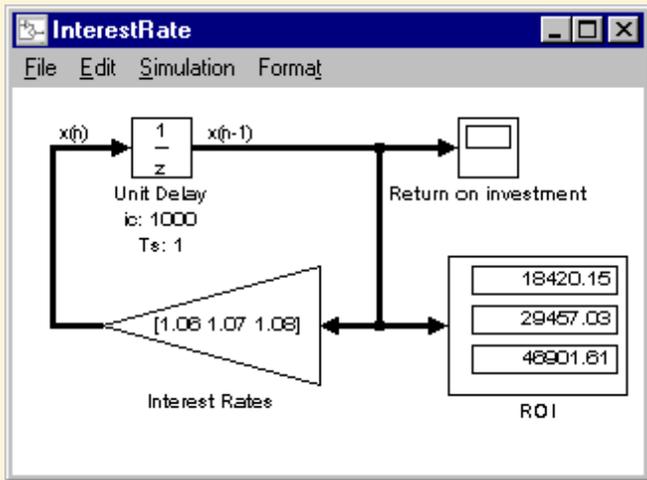


An interest rate example

Let's look at what happens if you gain 6%, 7% and 8% interest on a \$1000 investment over 50 years

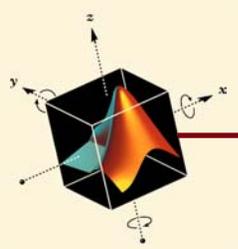
$$M_n = 1.06 M_{n-1}$$

$$M_0 = 1000$$



WOW!!!
That's a
\$28,000
difference!

» InterestRate



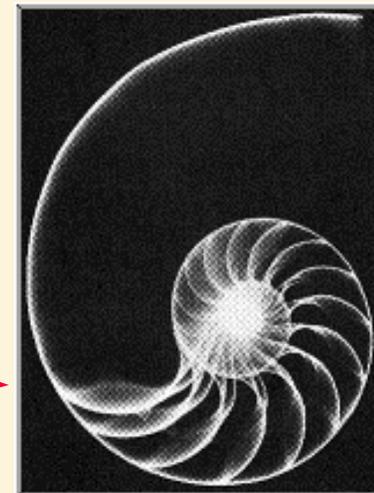
Exercise: Fibonacci

Generate the Fibonacci sequence with **SIMULINK** using unit delay, summing junction, and to workspace blocks, create a system that will generate the first 20 numbers in this series.

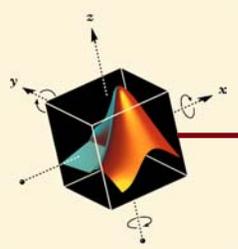
$$x_n = x_{n-1} + x_{n-2}$$

$$x_1 = 1, x_2 = 1$$

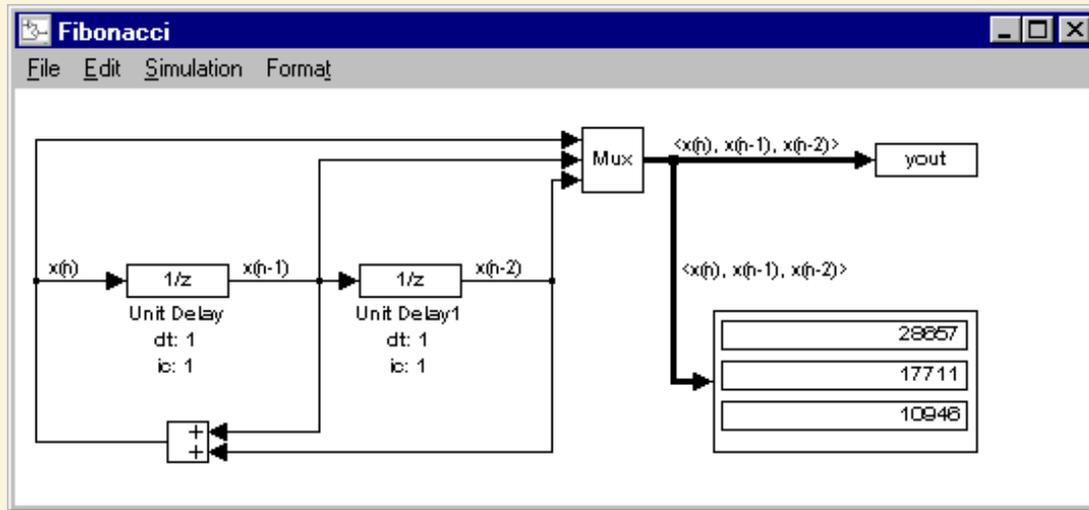
A chambered nautilus is really an example of the golden rectangle which is the Fibonacci sequence.



Display the results by sending the entire series to the **MATLAB** workspace and by viewing the last three data points in the series from within the model.

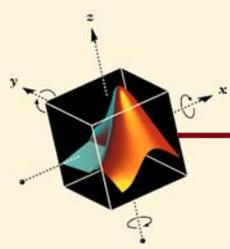


Solution: Fibonacci



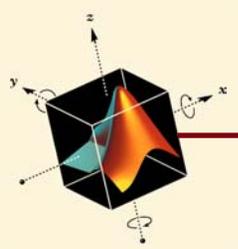
```
MATLAB Command Window
File Edit Window Help
>> yout
yout =
     2     1     1
     3     2     1
     5     3     2
     8     5     3
    13     8     5
    21    13     8
    34    21    13
    55    34    21
    89    55    34
   144    89    55
   233   144    89
   377   233   144
   610   377   233
   987   610   377
  1597   987   610
  2584  1597   987
  4181  2584  1597
  6765  4181  2584
 10946  6765  4181
 17711 10946  6765
 28657 17711 10946
```

» fibonacci



Exercise (Bonus): Fibonacci

- ◆ **Copy all of the blocks in the previous solution so that you have two of the same solution for the series in one model.**
- ◆ **Use a sample time of 1 for all of the blocks in the first series and a sample time of 2 for all of the blocks in the second series. Rerun the model and look at what happens using different integrators (i.e. fixed step and variable step).**
- ◆ **Use a sample time of 1 for all of the blocks in the first series and a sample time of 0.75 for all of the blocks in the second series. Rerun the model and look what happens using different integrators.**



Exercise: MATLAB Markov market model (MMMM)

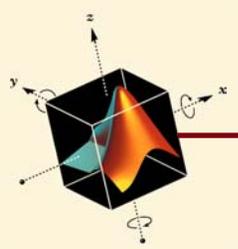
- Every year 1% of the people who don't currently use MATLAB see a marketing piece and then start using MATLAB
- BUT every year 5% of the people who currently use MATLAB win the lottery and no longer need to use it
- There are 5 billion people in the world
- There are 400,000 current MATLAB users
- What happens in 20 years? 50 years?

$$M_n = 0.95 M_{n-1} + 0.01 N_{n-1}$$

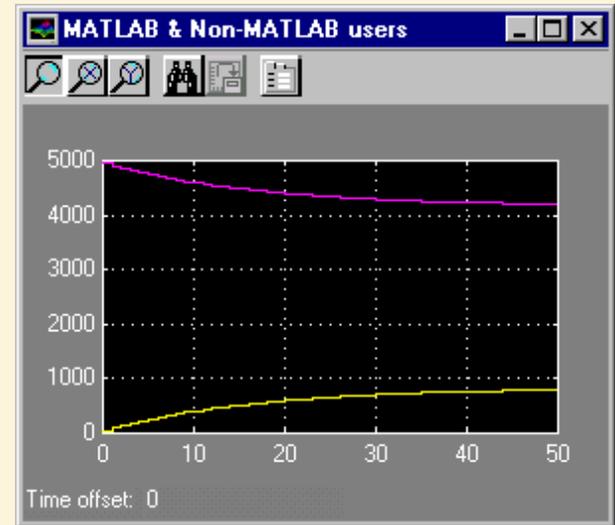
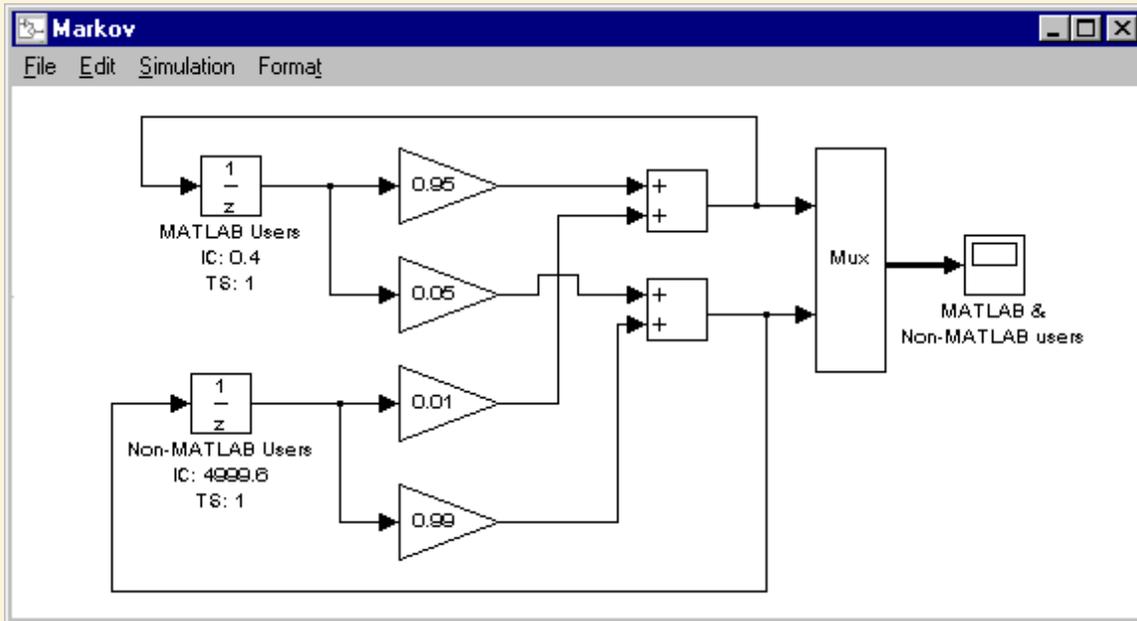
$$N_n = 0.05 M_{n-1} + 0.99 N_{n-1}$$

$$M_0 = 0.4$$

$$N_0 = 4999.6$$

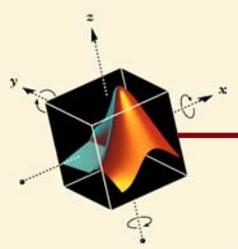


Solution: MATLAB Markov market model

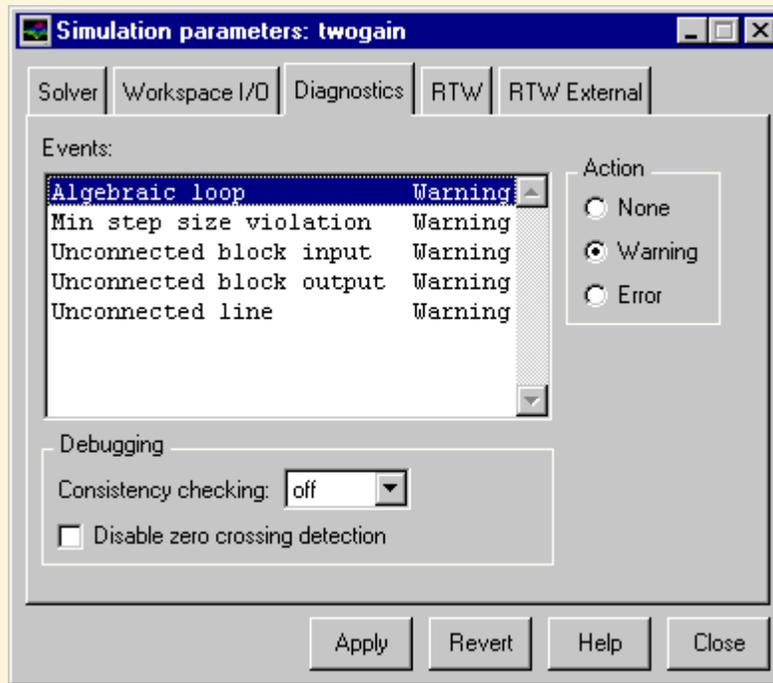


No matter what the initial number of MATLAB users is (assuming it's a positive number), the steady state is always the same.

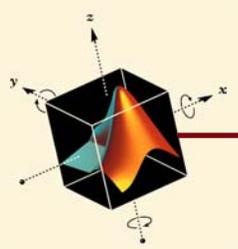
» Markov



Controlling Model Diagnostics

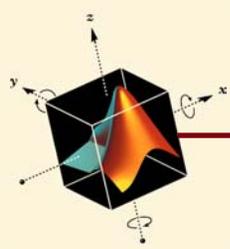


Control warnings and errors



From the Command Line

- ◆ **Why simulate from the command line?**
 - ◆ Automate repeated simulations
 - ◆ Parameter tuning
 - ◆ Any number of other reasons...



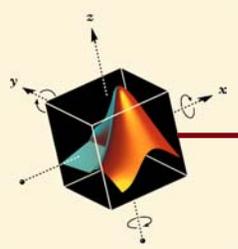
How Simulink Works

Initialization

- Block parameters are passed to MATLAB for evaluation.
- The model hierarchy is flattened. (Conditionally executed subsystems are seen as atomic units)
- Blocks are sorted in the order they need to be updated. (Algebraic loops are detected)
- Block connectivity is checked to make sure that input and output vector lengths match.

Simulation

- Numerical Integration
- State Events
- Implicit Equations
- and more...



Zero Crossings



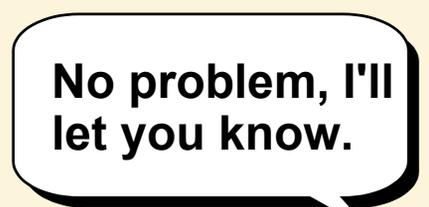
Solver

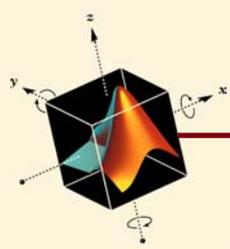
System



Solver

System





Zero Crossings

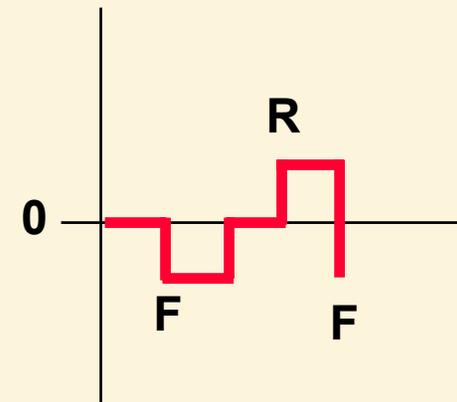
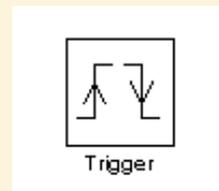
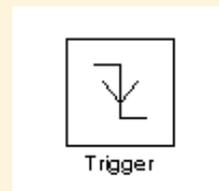
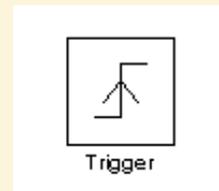
- **Blocks with Zero Crossing Support:**

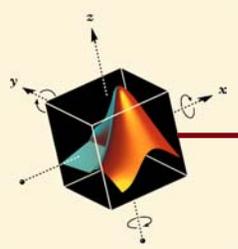
Abs	Backlash	Dead Zone	Hit Crossing
Integrator	MinMax	Relay	Relational Operator
Saturation	Sign	Subsystem Switch	

- **Each block defines their own zero crossings**

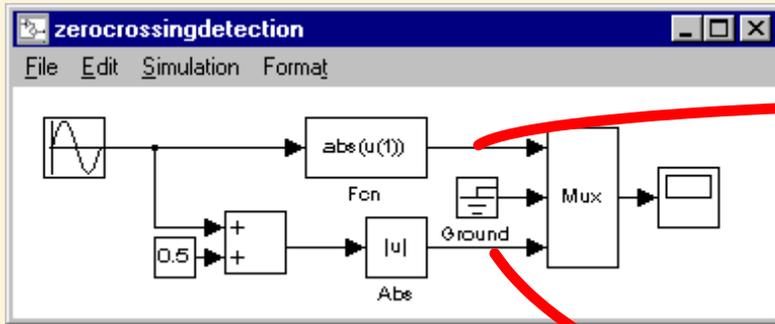
- **Zero Crossings may be**

- ◆ Rising - signal rises to or through zero and becomes positive
- ◆ Falling - signal falls to or through zero and becomes negative
- ◆ Either - either the rising or falling condition occurs

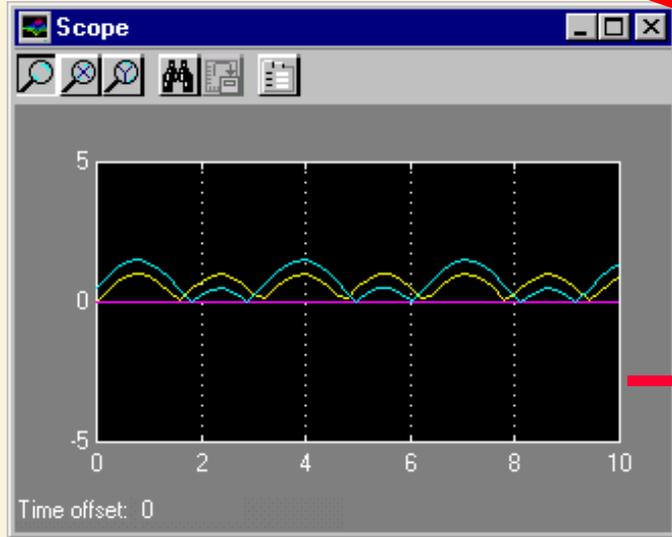
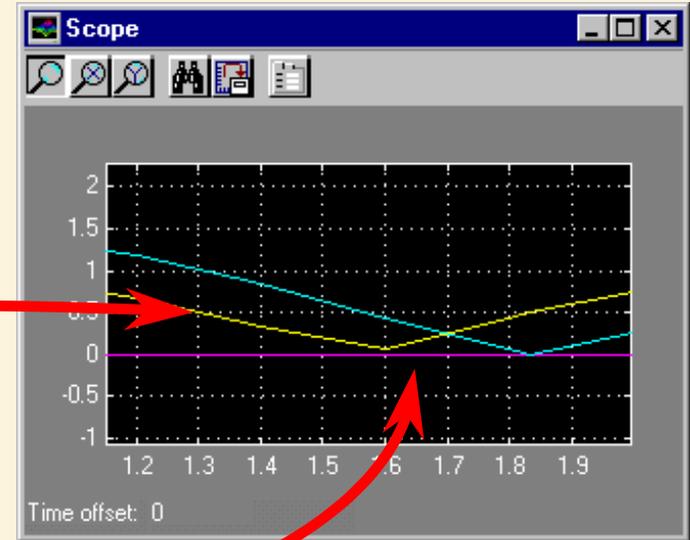




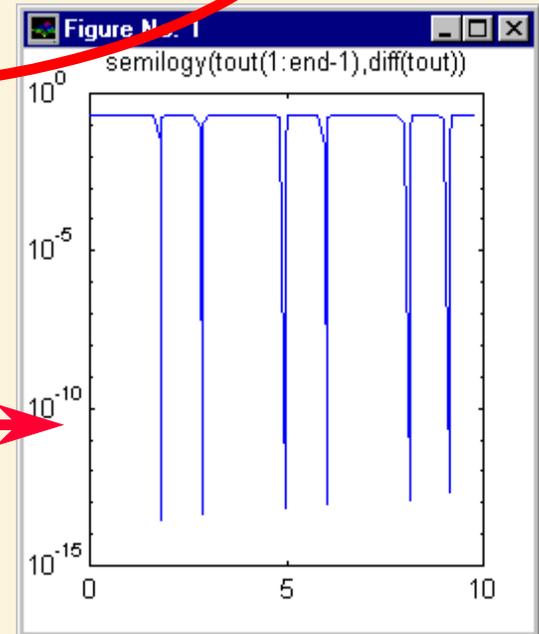
Zero Crossings



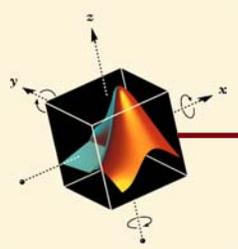
No zero crossing detection



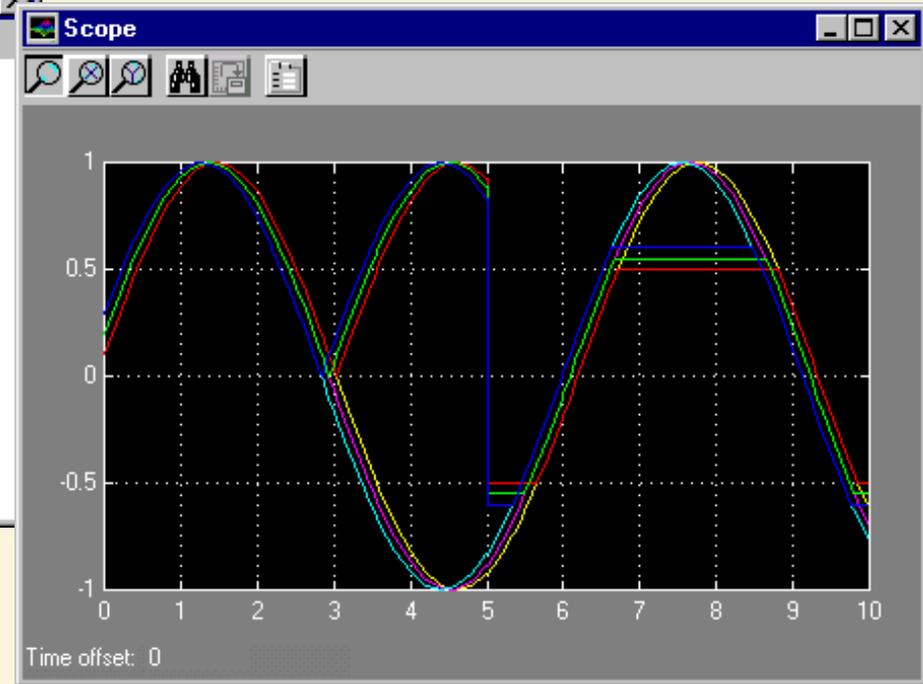
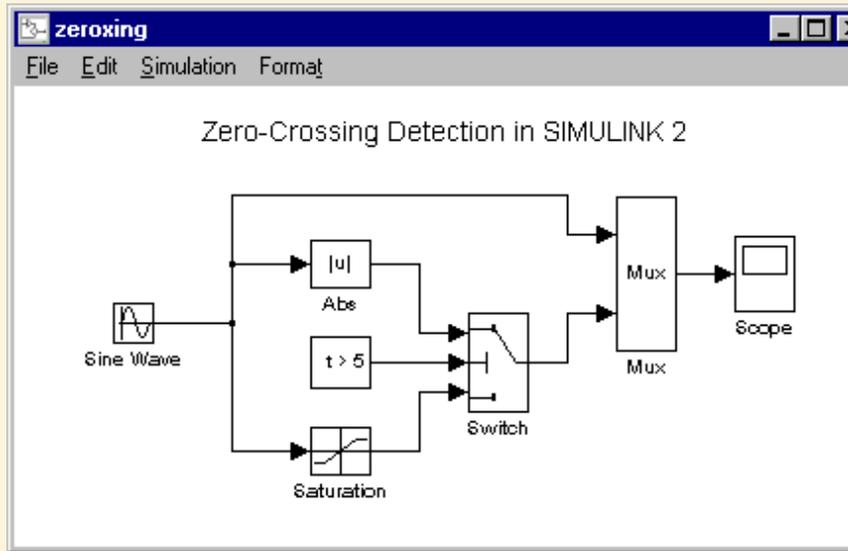
The zero crossing is detected

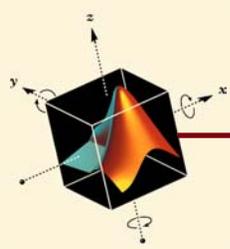


» zerocrossingdetection



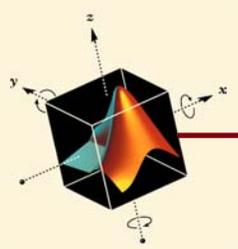
Zero Crossings





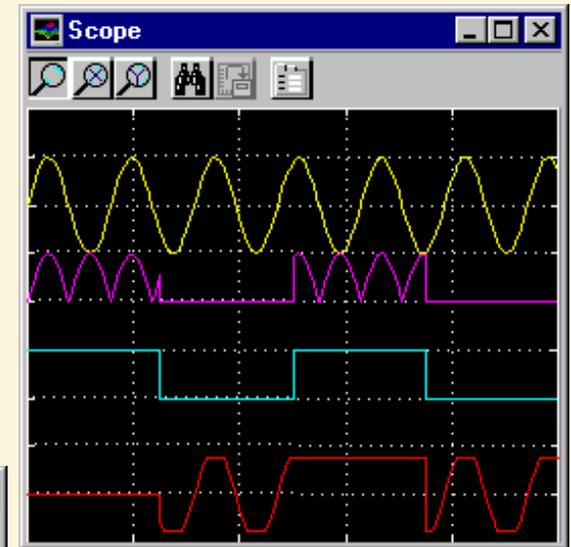
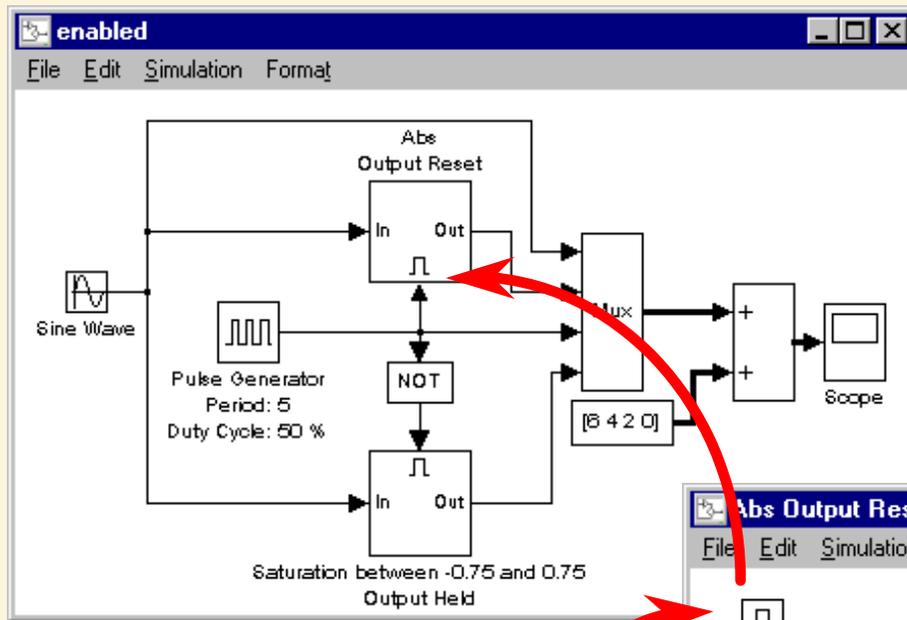
Zero Crossings (caveats)

- **Although discrete blocks causes discontinuities, they do not require event detection because the solver knows a-priori where the events occur.**
- **It is possible to create models that chatter**
 - **Usually these models are not physically realizable**
 - a massless spring
 - a pneumatic system with zero time delay
 - **Chattering causes repeated detection of zero crossings which means the step size becomes very small**
- **Use the Disable zero crossing detection check box on the Parameters Dialog Diagnostics page to disable the zero crossing detection**
 - **The Hit Crossing block is unaffected by this**
 - **Simulation results may be less accurate**

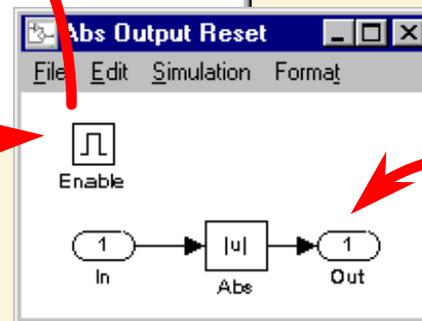


Enabled Subsystems

An enabled subsystem is **conditionally executed** at each simulation step for which the control signal to the subsystem is positive.

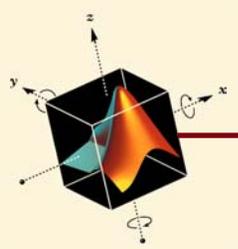


States may be held or reset



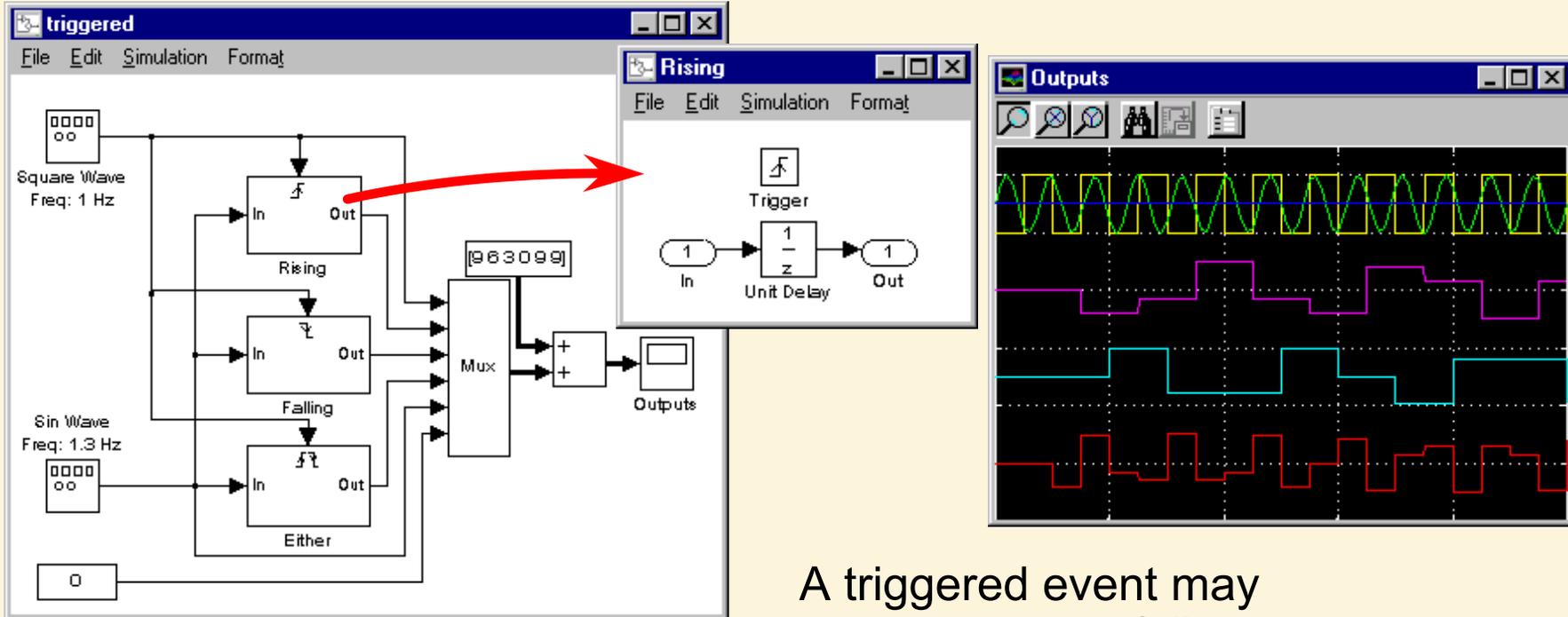
Outputs may be held or reset

- » enabled
- » clutch



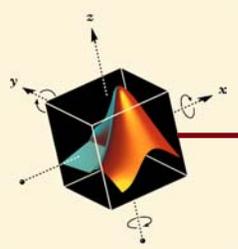
Triggered Subsystems

A triggered subsystem executes each time a trigger event occurs. The execution is for one time step only.



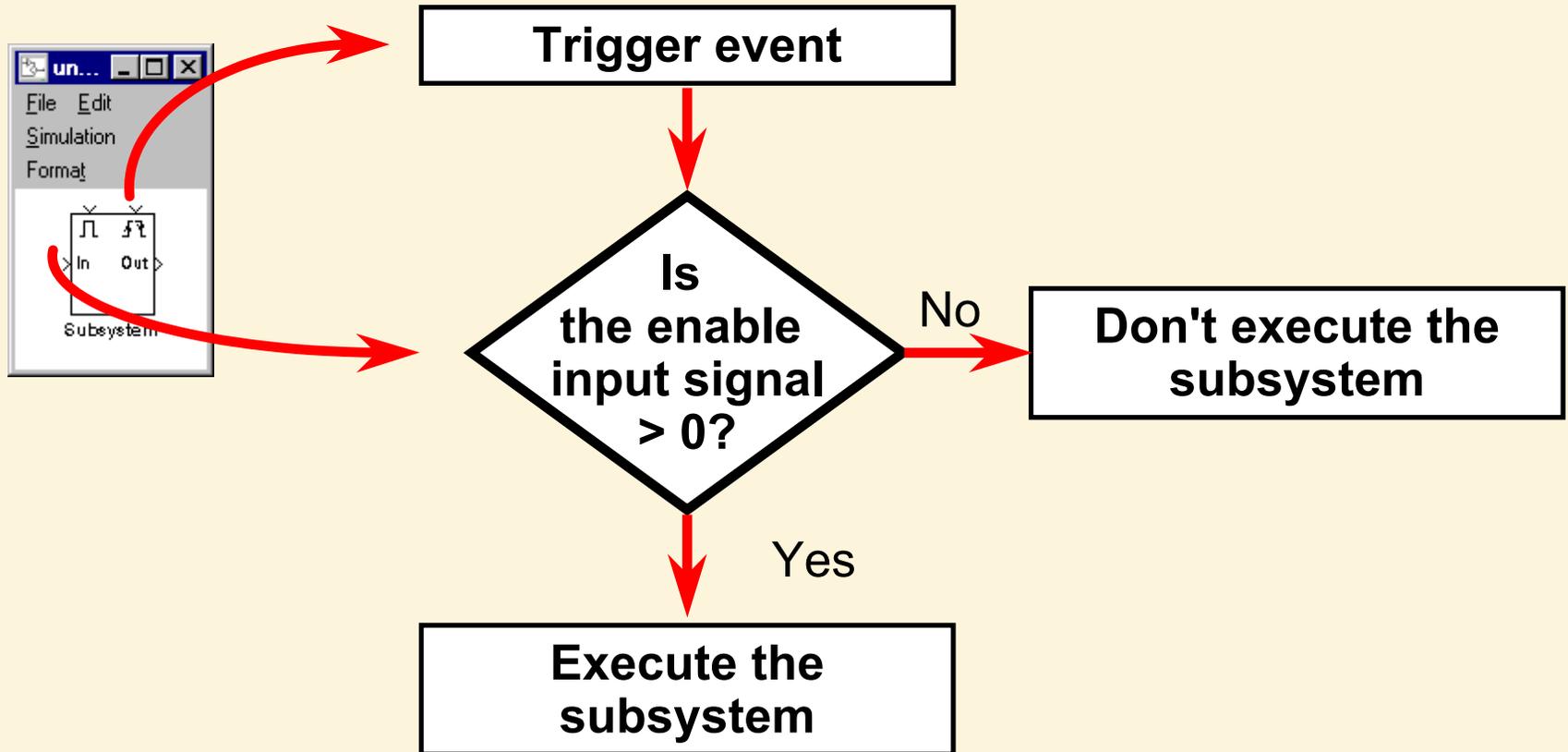
A triggered event may occur on rising, falling or both rising and falling edges of signal.

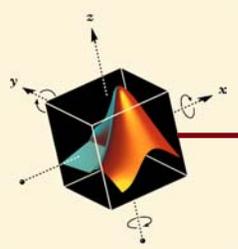
» triggered



Triggered and Enabled Subsystems

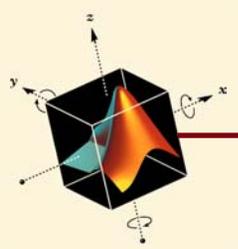
A triggered subsystem executes each time a trigger event occurs. The execution is for one time step only.





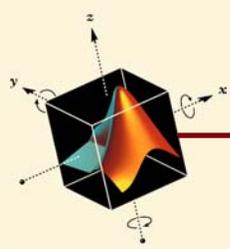
Improving Performance and Accuracy

- ◆ **Inappropriate error tolerances (orbit)**
- ◆ **Inappropriate maximum step size**
 - ◆ Systems with periodicity (periodic)
- ◆ **Inappropriate integration algorithm**
 - ◆ Stiff systems (stiffdemo, vdp with gain=1000)
 - ◆ Multirate systems (solution to bonus exercise with discrete model)
- ◆ **Algebraic loops (alglloop1, alglloop2)**
- ◆ **Bad numerical scaling (numerical_scaling)**
- ◆ **Unstable systems**



Making Models Run Fast

- ◆ **Avoid MATLAB Fcn blocks, since they invoke the MATLAB parser.**
- ◆ **For the same reason, avoid M-File S-function blocks if at all possible.**
- ◆ **Try simulating from the command line.**
- ◆ **Close all scopes to avoid the computational overhead associated with graphics.**
- ◆ **Experiment to make sure you're using the fastest reasonable algorithm with the biggest reasonable step size.**
- ◆ **Never send a random number block into an integrator; always use the band-limited white noise block instead.**



Recap

- **Learn your way around the libraries provided with Simulink**
- **Keep blocks you use frequently in the same place — build your own library**
- **Gather all the blocks you think you need first, then connect them: collect, then connect.**
- **Calibrate your simulation parameters and your scopes.**