

# Introduction to Simulink for Dynamic System Modeling and Simulation

Atilla Dogan

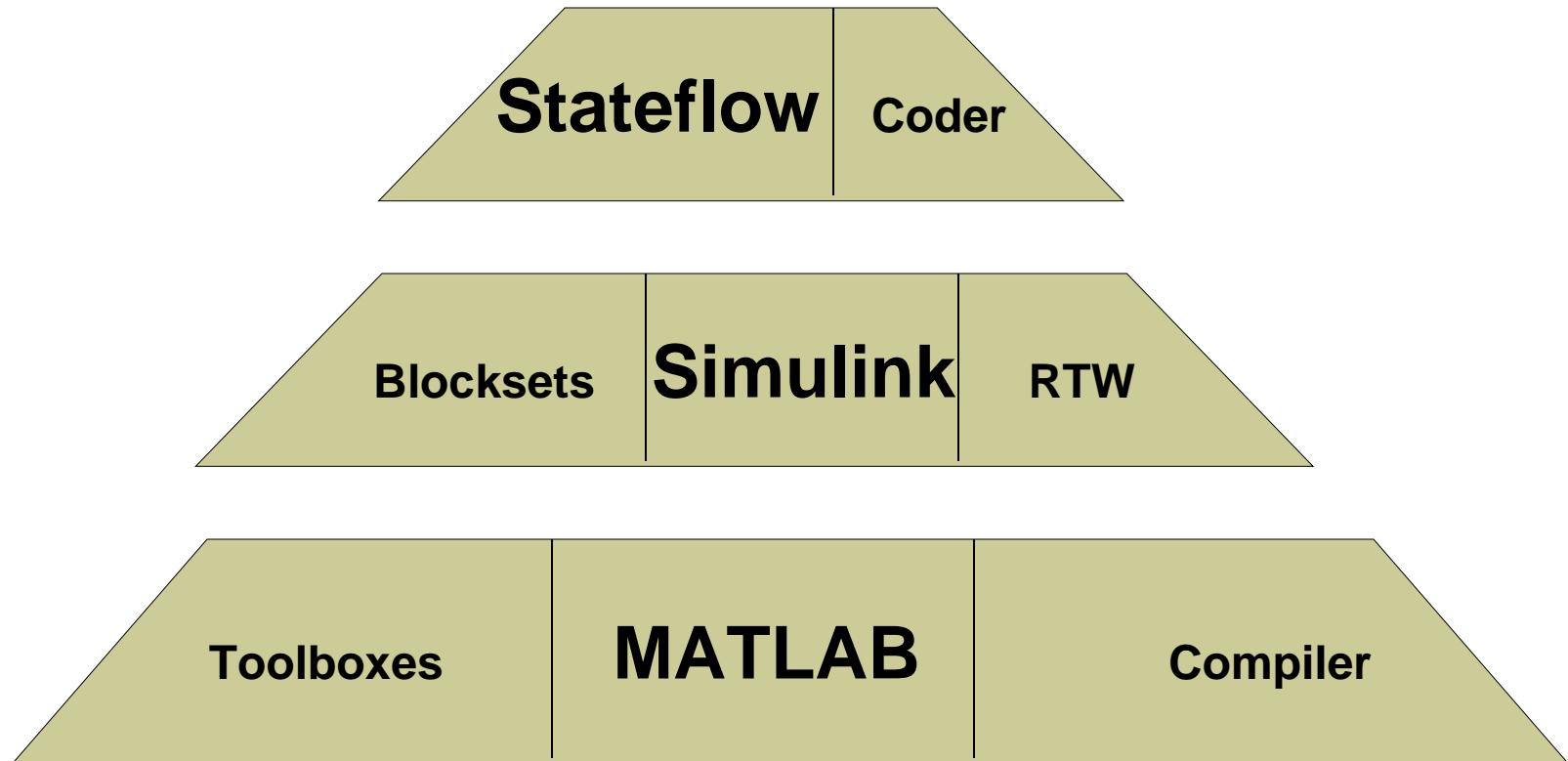
XX4378 and XX5378 – Introduction to UVS

September 13, 2017

Fall 2017

# The MATLAB Product Family

---



# What is Simulink

---

- Simulink is a tool that allows you to model and simulate a wide variety of dynamic systems
- Among other things, this means
  - Block diagram editing
  - Nonlinear Simulation
  - Continuous and discrete simulation
  - Hybrid simulation
  - Asynchronous (non-uniform sampling)
  - Integration with MATLAB, Extensions, Blocksets & Toolboxes

# What can you model with Simulink

---

- Just about anything you can model mathematically
  - Aircraft and spacecraft dynamics and systems
  - Automotive systems
  - Ship systems
  - Communications and satellite systems
  - Monetary systems
  - Biological systems

# What we are going to learn

---

- Building and running models
- Viewing Data
- Masking
- GUI interfaces
- And more

# Building a model

---

- There are 4 steps
  - Collecting and connecting blocks
  - Simulating the system
  - Analyzing the results
  - Refining the model
  
- Each of these steps will be done several times

# Creating a model

---

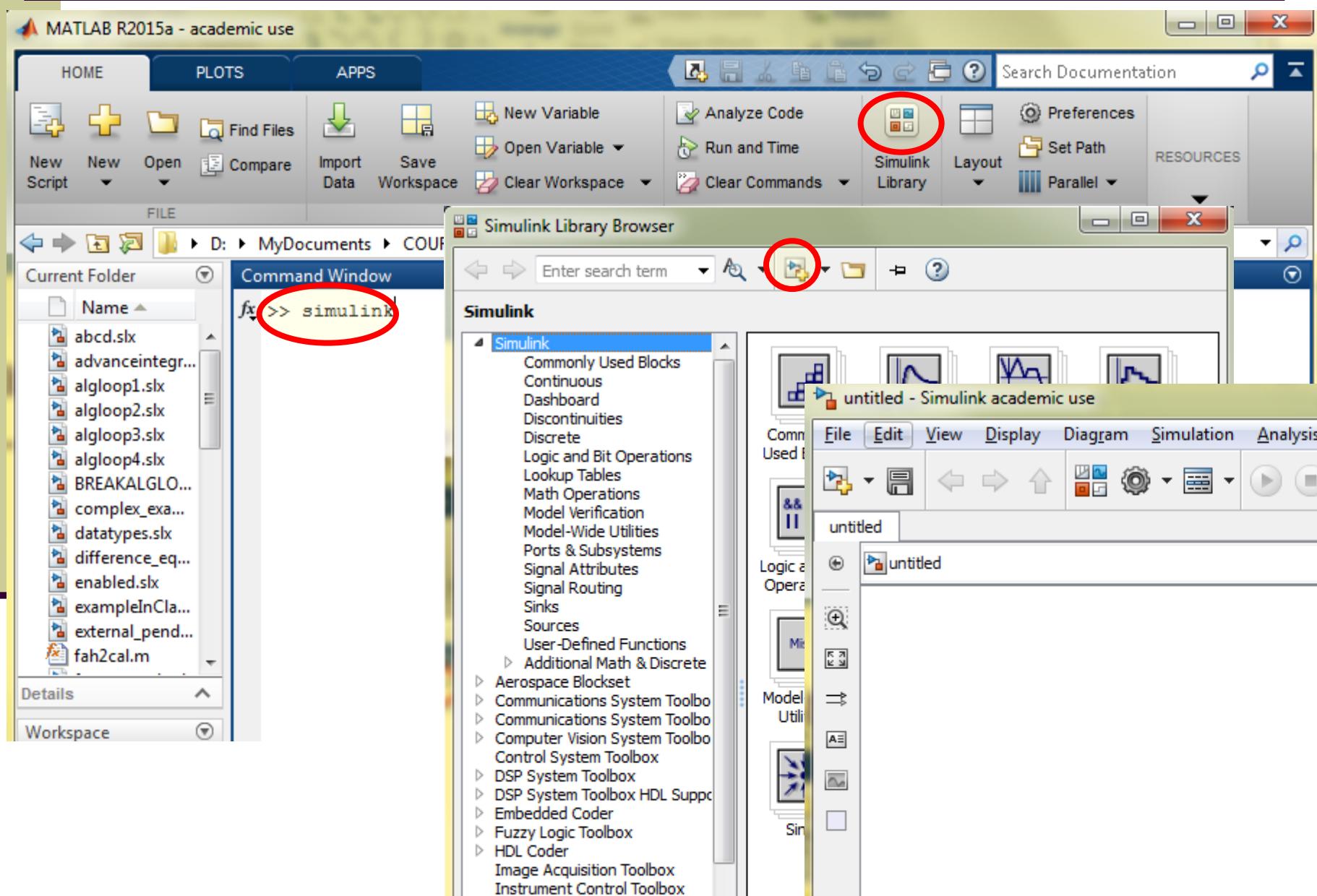
- Simulink basics
- Starting Simulink
- The Simulink block library
- Editing models
- Setting block dialog parameters
- Creating subsystems
- Saving (SLX File format)
- Annotations and signal labels
- Printing
- Getting Help
- Tips for building models

# Simulink basics

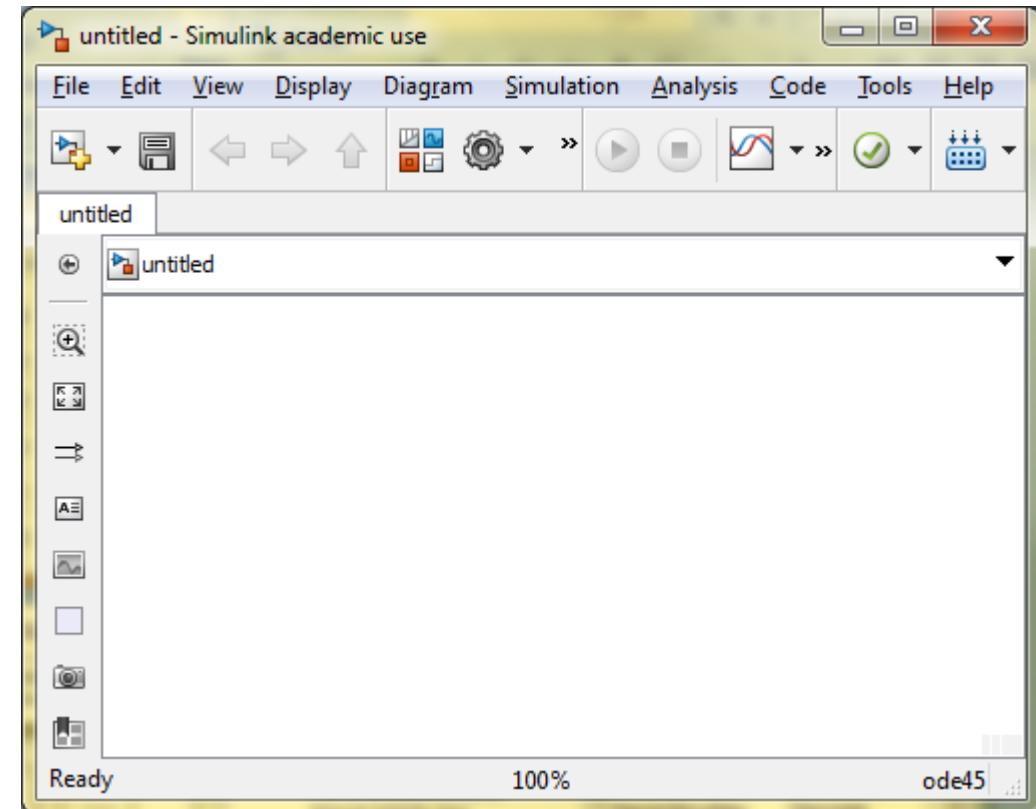
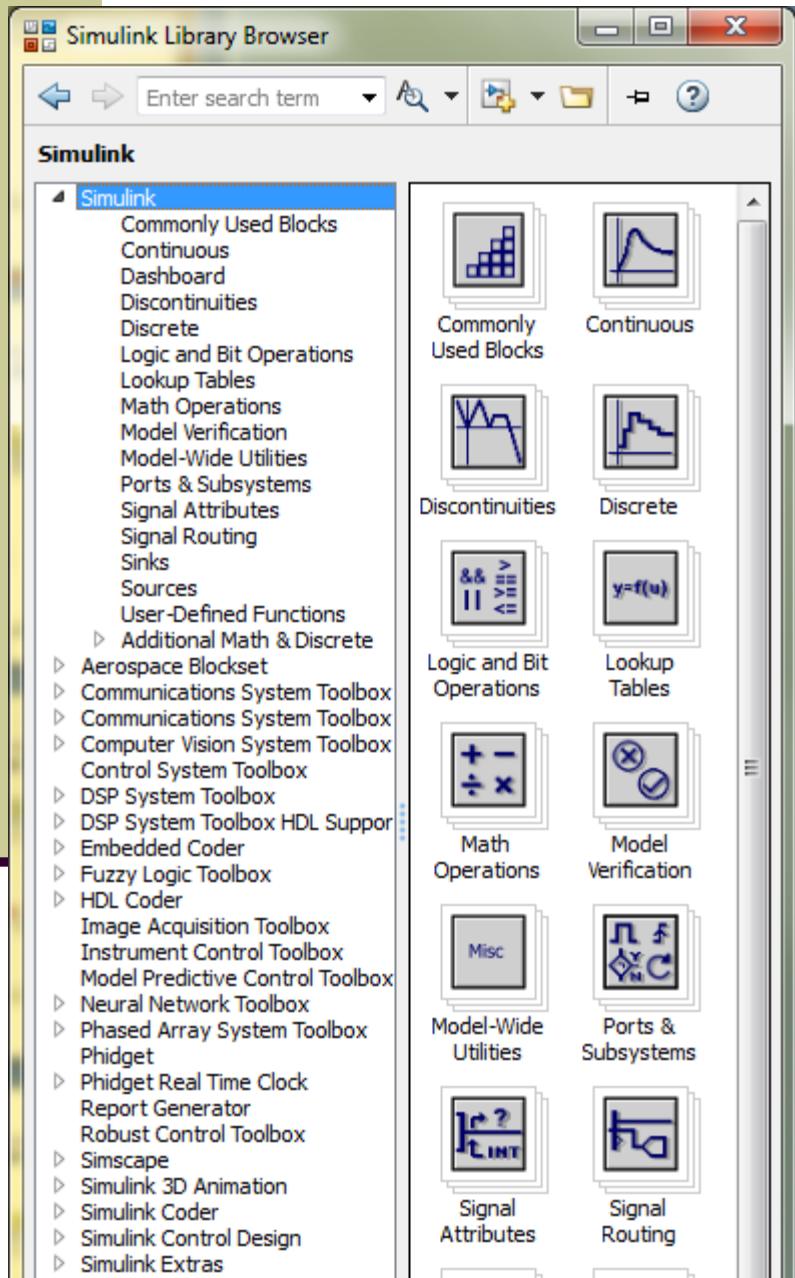
---

- 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

# Starting Simulink

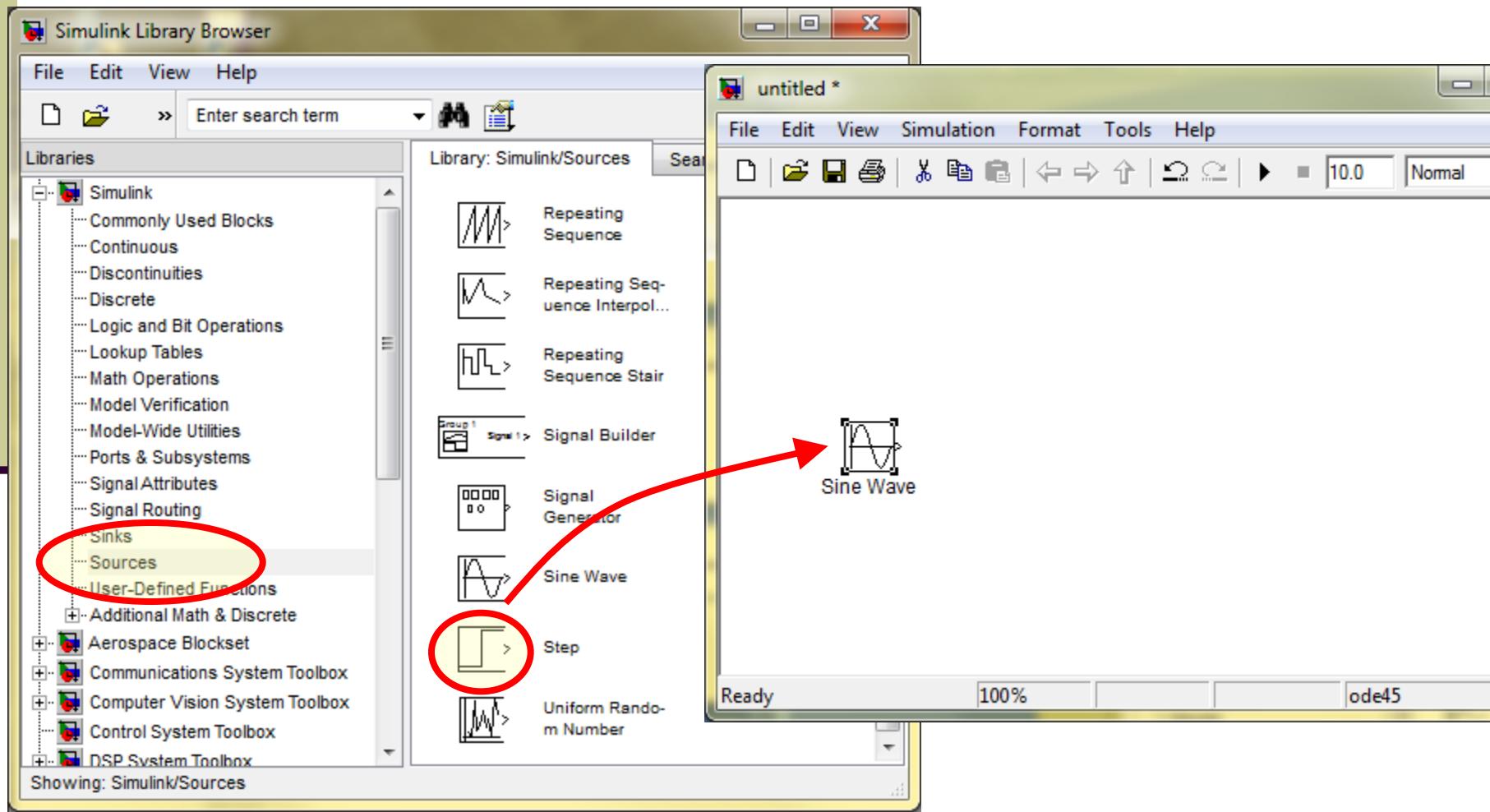


# The Simulink block library



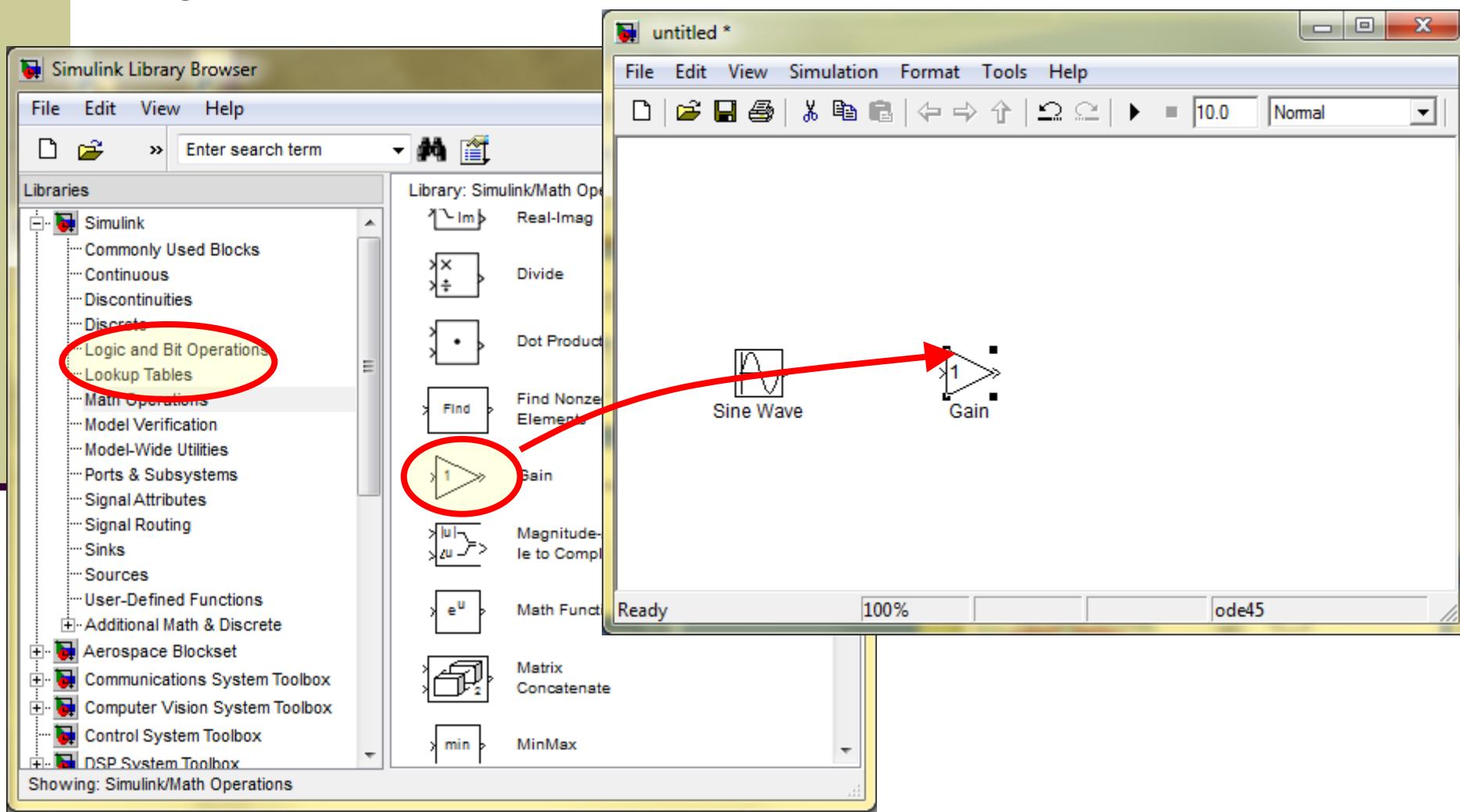
# Collecting blocks

Pull the required blocks into a new Simulink block diagram window



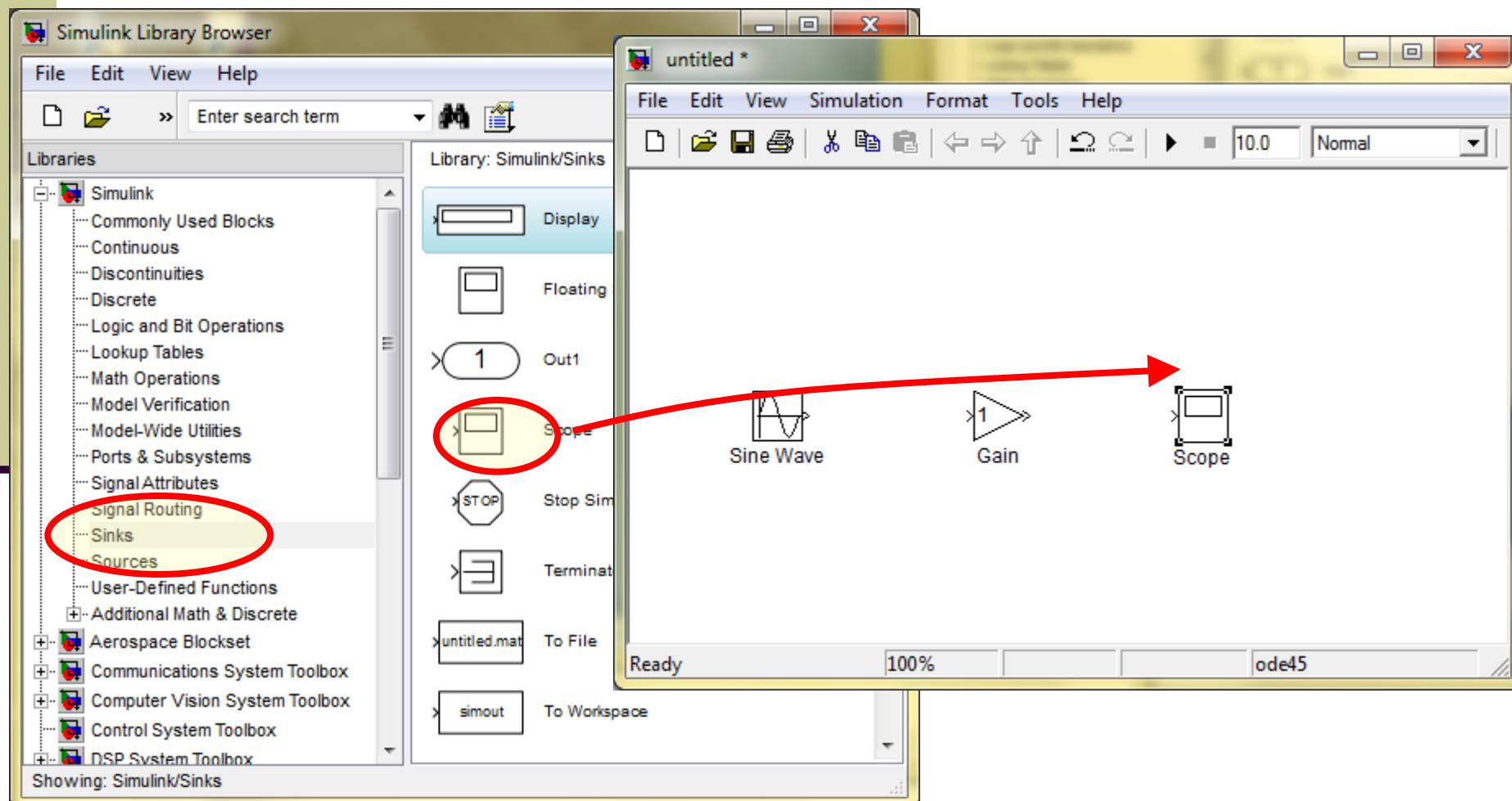
# Collecting blocks

Pull the required blocks into a new Simulink block diagram window



# Collecting blocks

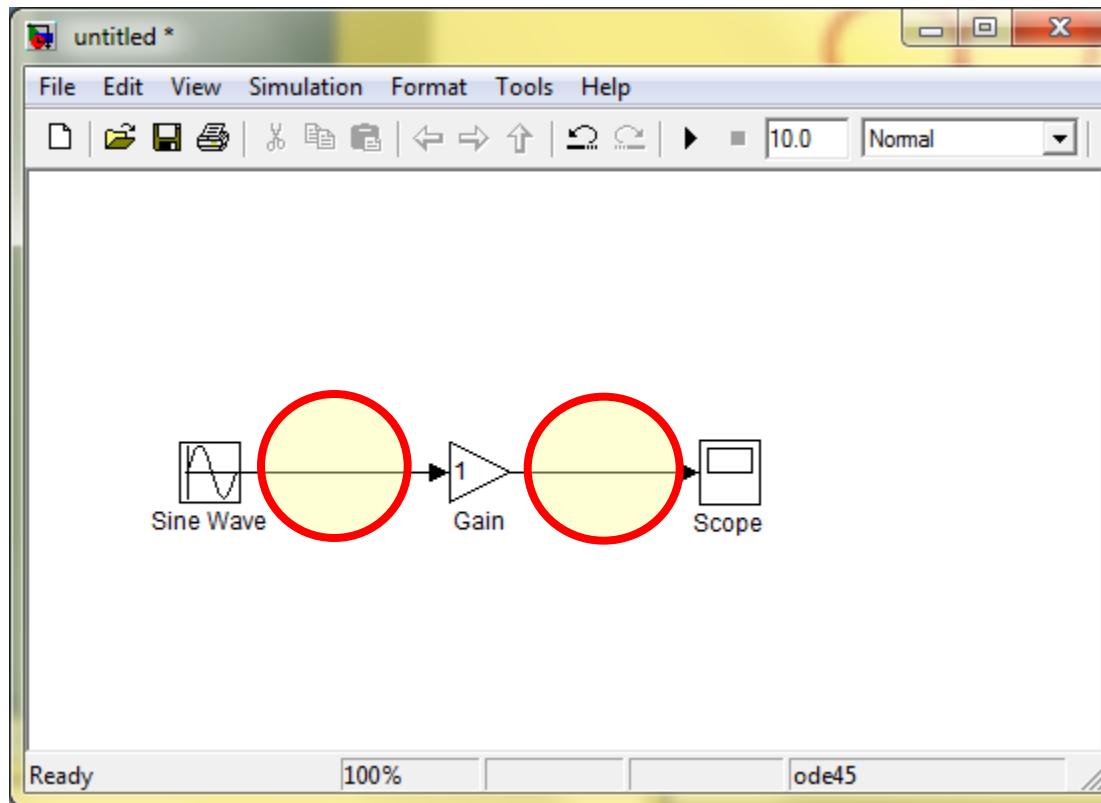
Pull the required blocks into a new Simulink block diagram window



# Connecting blocks

---

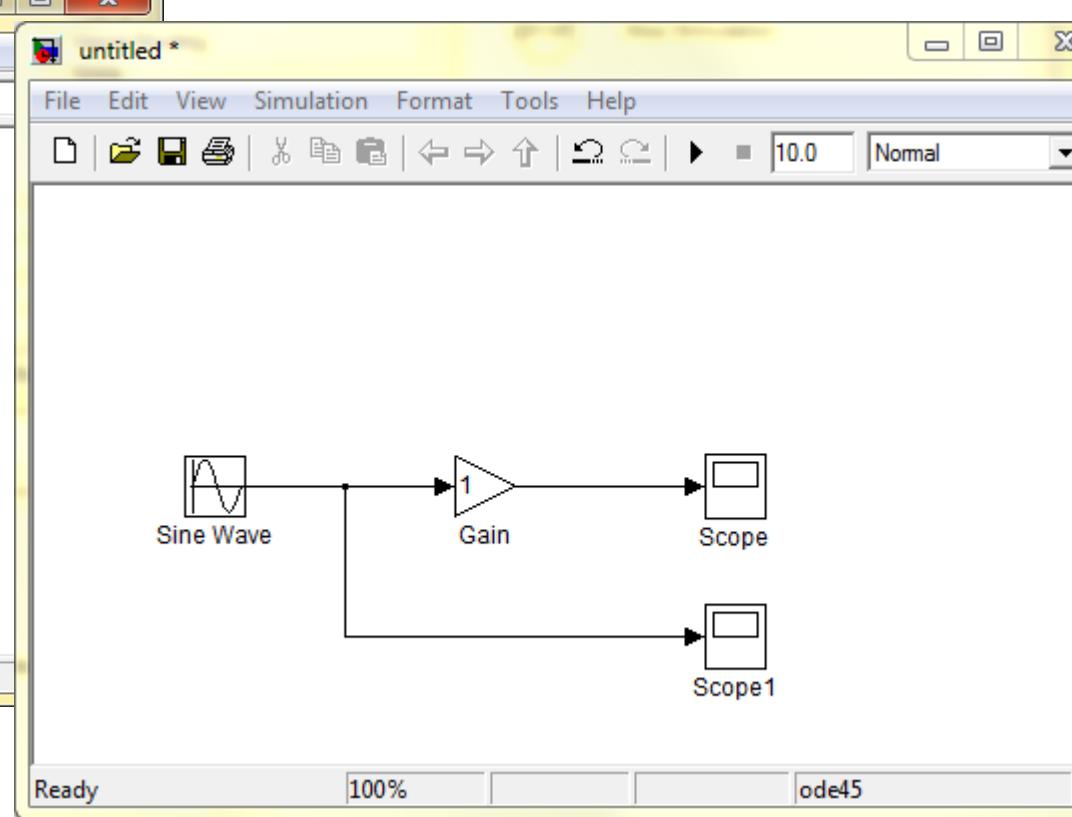
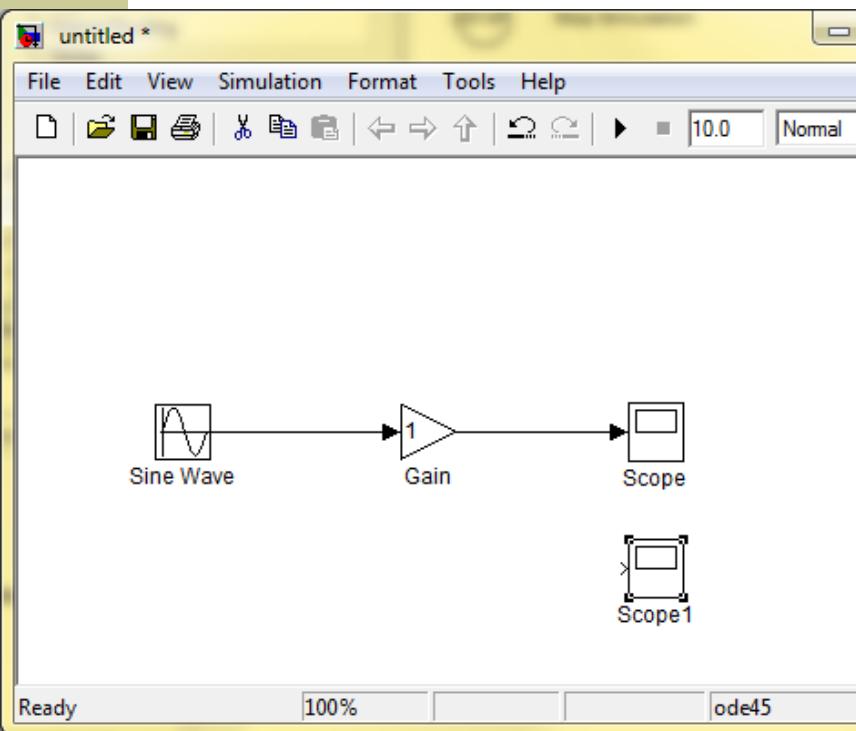
Left Mouse drag from port to add line



# Copying blocks & branching signal

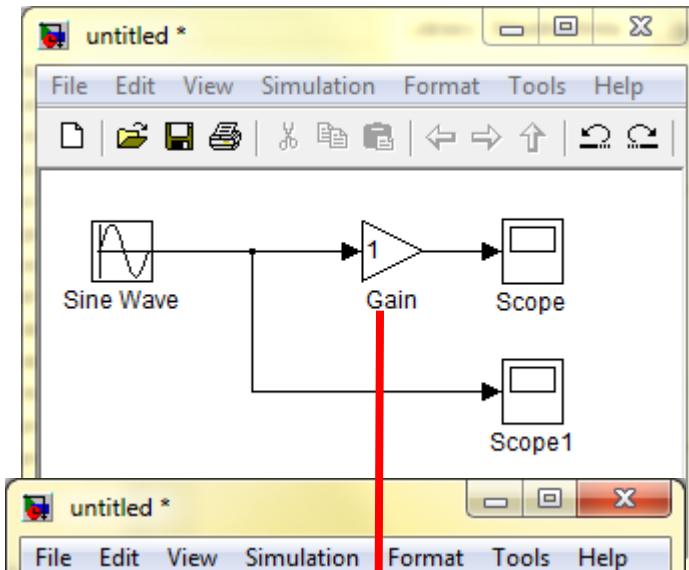
Right Mouse drag  
to copy

Right Mouse drag  
to add branch line

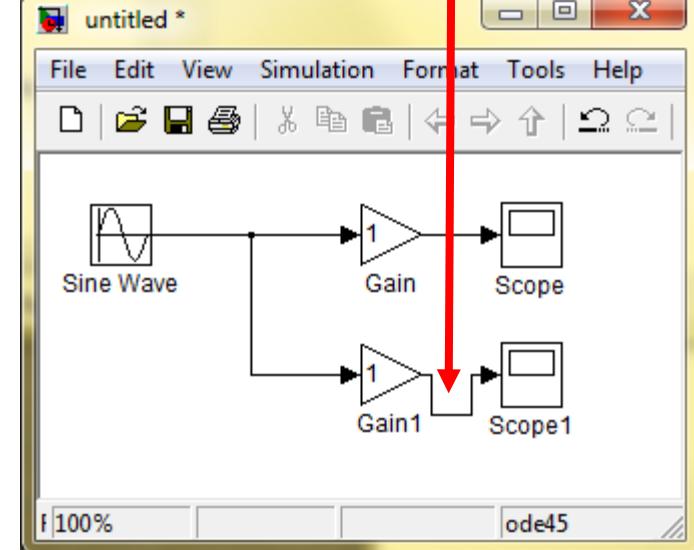
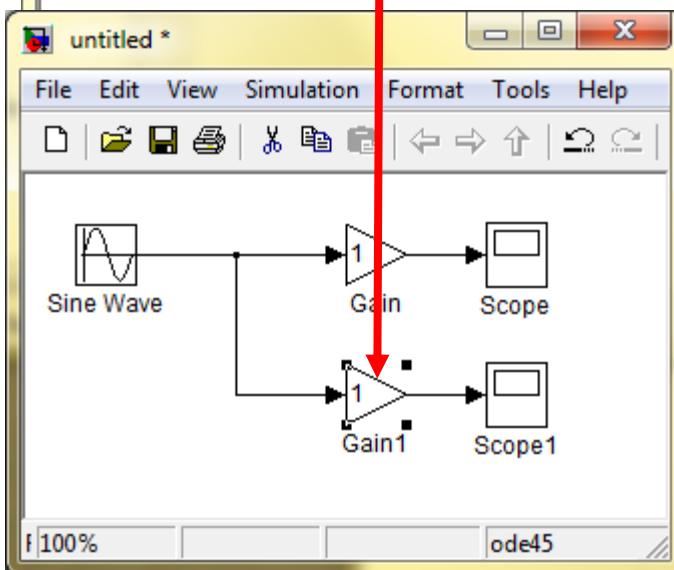
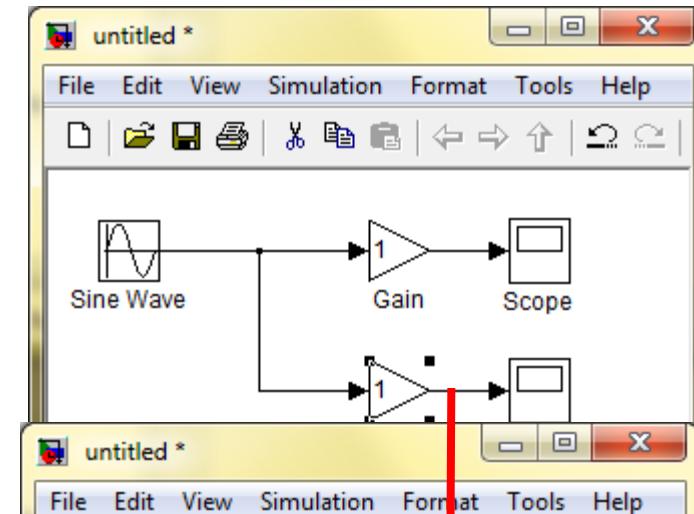


# Inserting Blocks, rerouting lines

Place block over line to insert

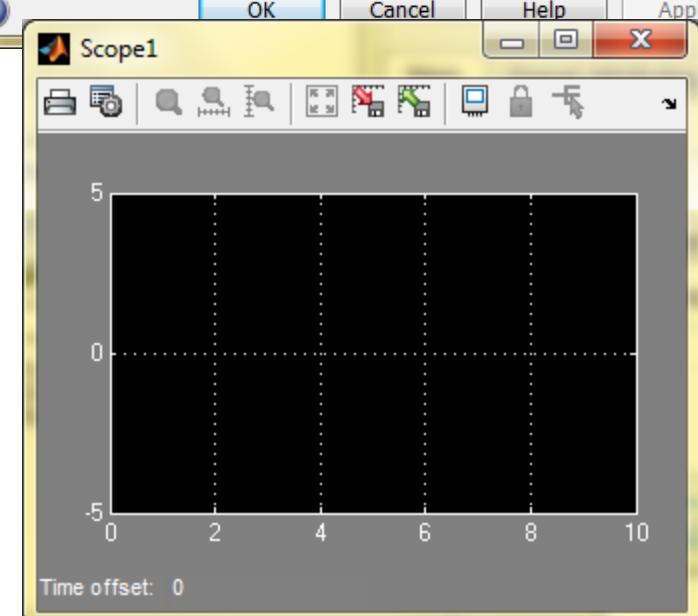
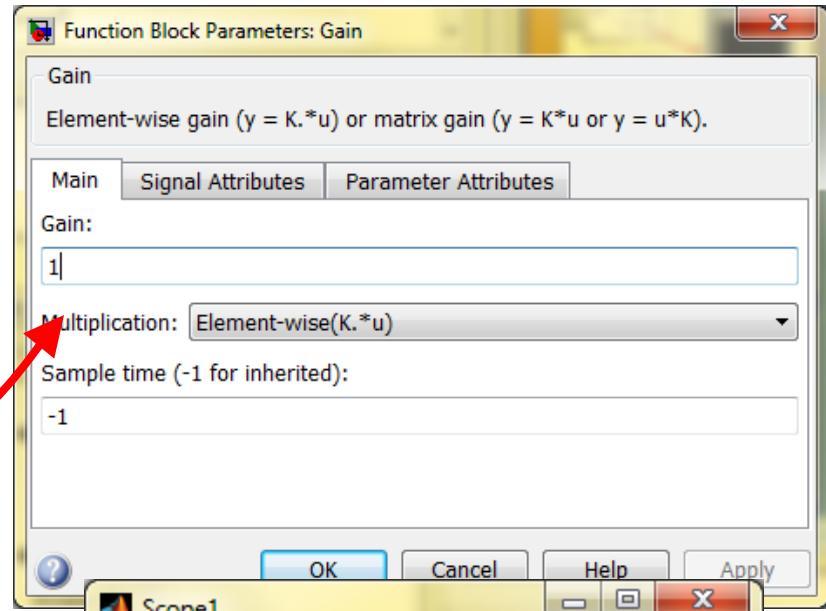
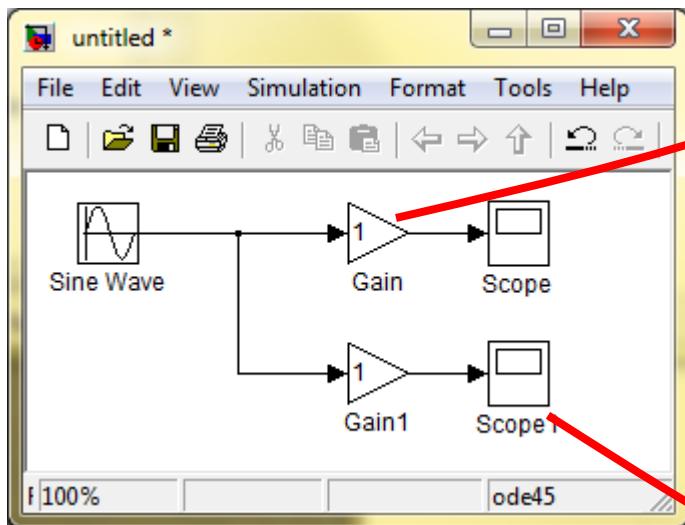


Left mouse drag between  
breakpoints



# Setting block parameters

Double or right click on block to access parameter window

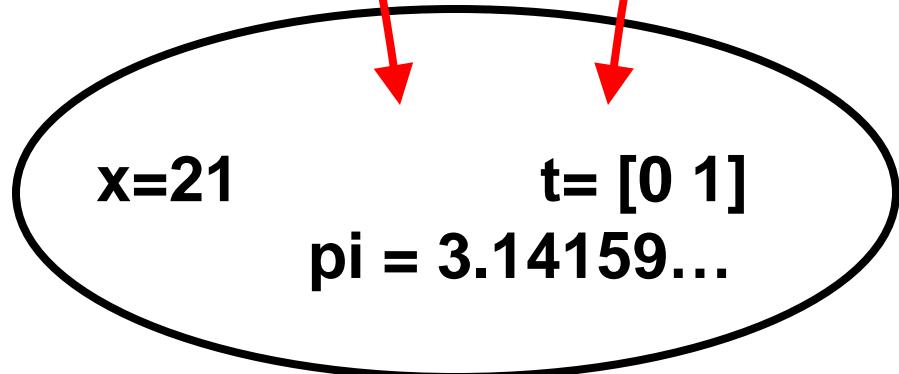
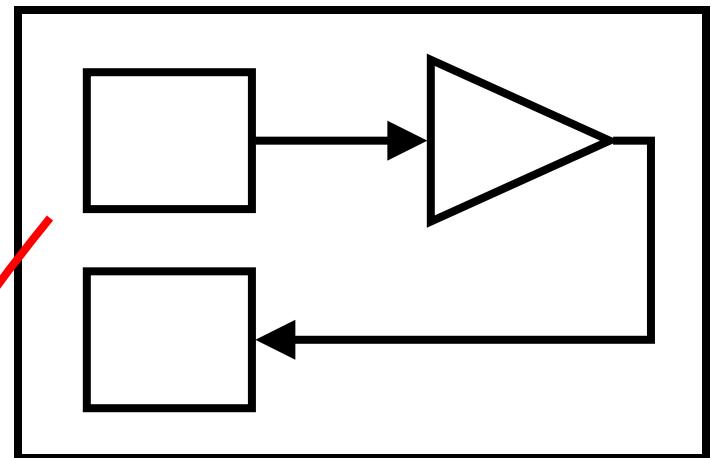


Double or right click on Scope block to open

# Defining variables

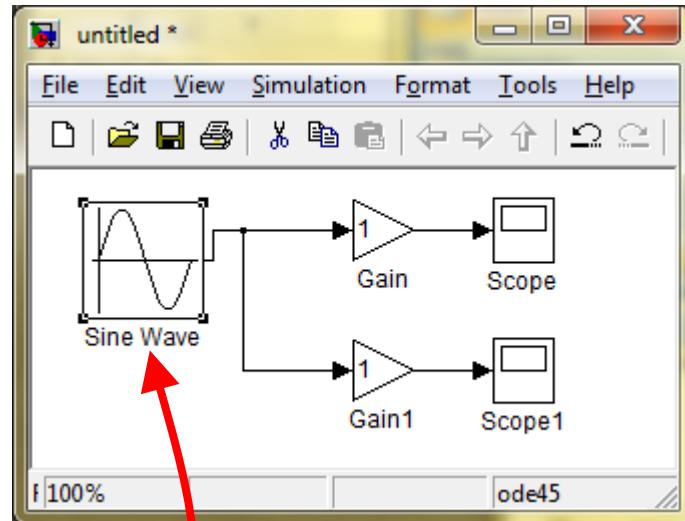
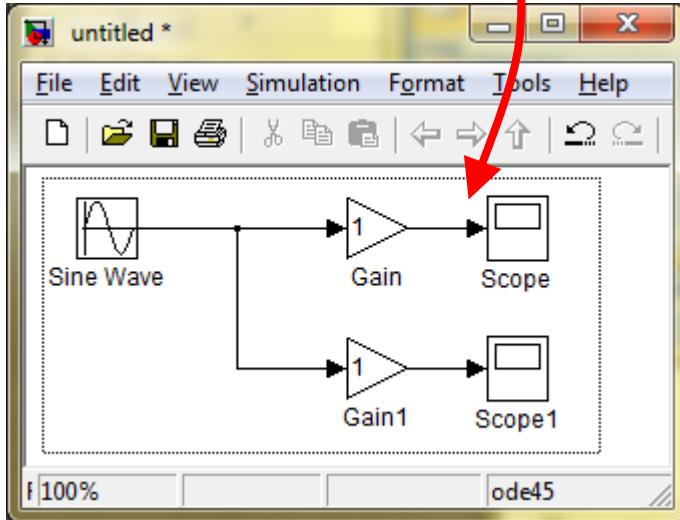
Both MATLAB and Simulink Windows “see” into the same Workspace

```
>> x = 21;  
>> t = 0:1
```

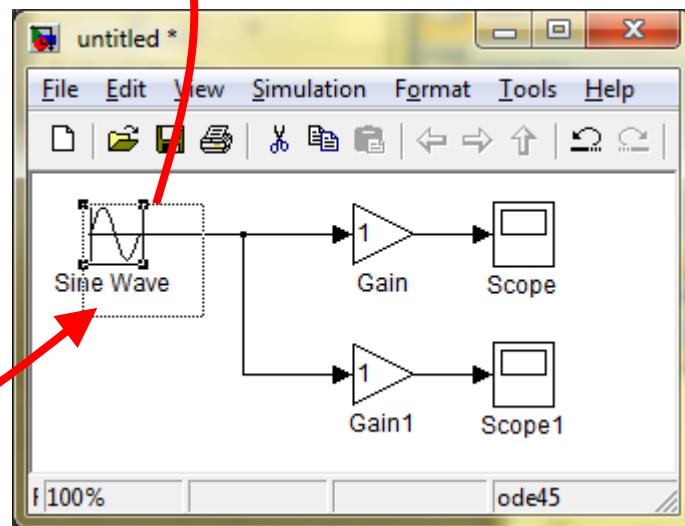


# Setting block characteristics

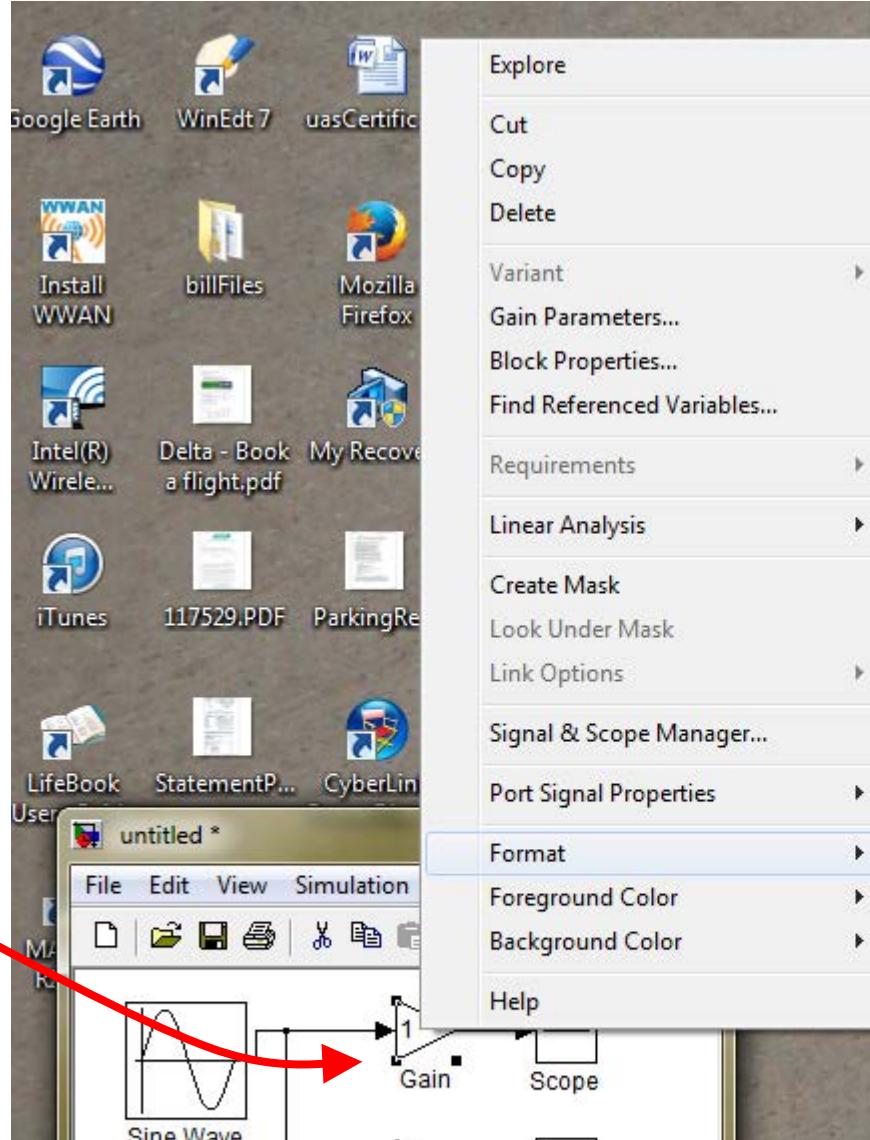
Select blocks by drawing a box around them



Drag corner of selected block to resize



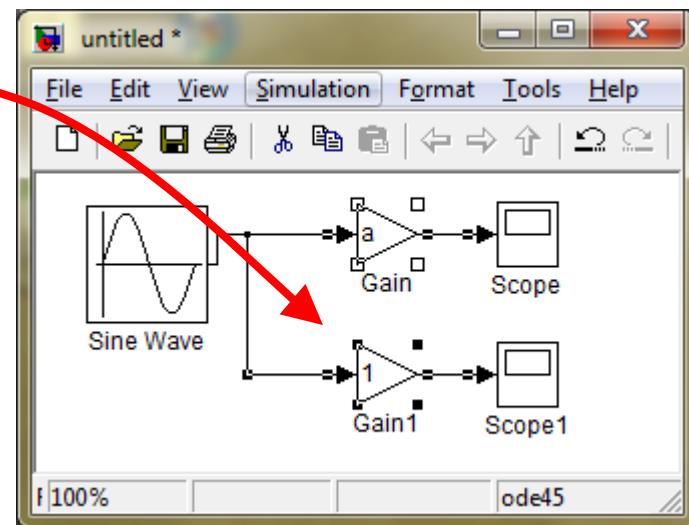
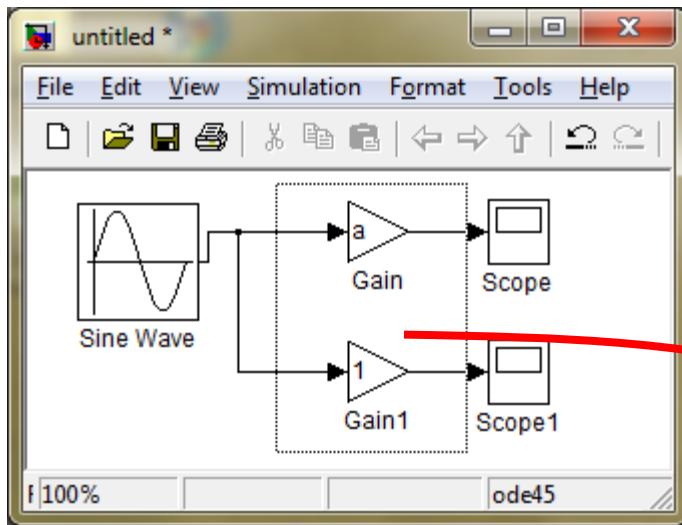
# Setting block characteristics



Right click on  
block to show  
formatting menu

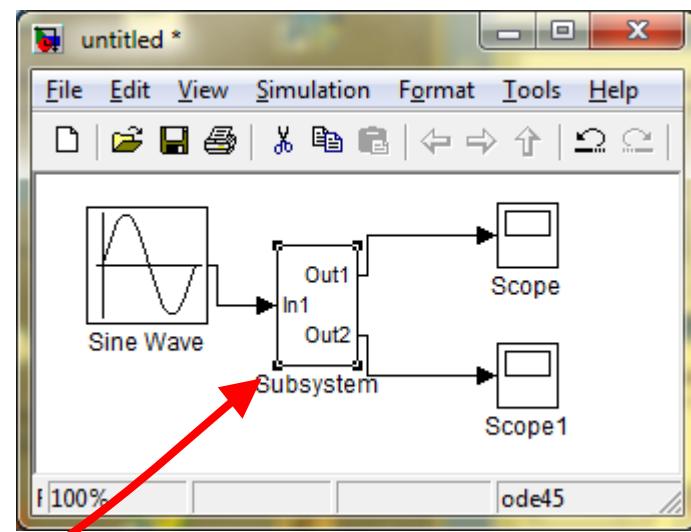
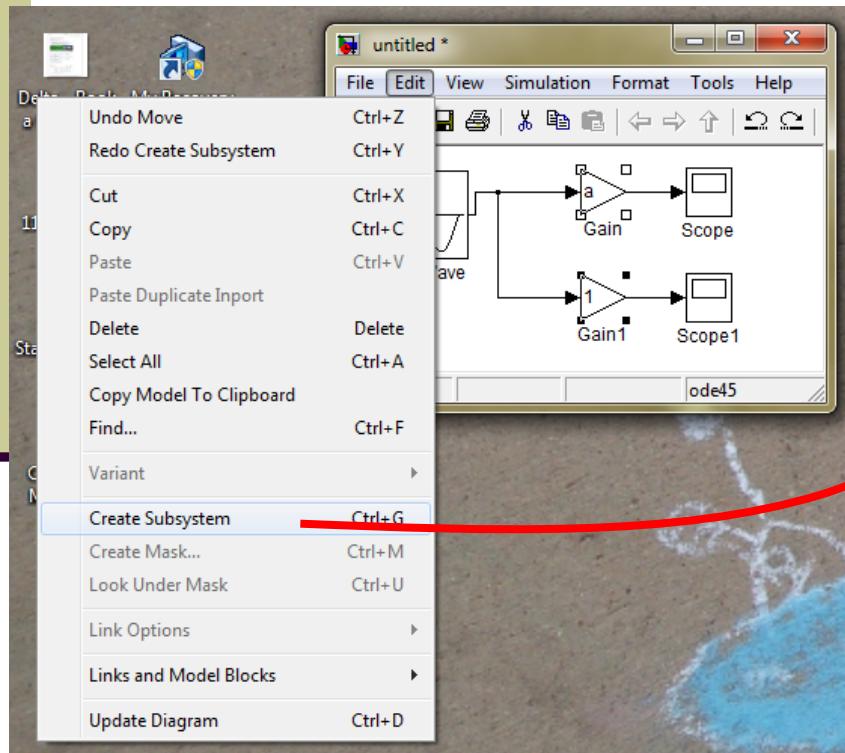
# Creating Subsystems

Draw a box around  
blocks to select area

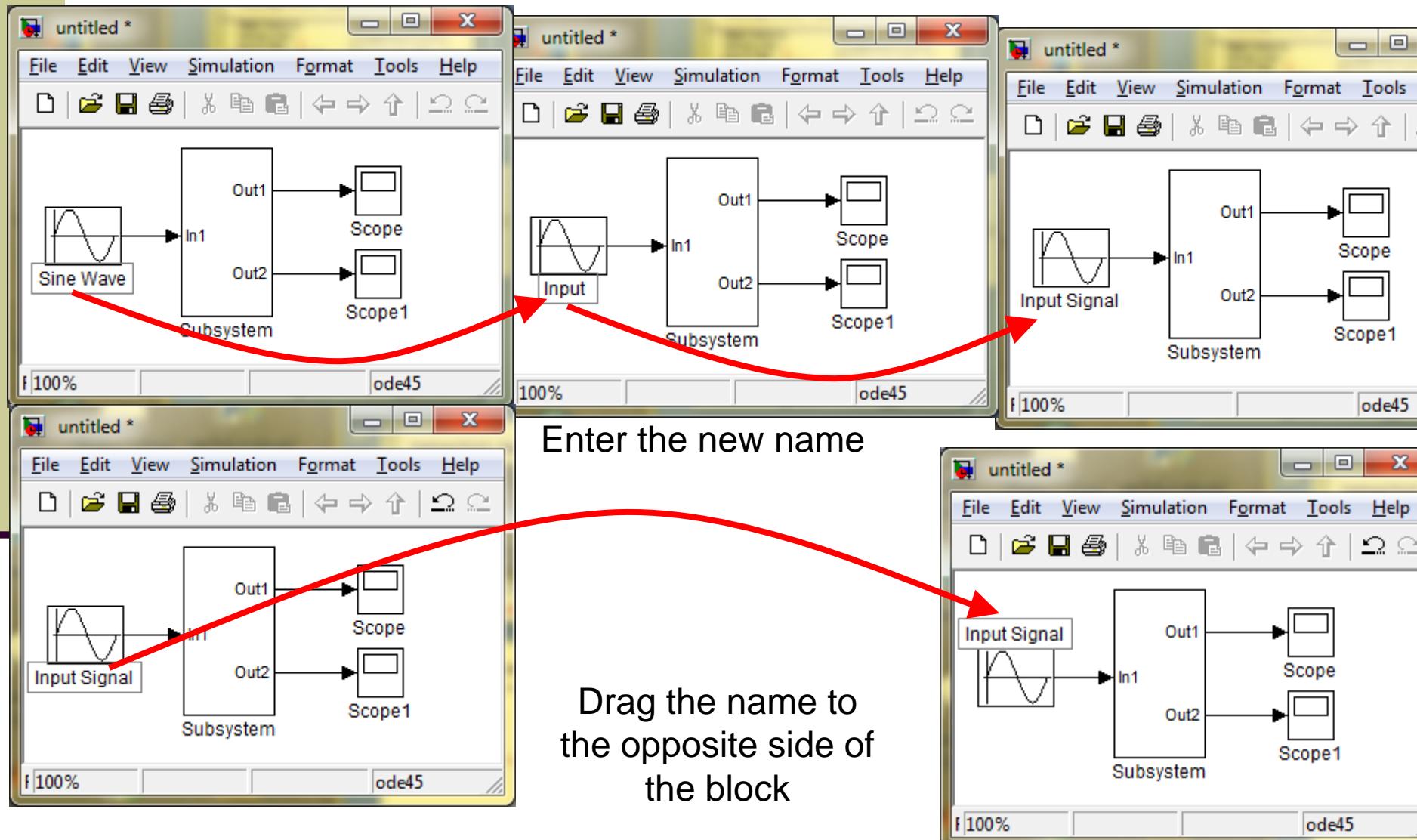


# Creating Subsystems

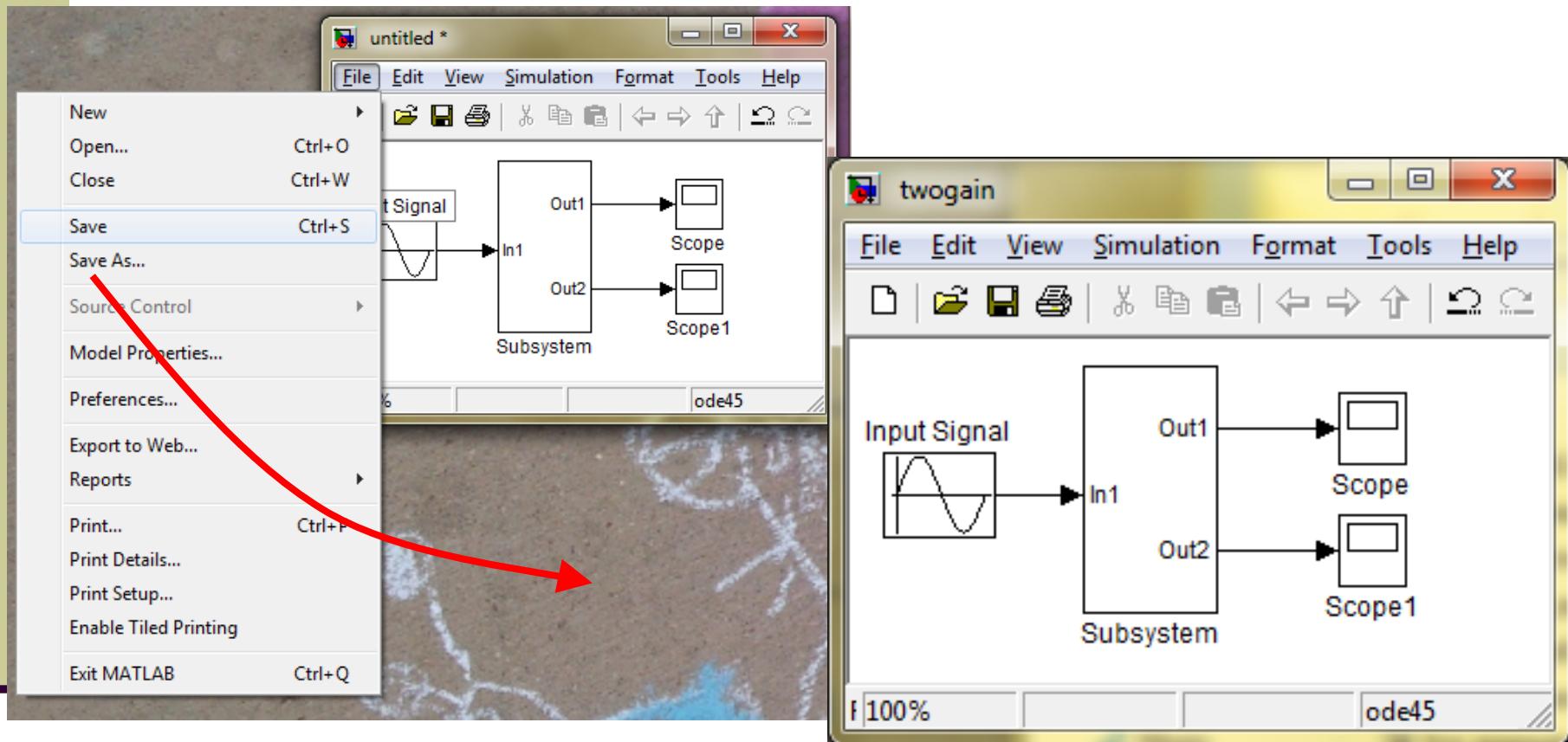
Select “Create Subsystem” under the “Edit” menu



# Changing block names



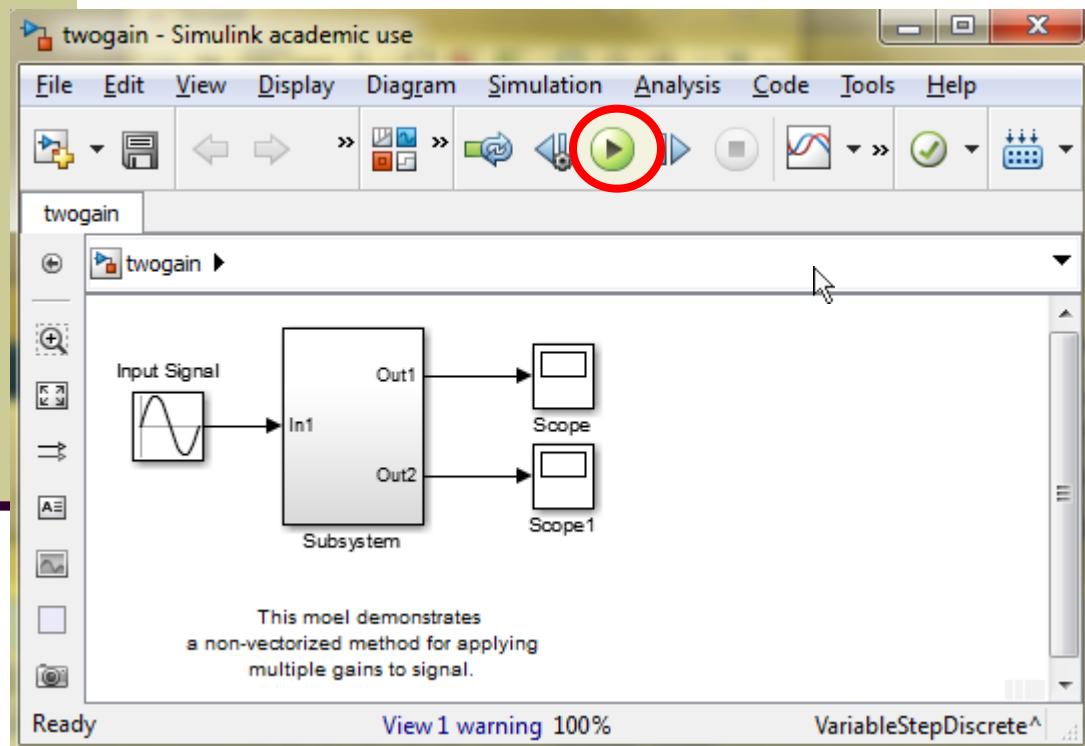
# Saving



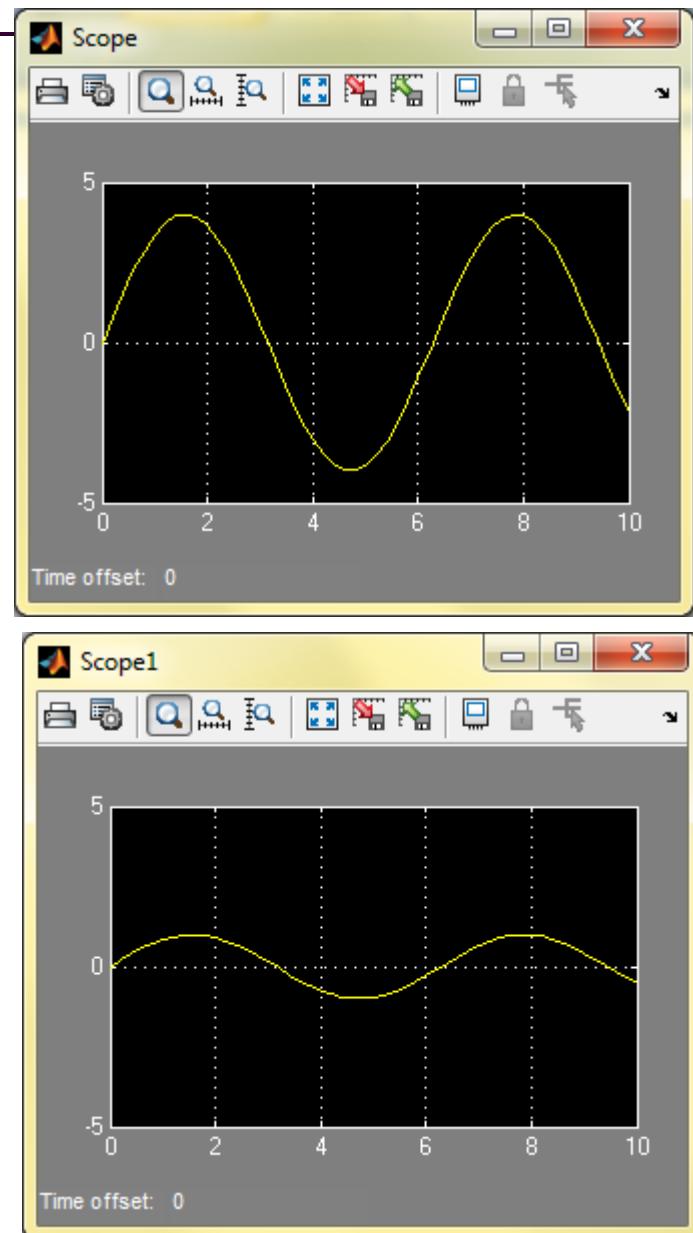
**>> twogain**

# Simulating

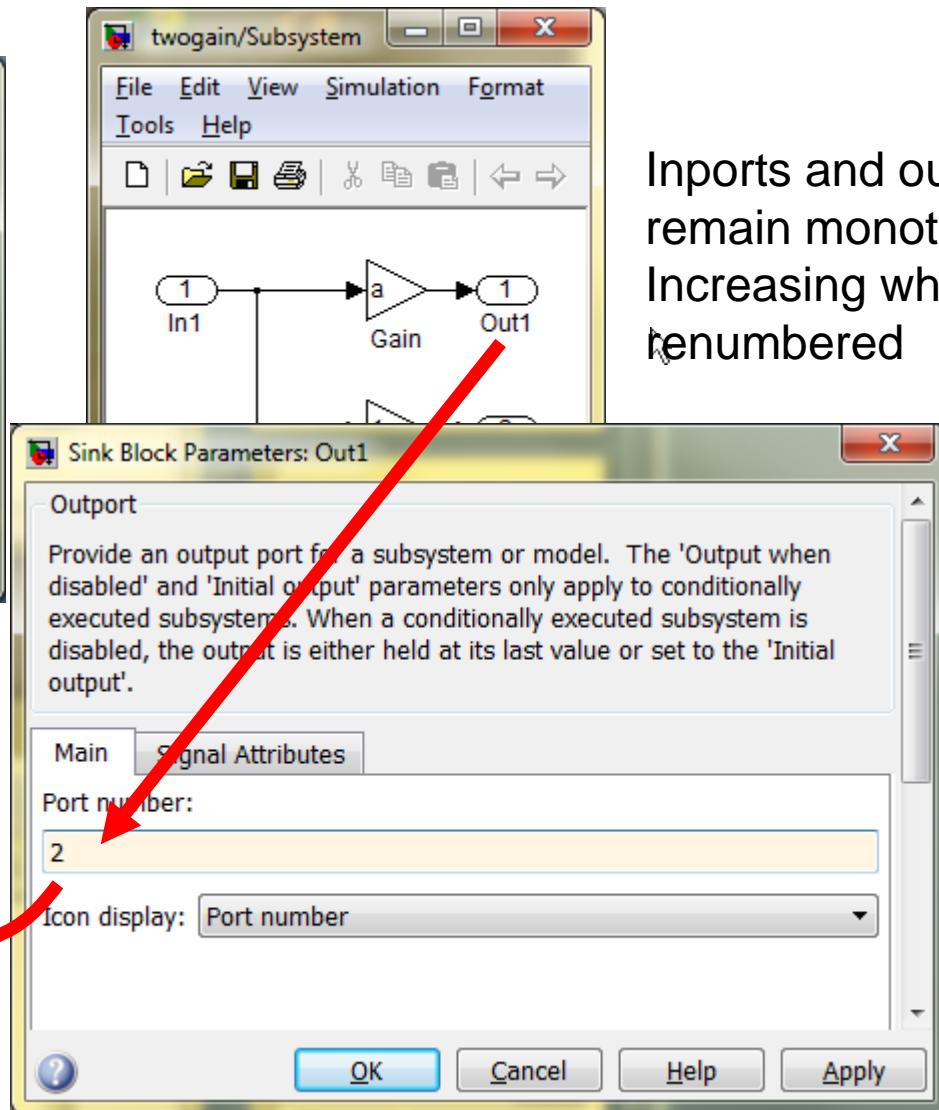
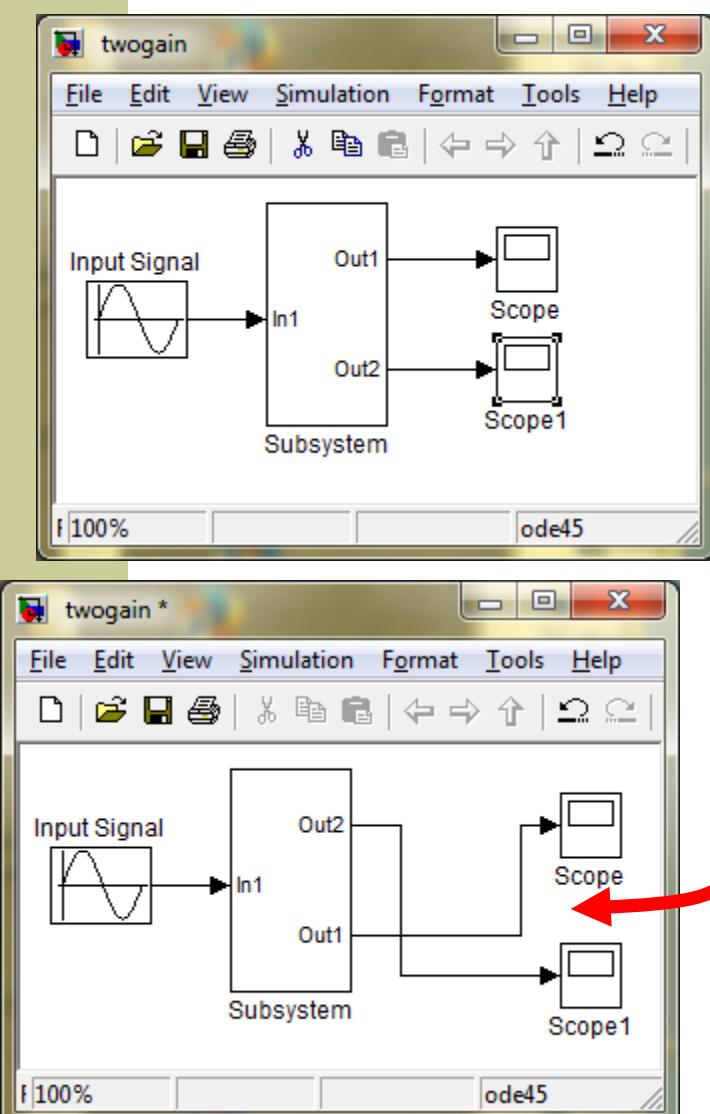
Click on the  
“Run” button to  
Run simulation



>> a=4 ;



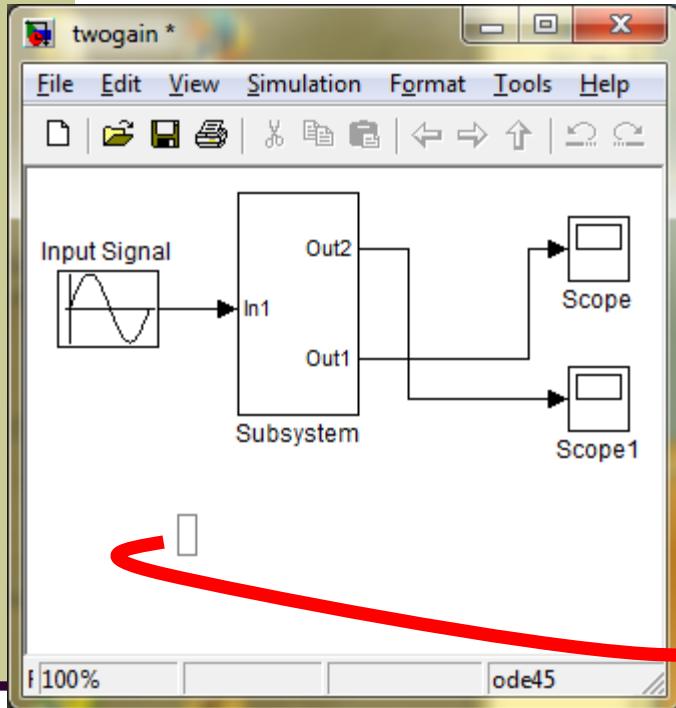
# Imports and Outports



Connectivity remains constant

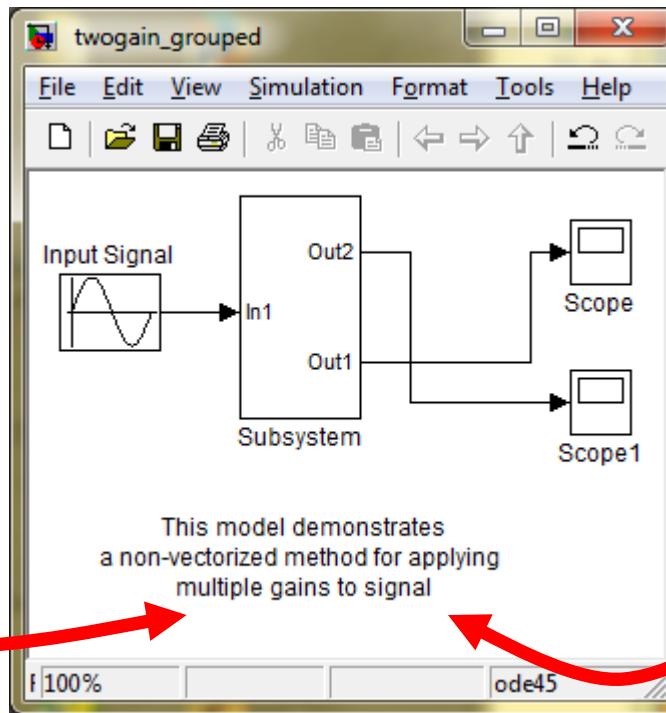
Imports and outports  
remain monotonically  
Increasing when  
renumbered

# Annotations



Double click on  
Background

>>twogain\_grouped

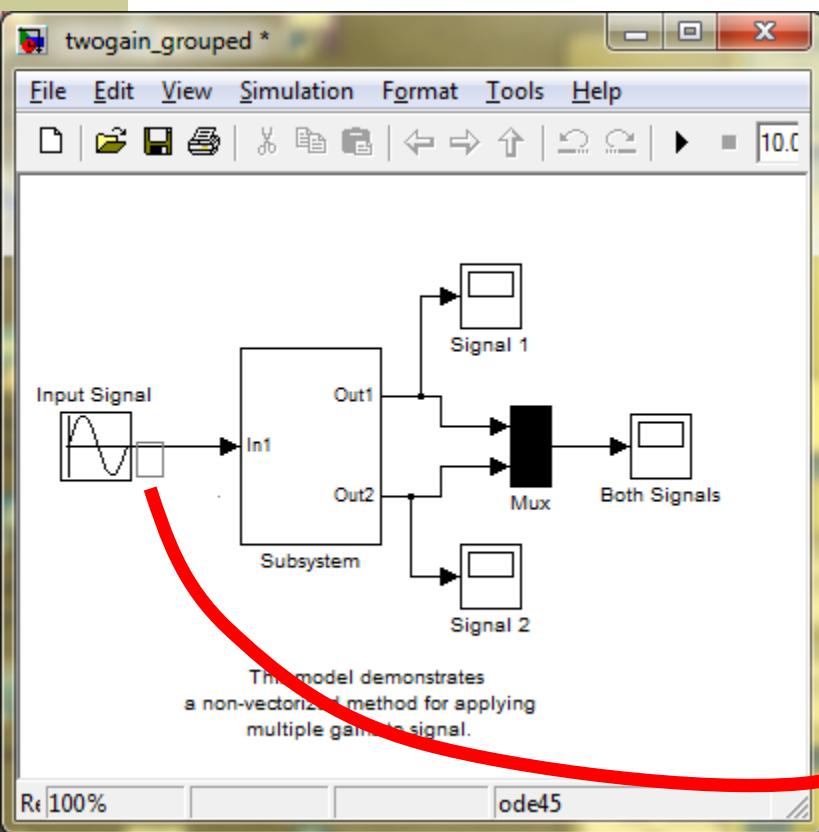


Type in the text  
and save as  
'twogain\_grouped'

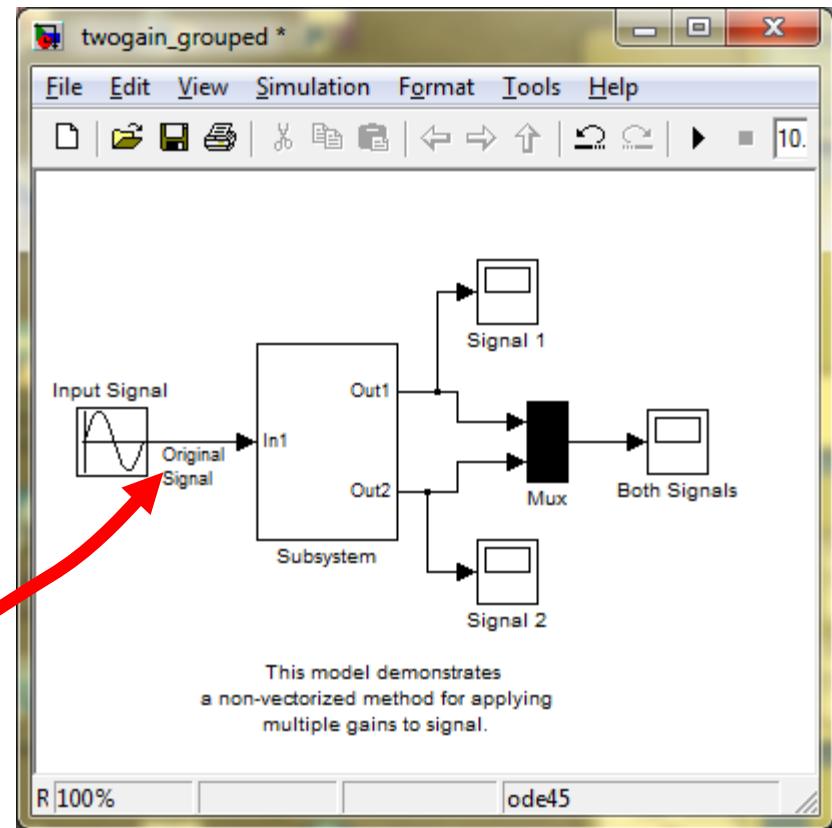
Resize font  
as desired by  
right-clicking  
on text

# Signal labels

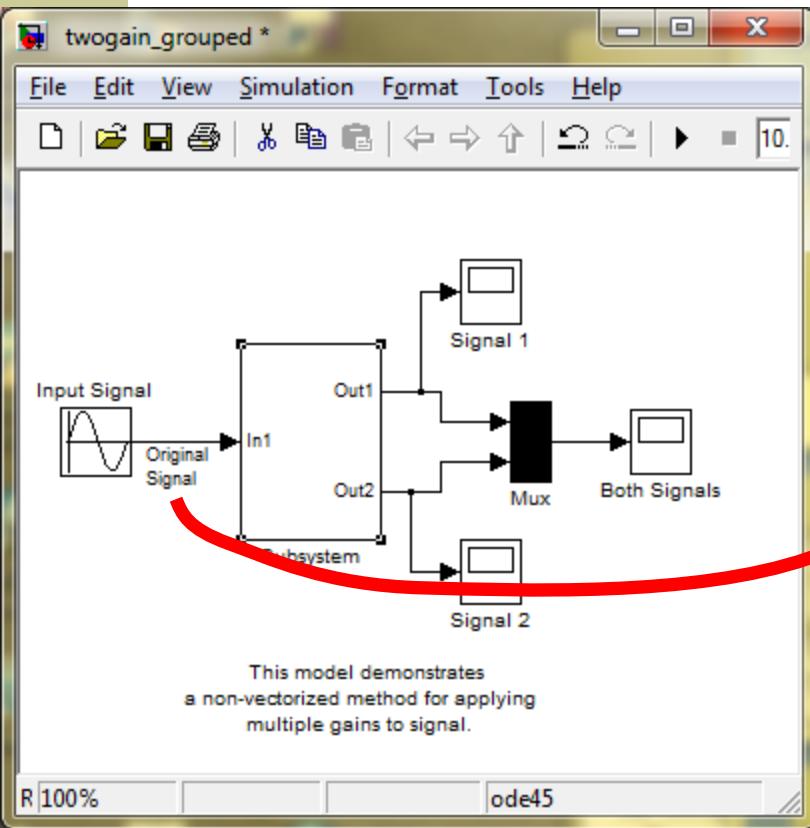
Add a **Mux** block and a third scope  
to twogain\_grouped



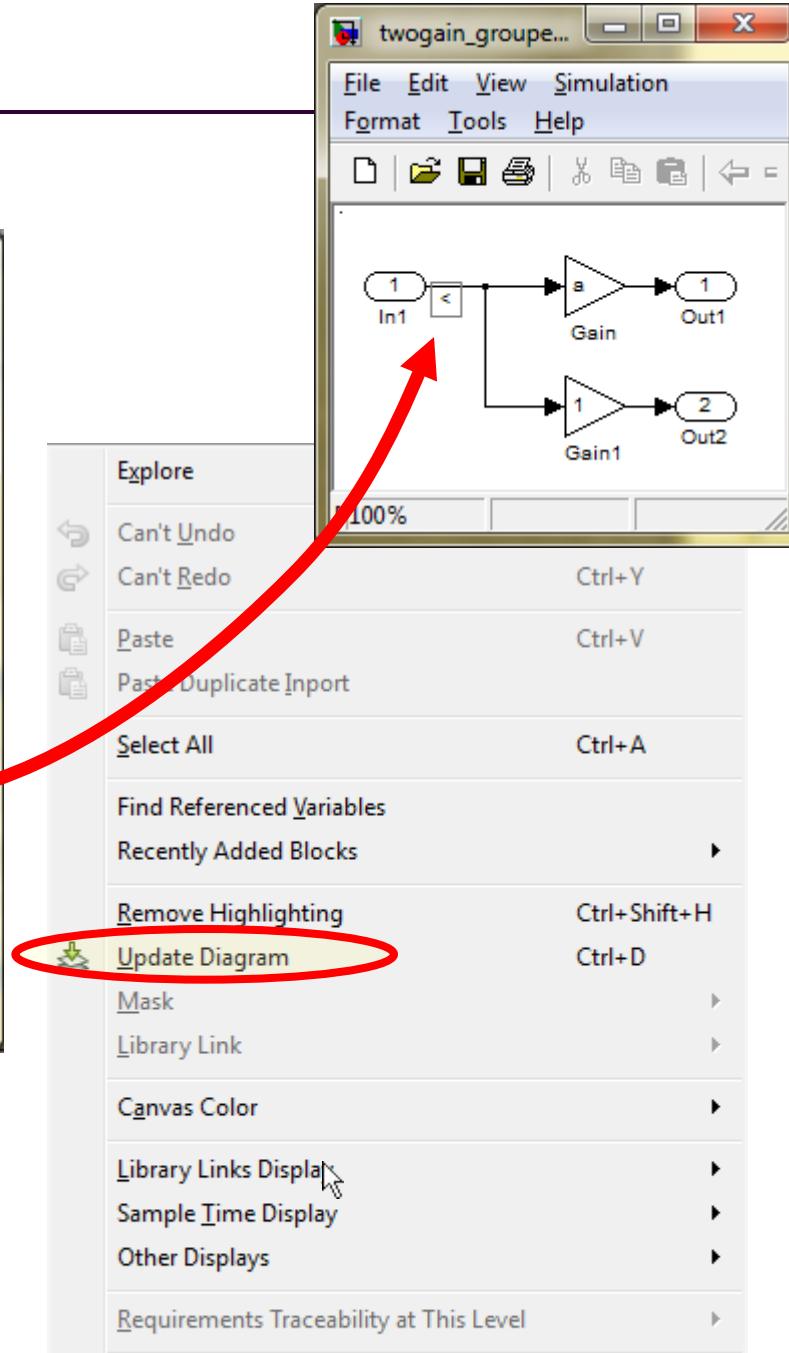
Signal labels works just like annotation, but are attached to signals



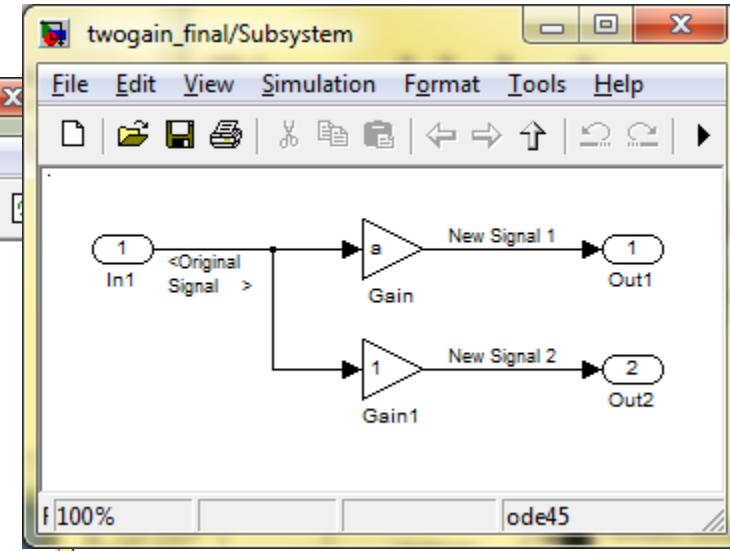
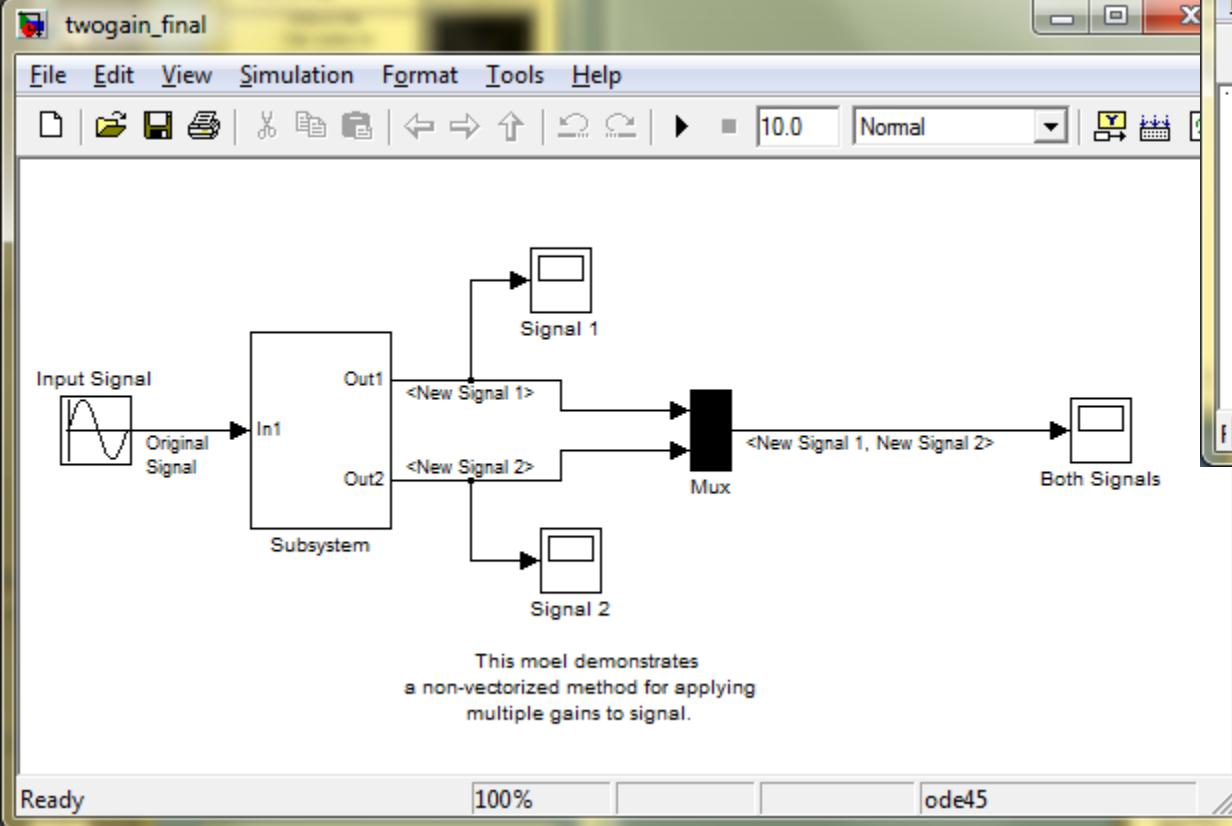
# Signal labels



Signal labels can  
pass through  
“virtual block”



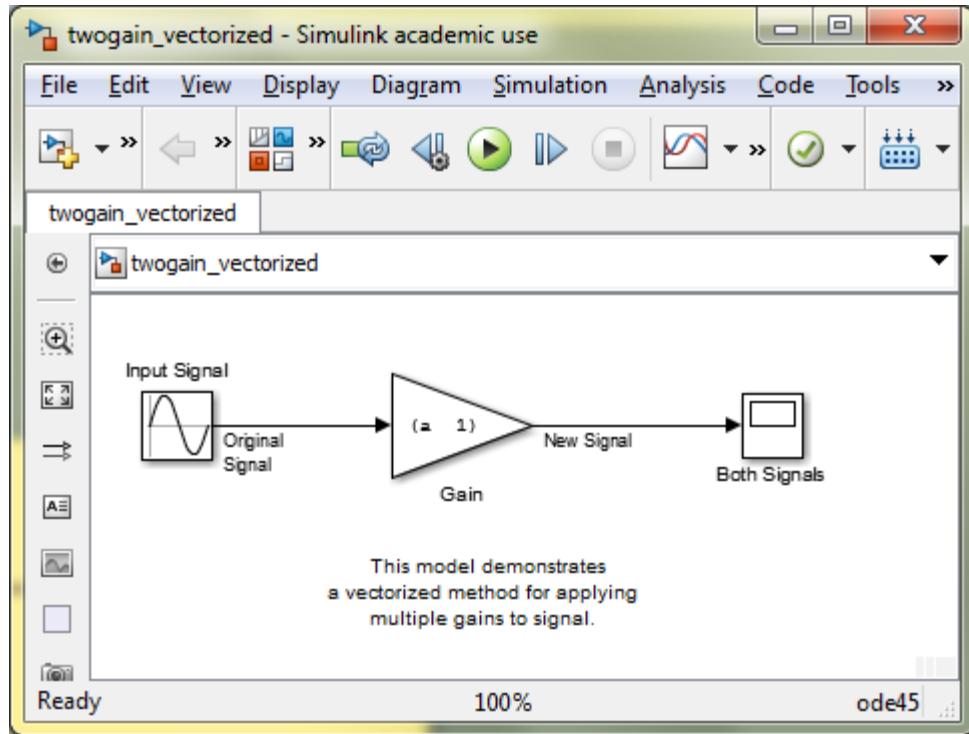
# Signal labels



Label properties can be set for each signal.

```
>>twogain_final
```

# Vectorized Signals

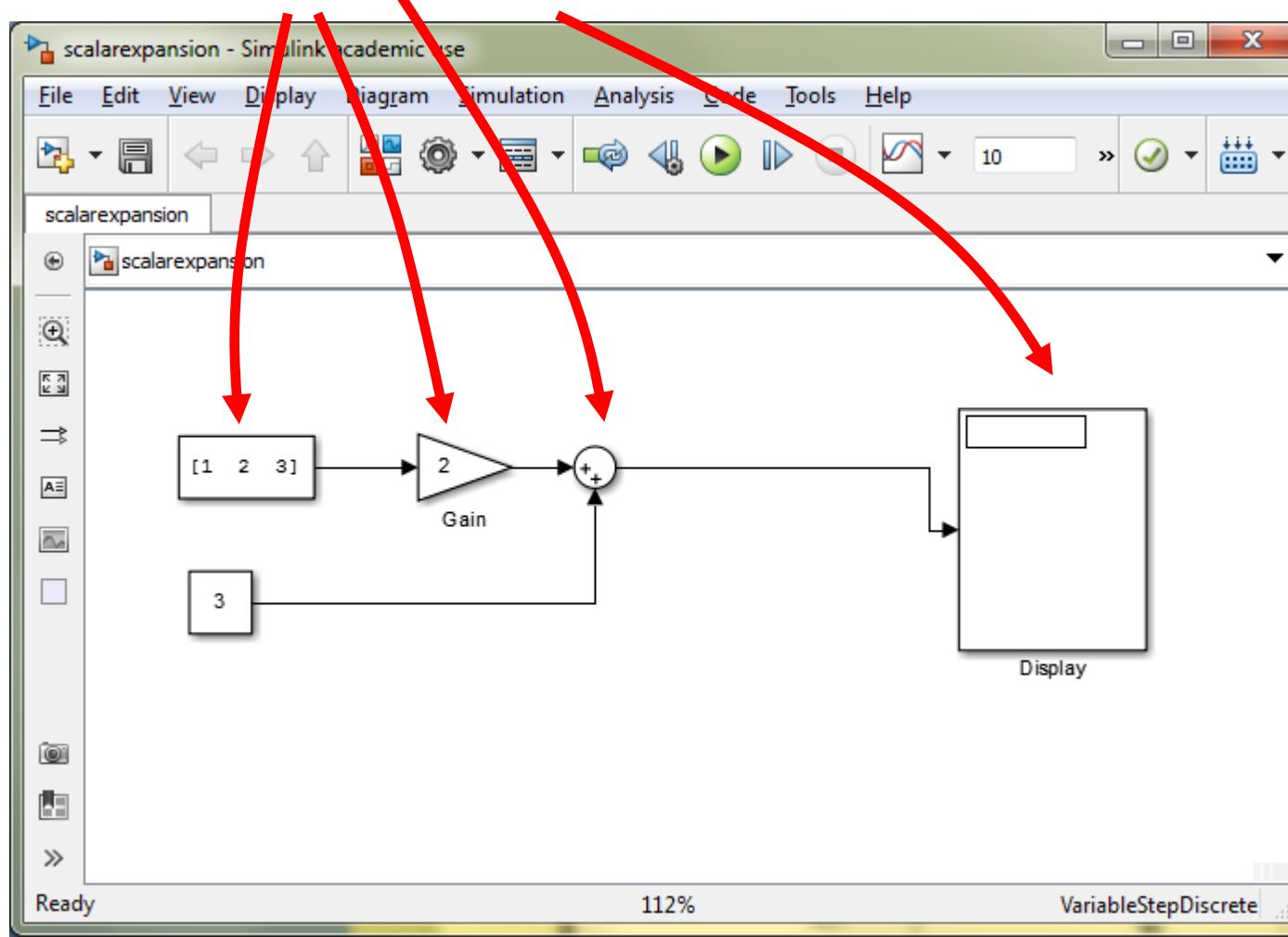


Signals can be  
vectorized.

```
>> twogain_vectorized
```

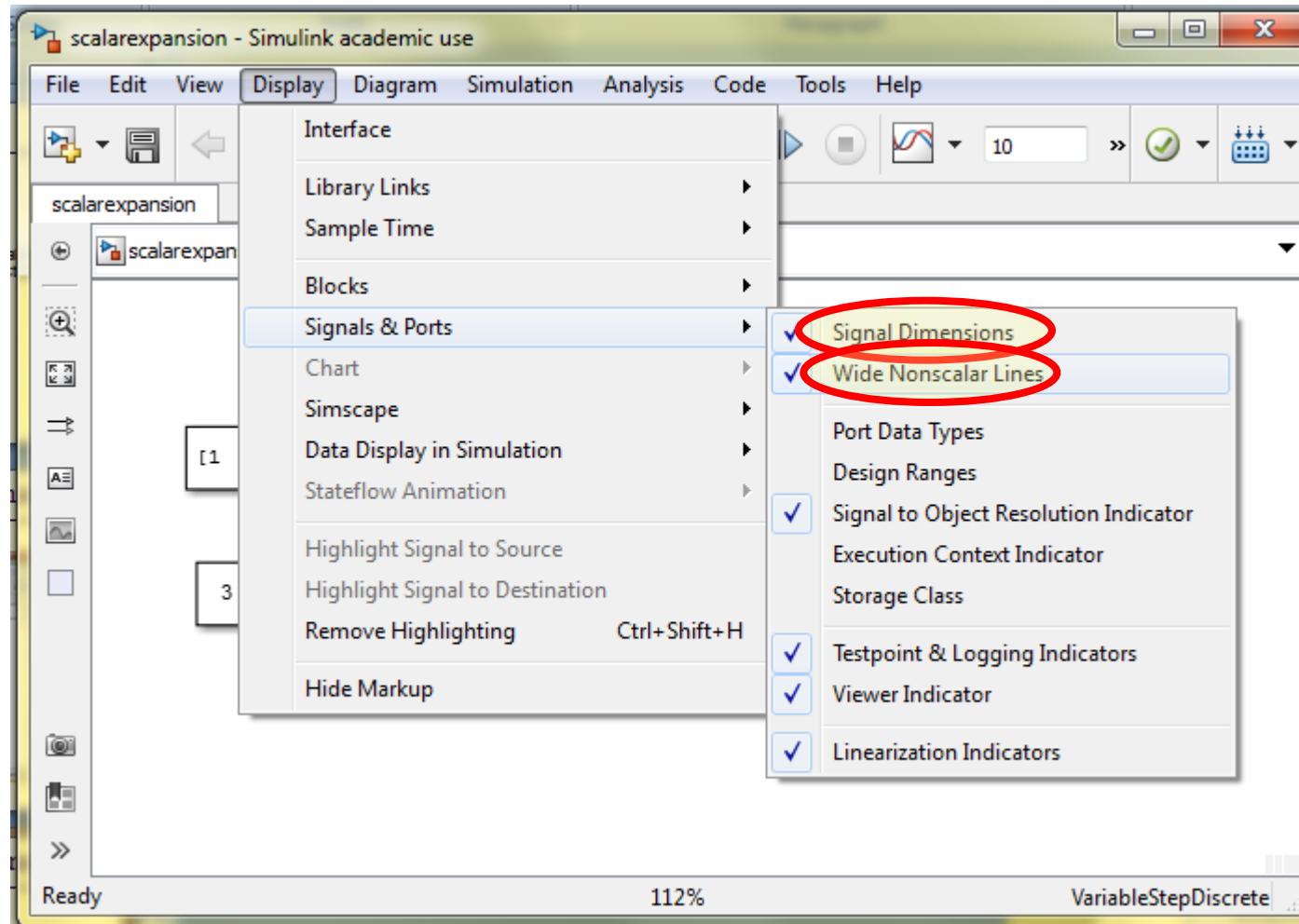
# Vector I/O and Scalar expansion

## Vectorized Inputs and Outputs



```
>> scalarexpansion
```

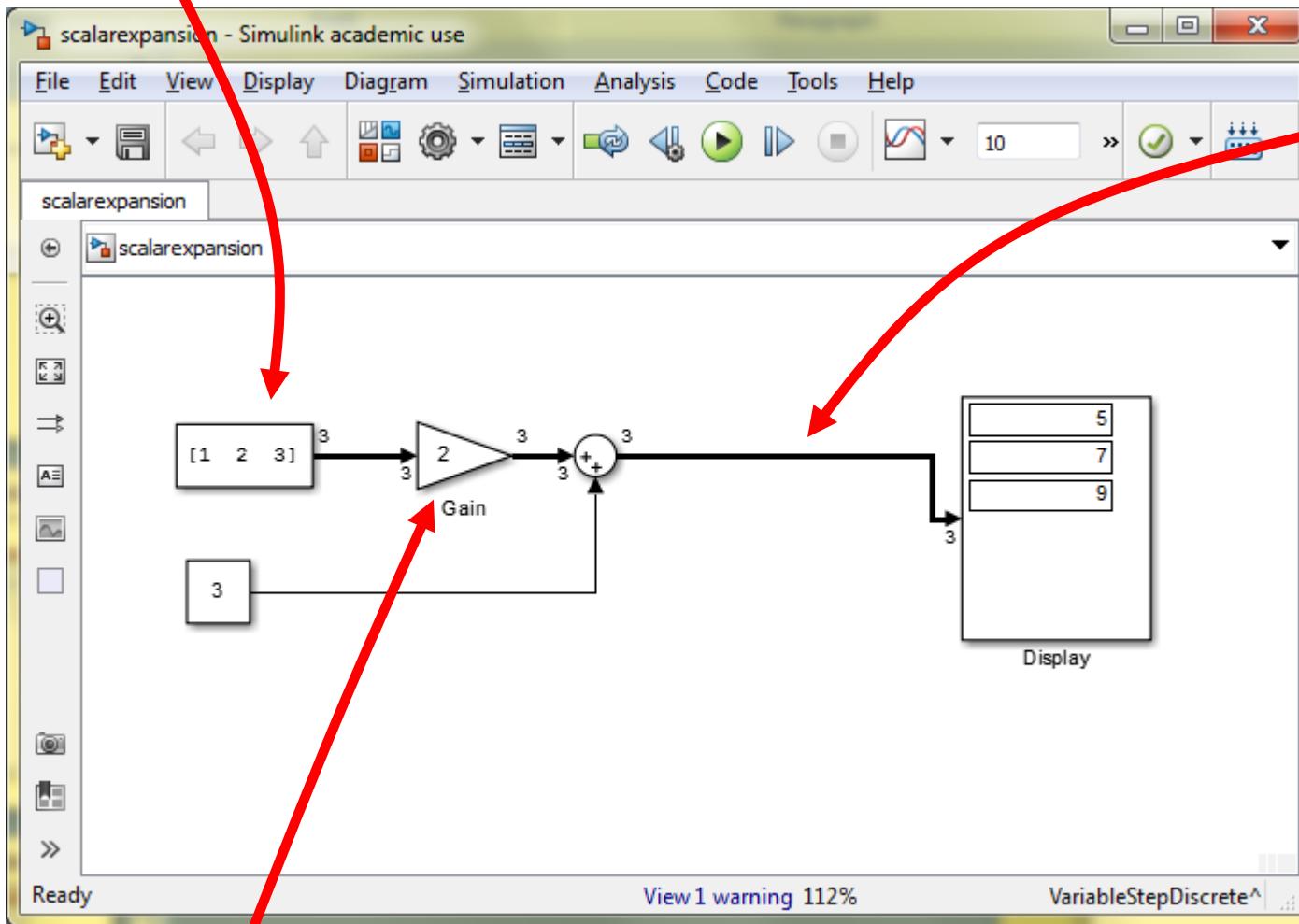
# Vector I/O and Scalar Expansion



# Vector I/O and Scalar Expansion

Scalar expansion of parameter

Wide  
Vector  
Lines  
and  
Line  
widths



Scalar expansion of inputs

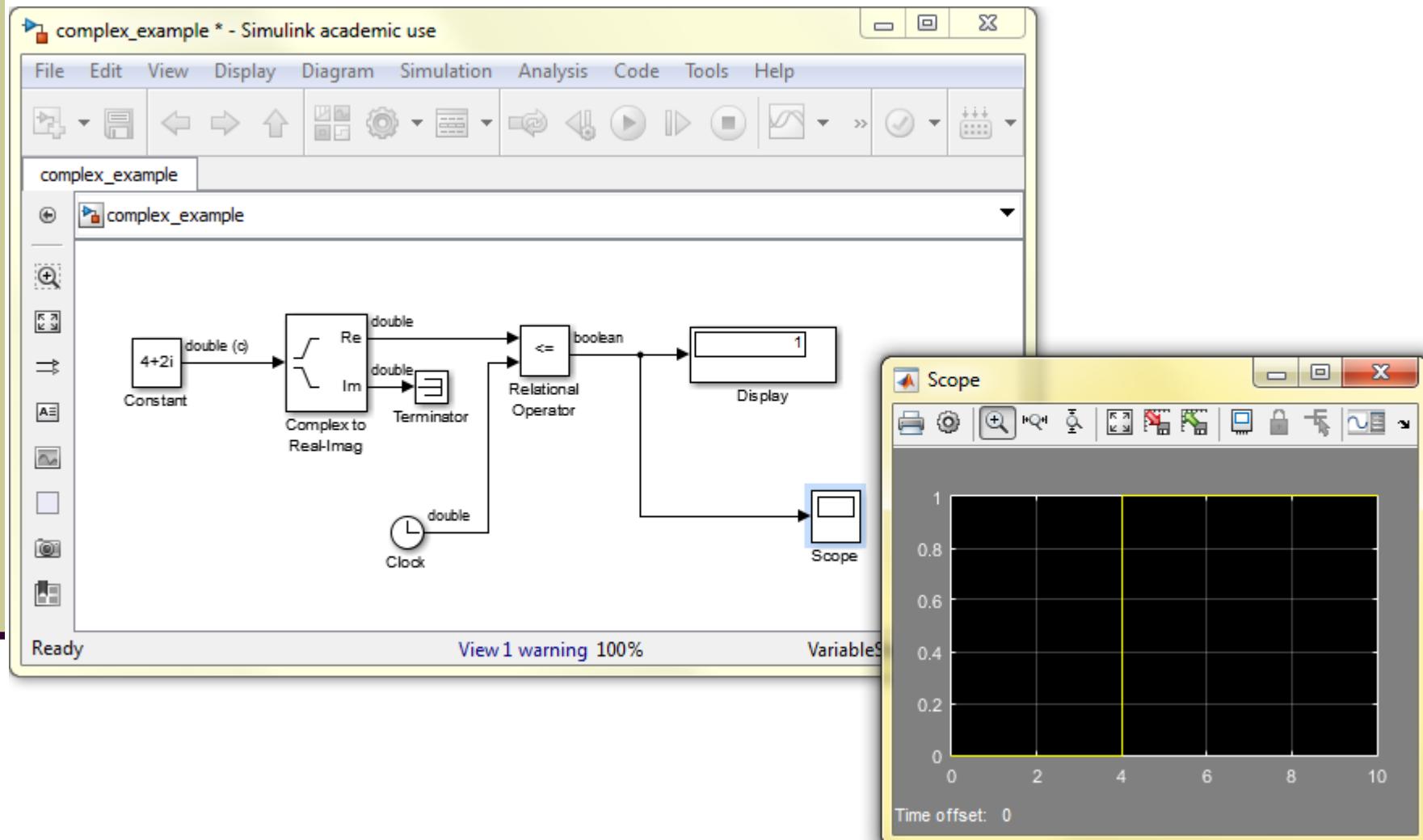
# Data Types

The screenshot shows the Simulink interface with a model named "datatypes". The model consists of three blocks: a "Constant" block set to 1, a "Data Type Conversion" block set to "uint8", and a "Display" block. A signal flows from the Constant block through the Data Type Conversion block to the Display block. The "Function Block Parameters: Data Type Conversion" dialog box is open, showing the "Data Type Conversion" tab. It contains a description: "Convert the input to the data type and scaling of the output." Below this, it says: "The conversion has two possible goals. One goal is to have the Real World Values of the input and the output be equal. The other goal is to have the Stored Integer Values of the input and the output be equal. Overflows and quantization errors can prevent the goal from being fully achieved." The "Parameters" section includes fields for "Output minimum" and "Output maximum", both currently empty. The "Output data type" dropdown menu is open, showing various options: "Inherit: Inherit via back propagation", "double", "single", "int8", "uint8" (which is highlighted with a blue selection bar), "int16", "uint16", "int32", "uint32", "boolean", "fixdt(1,16)", "fixdt(1,16,0)", "fixdt(1,16,2^0,0)", "Enum: <class name><data type expression>", and "--- Refresh data types ---". The "Input and output to workspace" and "Integer rounding mode" sections are also visible in the dialog box.

- Conversion Blocks
- Multiple Data types supported for most signal and block parameters
- Complex/Real conversions
- Automatic typecasting of parameters to signal type

>> **datatypes**

# Data Types



```
>> complex_example
```

# Model Browser

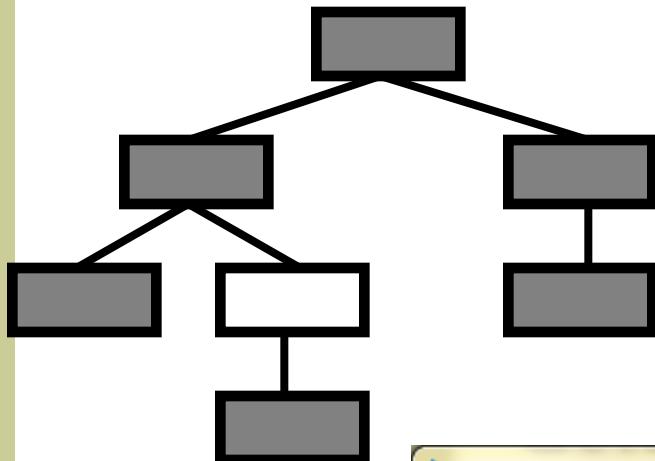
Current system and subsystems it contains

The screenshot shows the Simulink Model Browser window open, displaying a hierarchical tree of subsystems under the main model 'twogain\_final'. A red arrow points from the text 'Current system and subsystems it contains' to the Model Browser window. Below the Model Browser, a context menu is displayed for the 'Model Browser' option in the View menu. The menu includes options: 'Show Model Browser' (unchecked), 'Include Referenced Models' (checked), 'Include Library Links' (unchecked), and 'Include Systems with Mask Parameters' (unchecked). To the right of the Model Browser window, a portion of the Simulink diagram is visible, showing a subsystem block with two outputs ('Out1' and 'Out2') and two signal lines labeled '<New Signal 1>' and '<New Signal 2>'. The status bar at the bottom right shows '100%'.

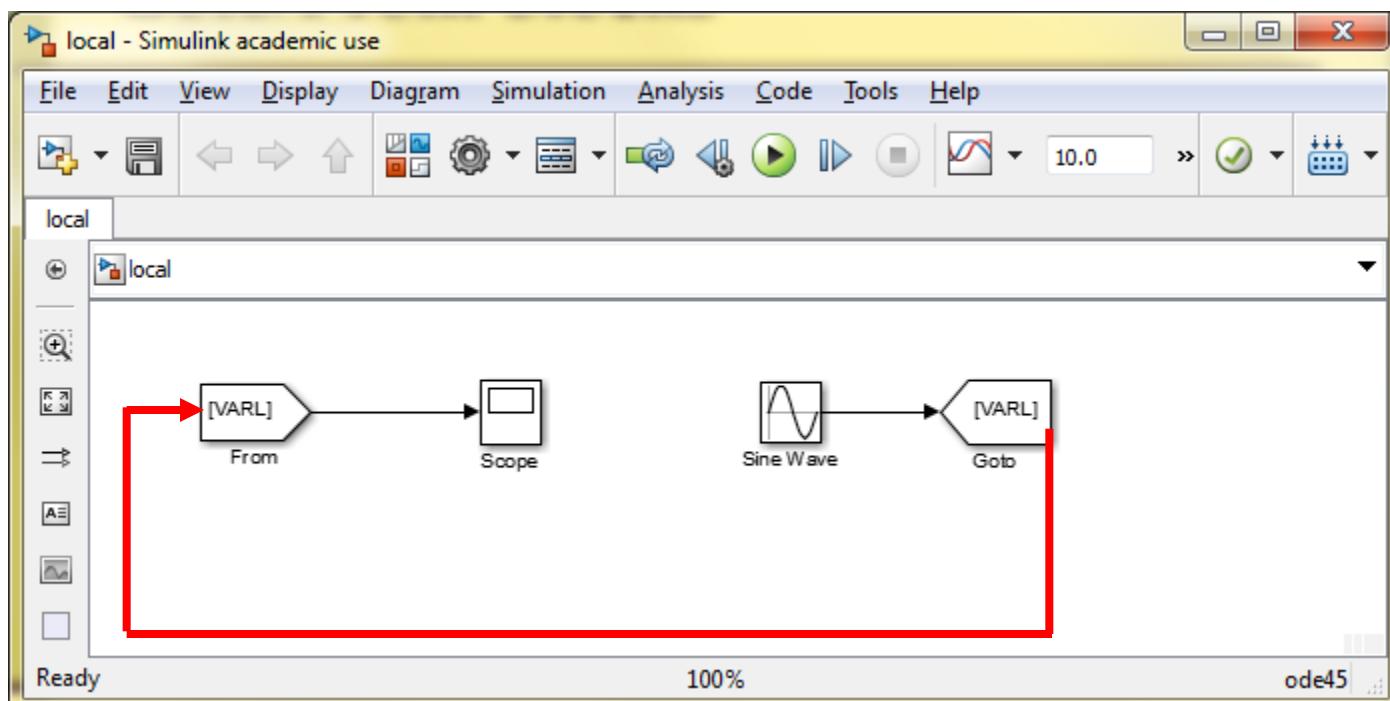
Enables the user to:

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

# Goto/From blocks

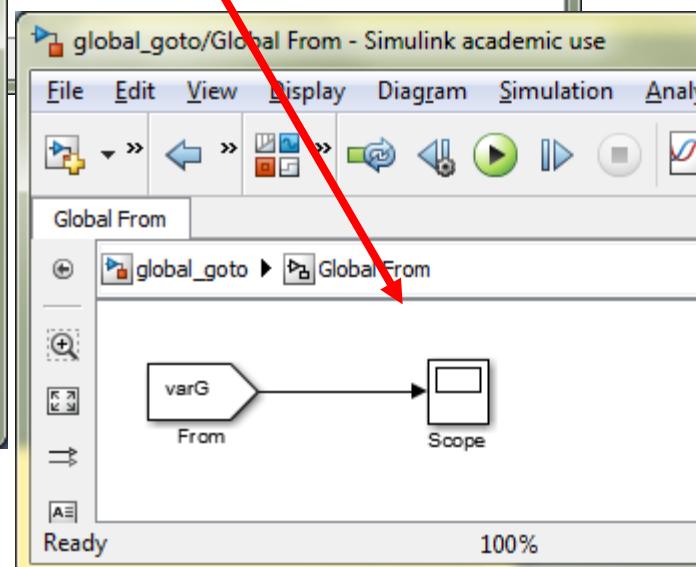
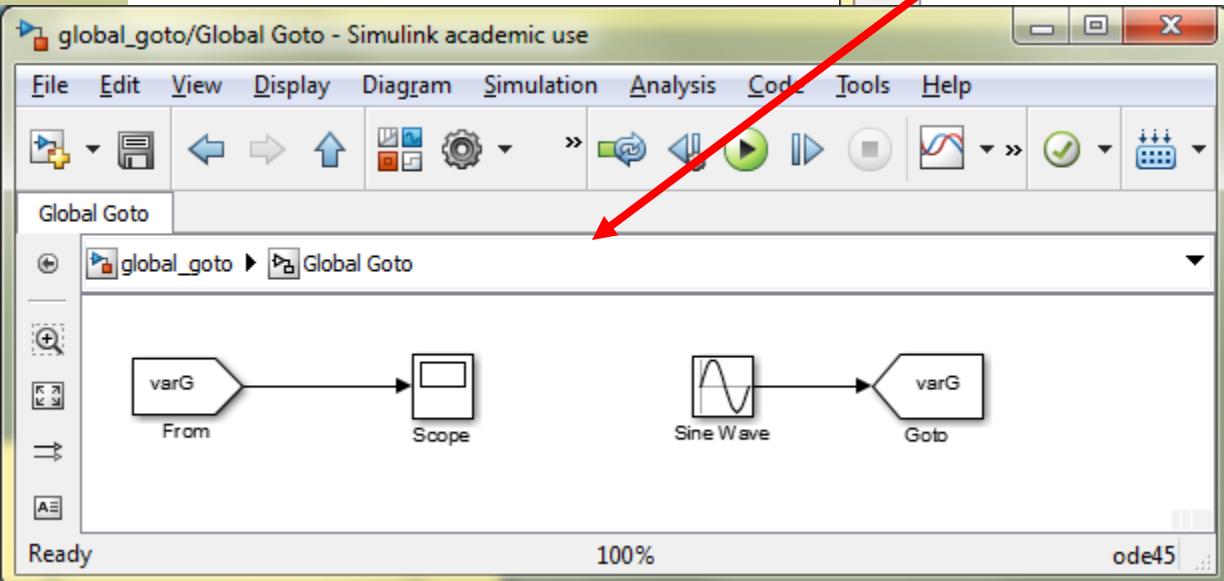
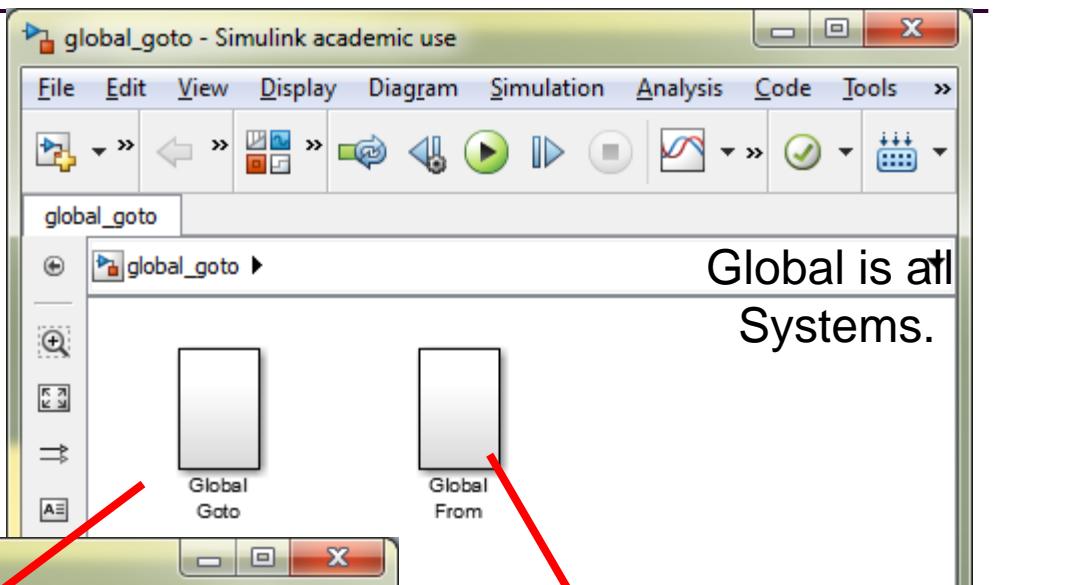
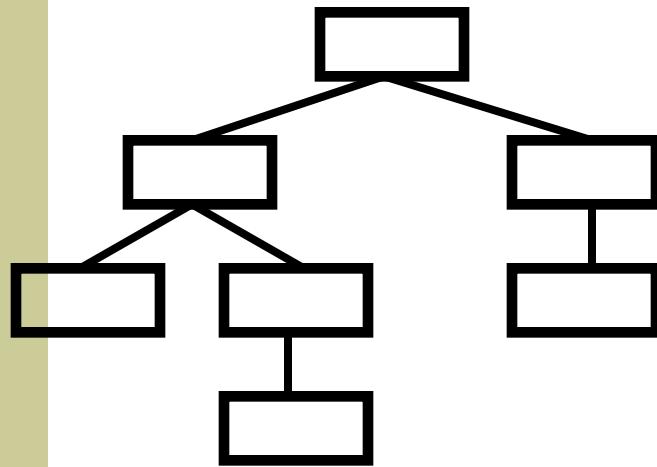


Local [ ] is only  
on the current  
window



>> local

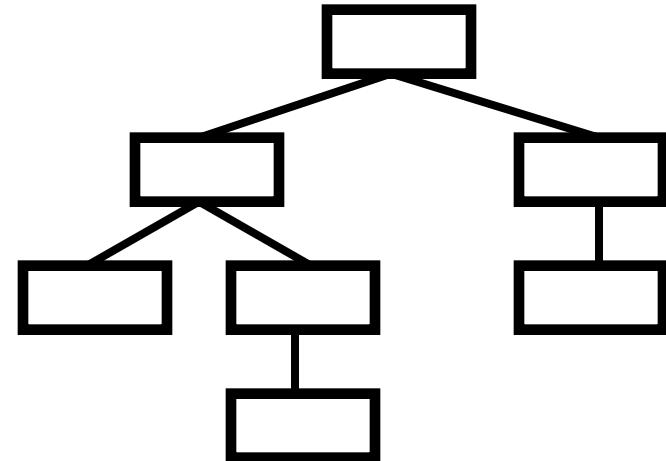
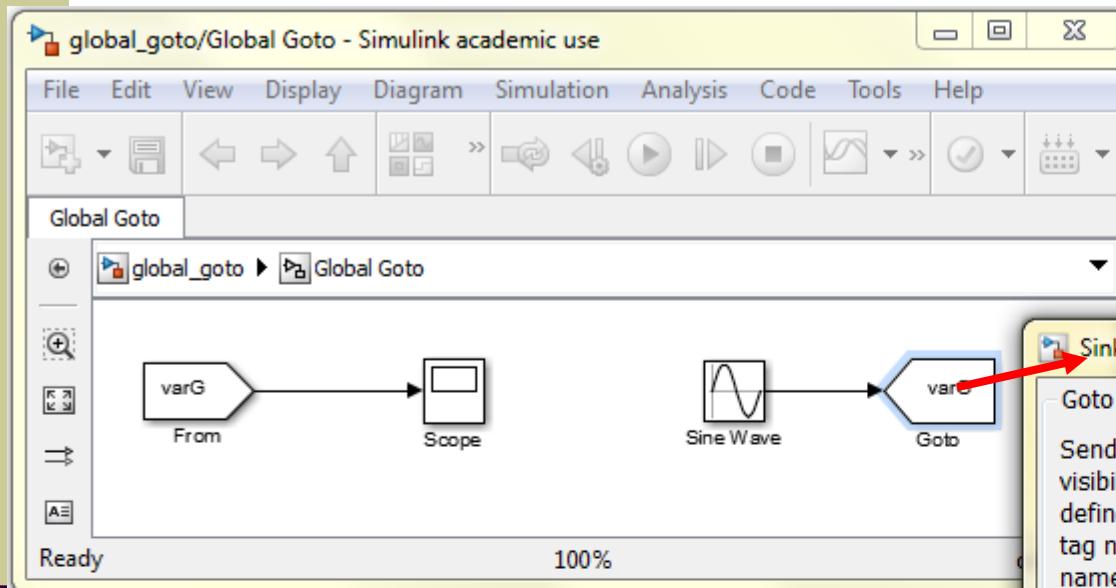
# Goto / From blocks



>> global\_goto

# Goto / From blocks

Global is all  
Systems.



**Sink Block Parameters: Goto**

**Goto**

Send signals to From blocks that have the specified tag. If tag visibility is 'scoped', then a Goto Tag Visibility block must be used to define the visibility of the tag. The block icon displays the selected tag name (local tags are enclosed in brackets, [], and scoped tag names are enclosed in braces, {}).

**Parameters**

Goto tag: varG      Tag visibility: global

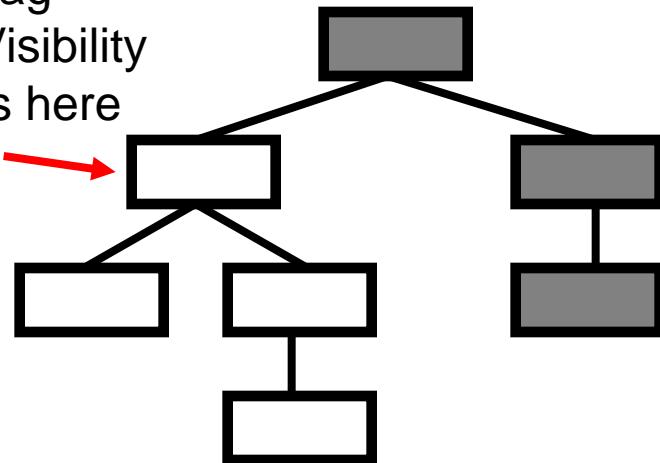
local  
scoped  
global

**Corresponding From blocks:**  
[global\\_goto/Global Goto/From](#)  
[global\\_goto/Global From/From](#)

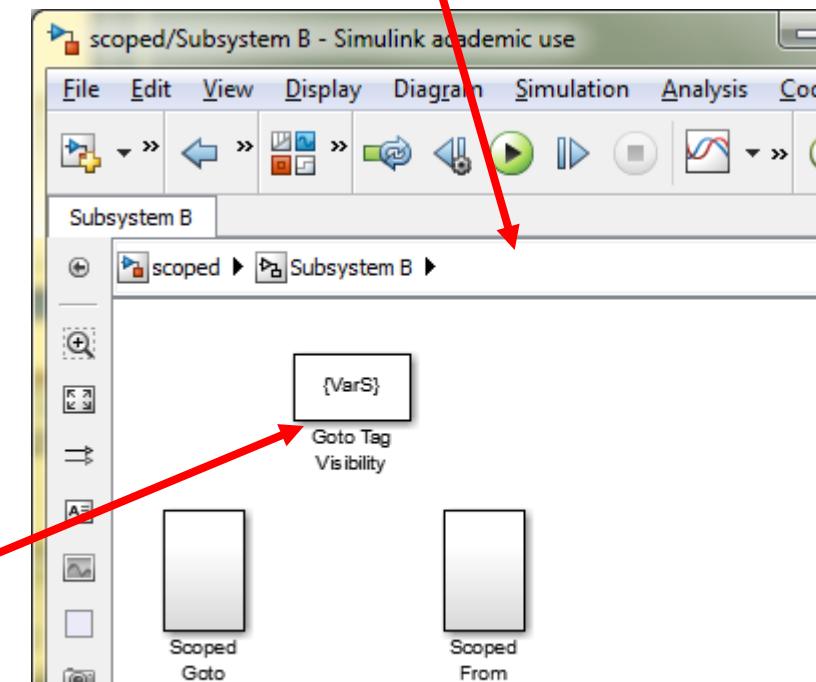
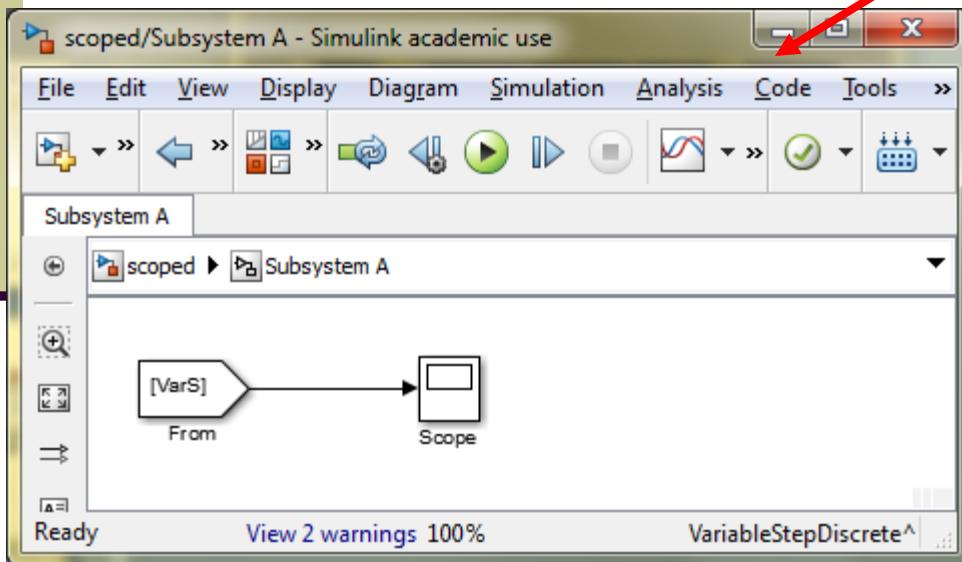
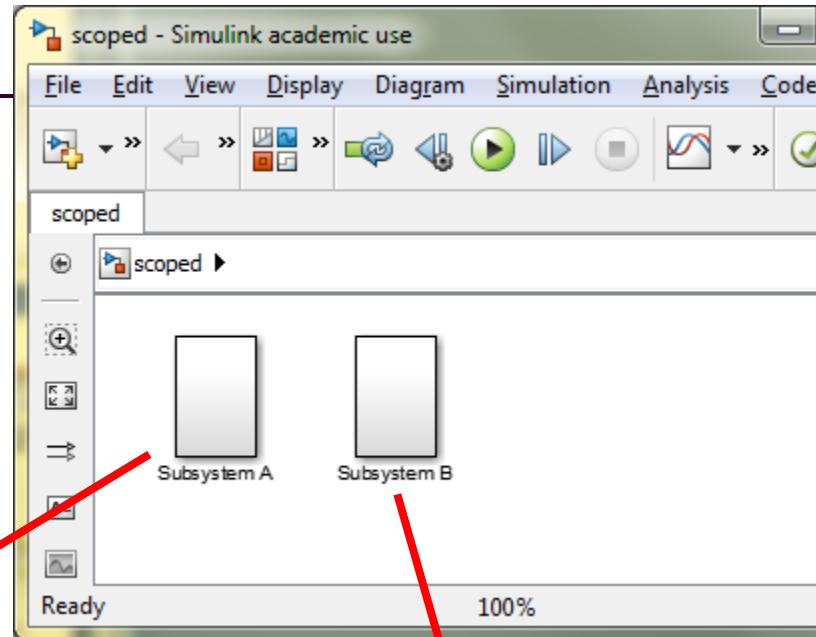
>> global\_goto

# Goto / From blocks

Tag  
Visibility  
Is here

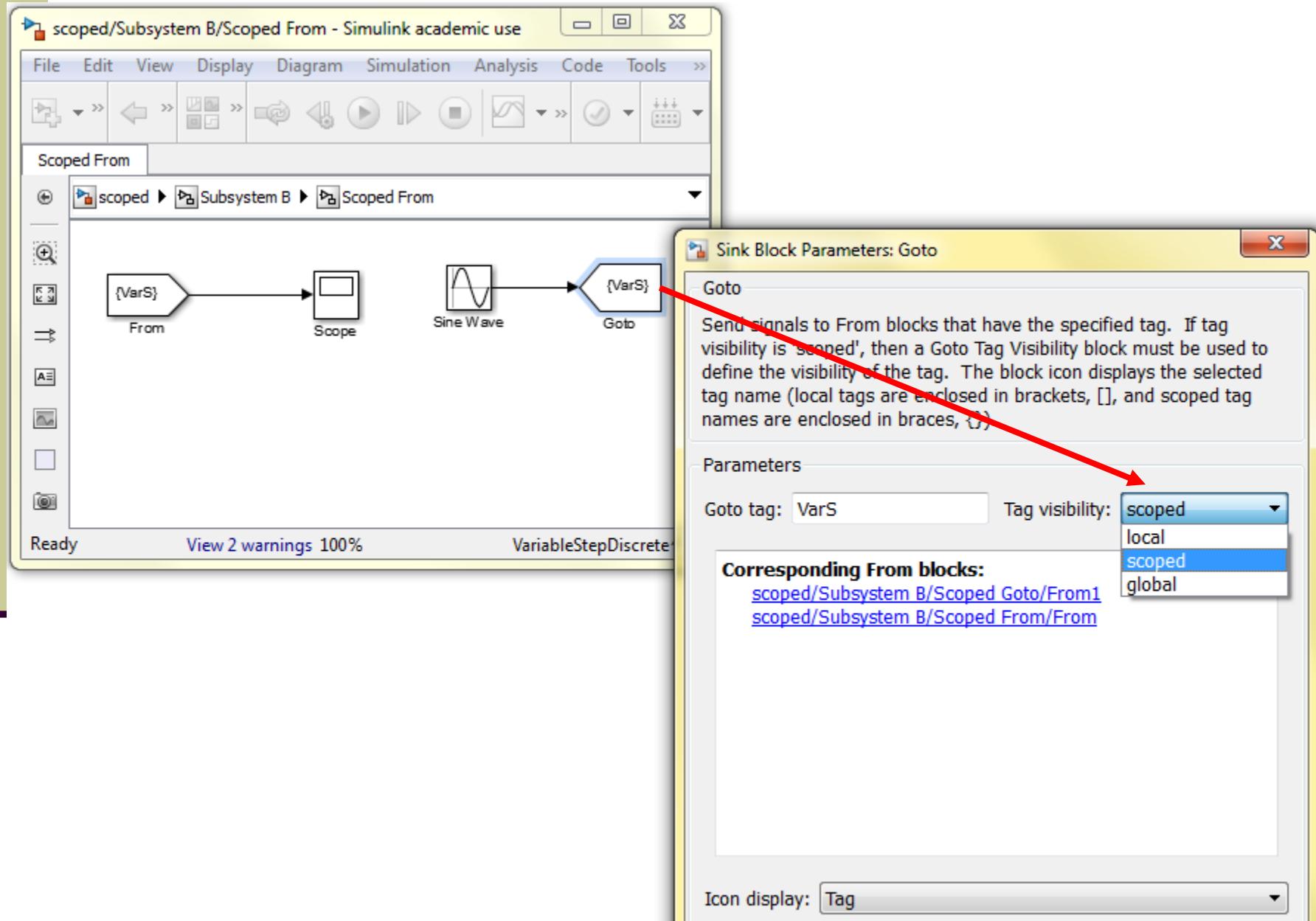


>> scoped

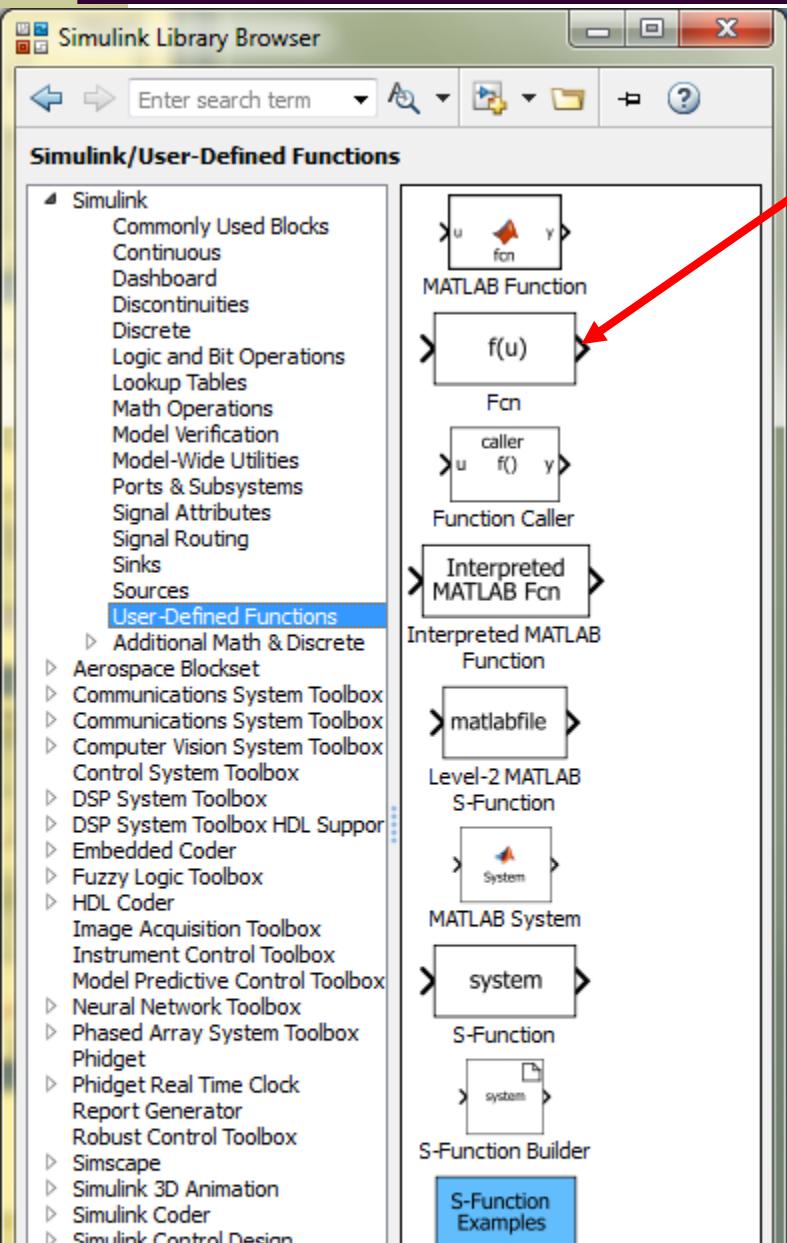


Scoped { } is all systems  
under the Tag Visibility block.

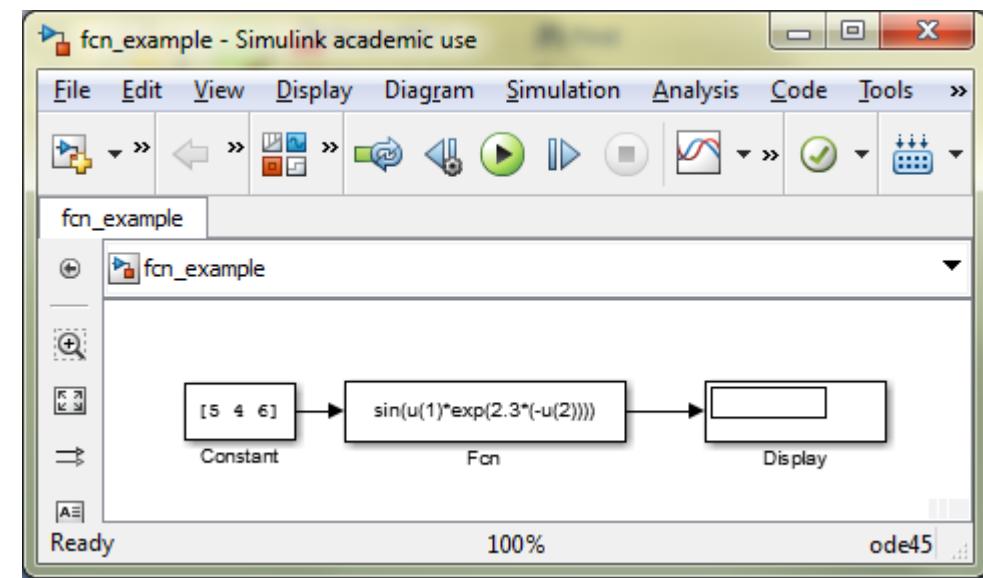
# Goto/From blocks



# User Defined Functions



Allows general expression in terms of u.



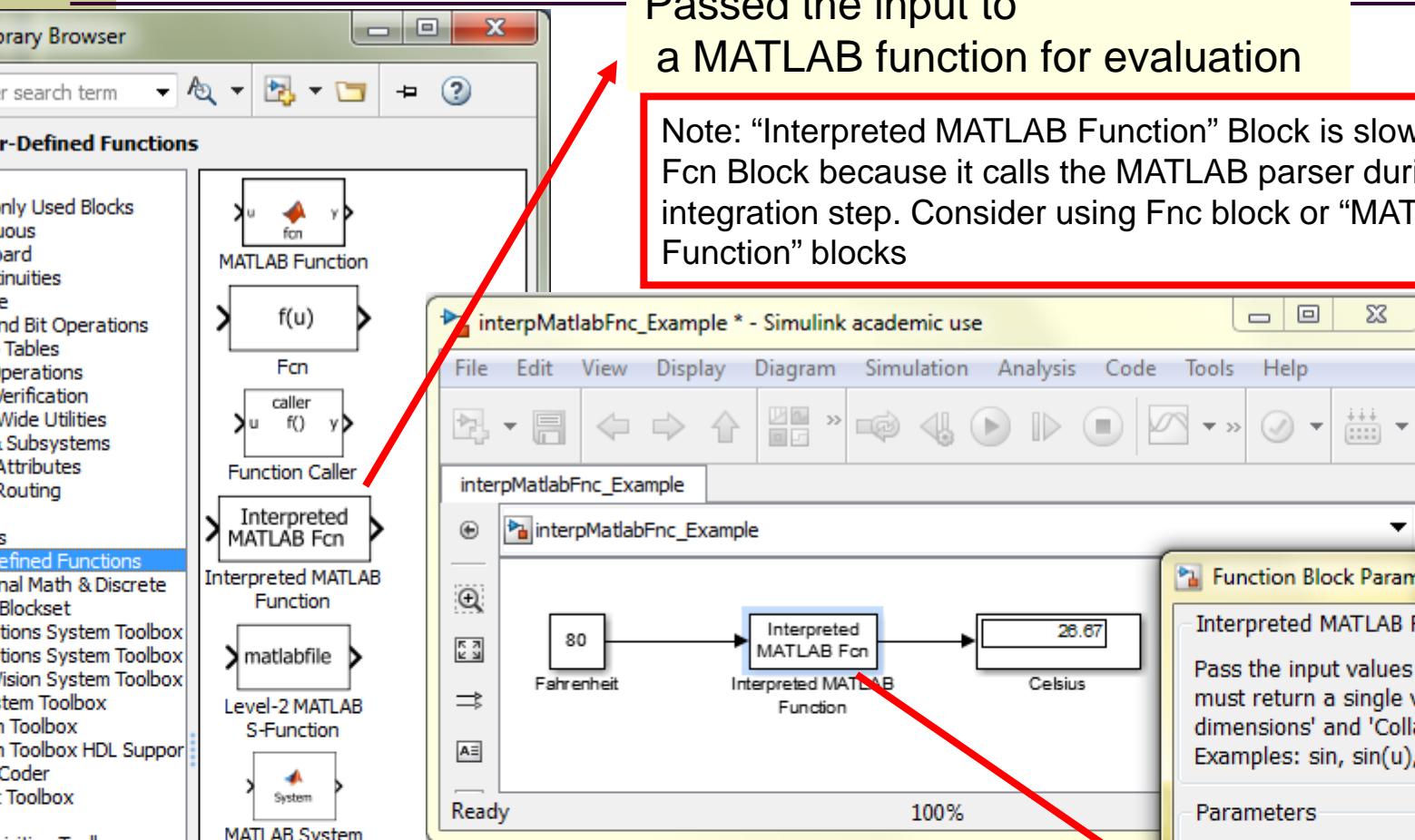
If signal are vectors, index into u.  
 $\sin(u(1)*\exp(2.3*(-u(2))))$

>> fcn\_example

# User Defined Functions

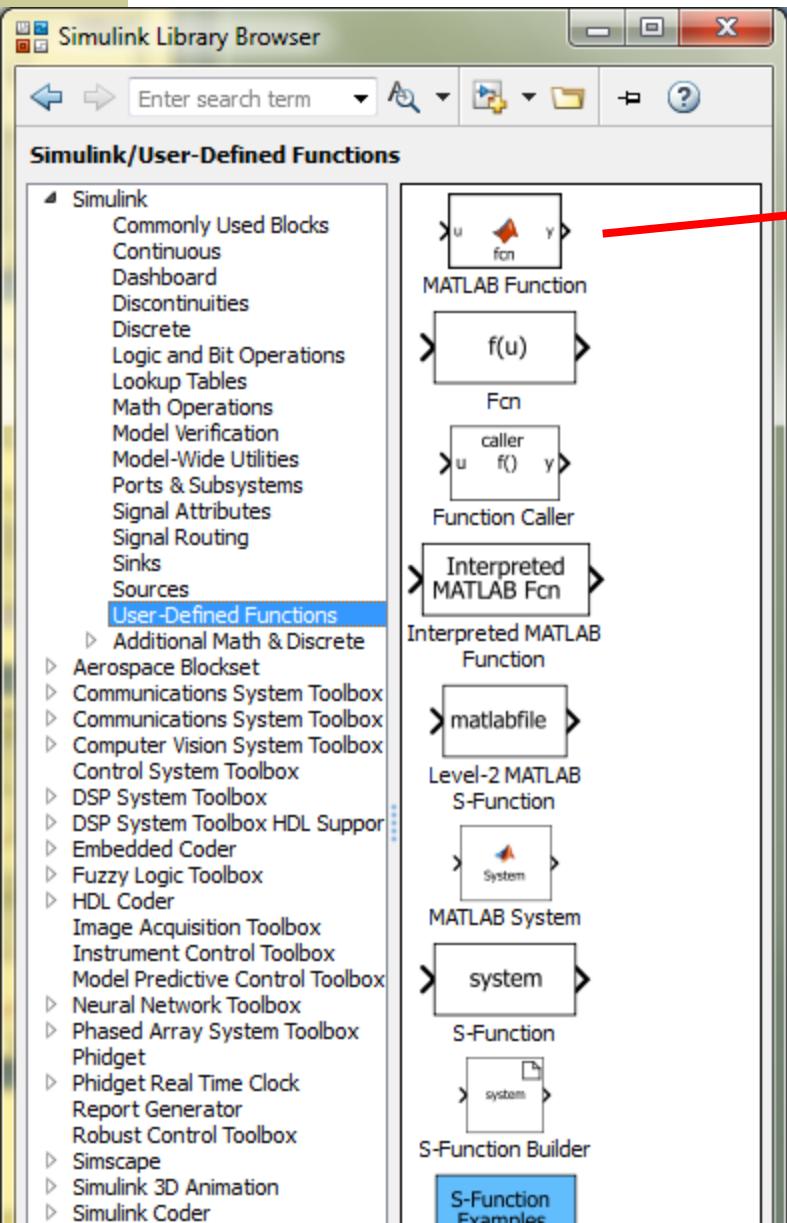
Passed the input to  
a MATLAB function for evaluation

Note: “Interpreted MATLAB Function” Block is slower than the Fcn Block because it calls the MATLAB parser during each integration step. Consider using Fnc block or “MATLAB Function” blocks



```
fah2cel.m x +  
1 %%%%%% Intro 2 UVS  
2 %  
3 % temperature conversion: Fahr  
4 %  
5 function y=fah2cel(x)  
6 - y = 5/9*(x-32);
```

# “MATLAB Function” Blocks



You can write a MATLAB function for use in a Simulink model.

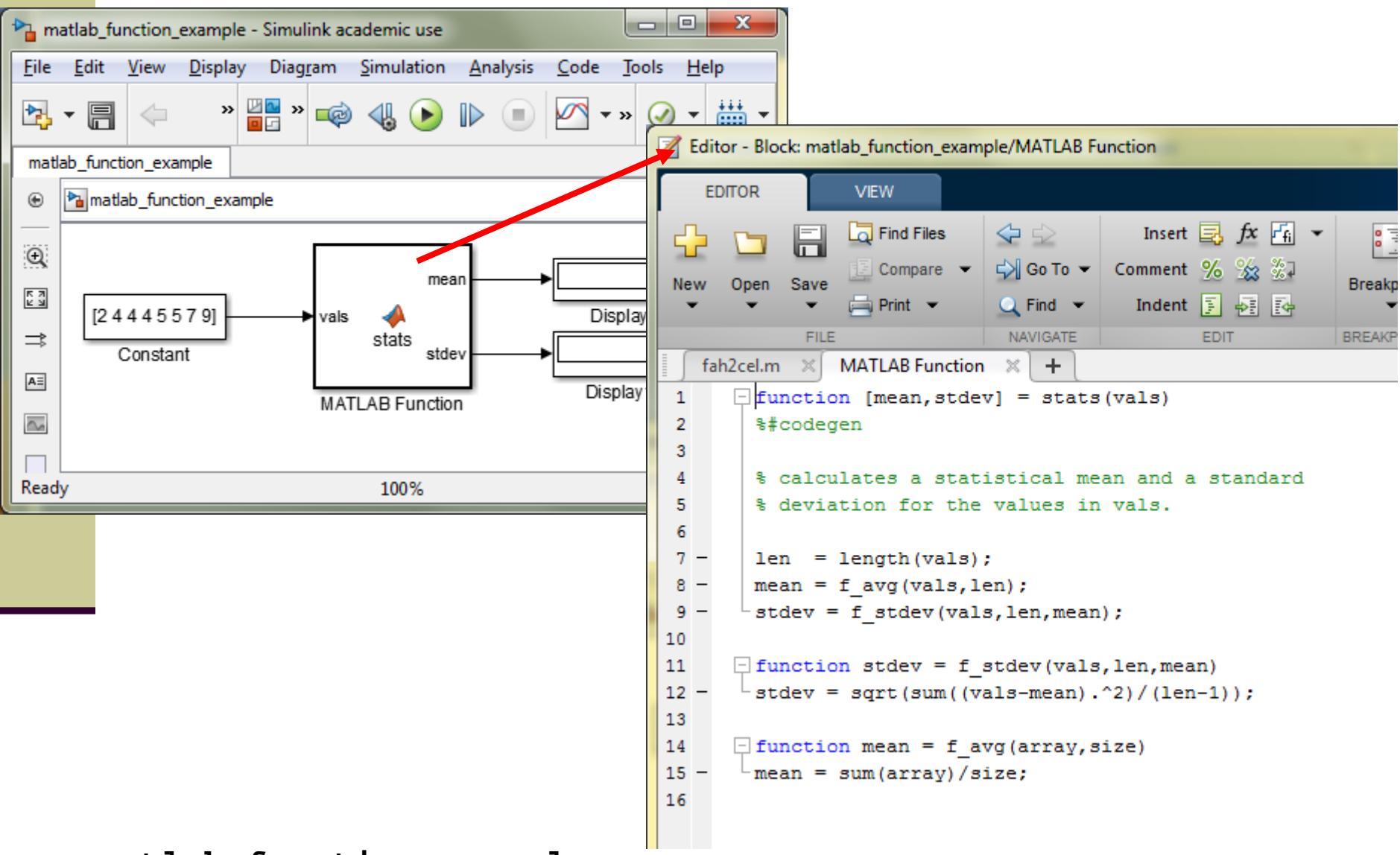
The MATLAB function you create executes for simulation and generates code for a Simulink Coder target.

The MATLAB Function block can generate efficient embeddable code.

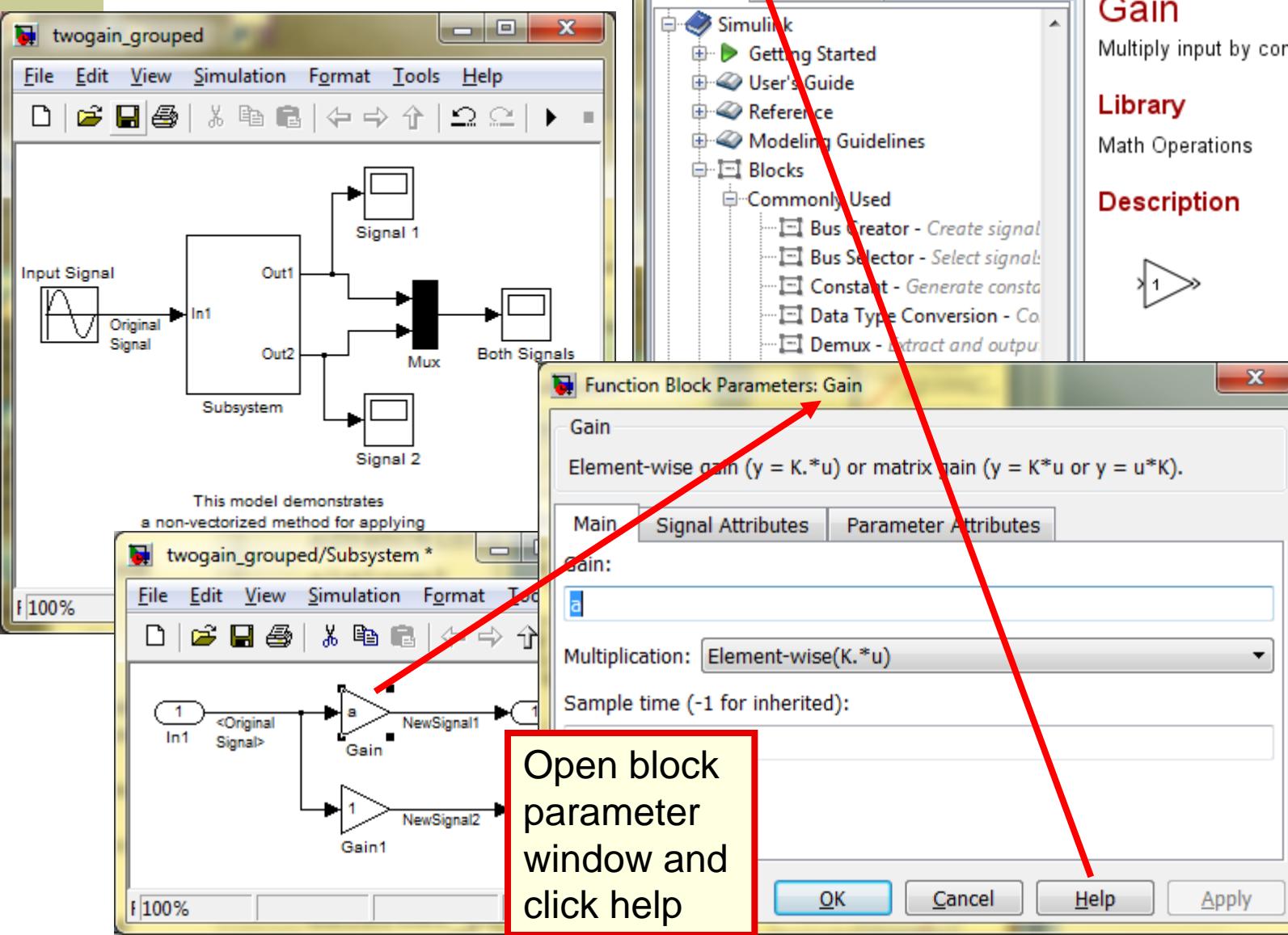
The MATLAB Function block **supports only a subset** of the functions available in MATLAB

`>> matlab_function_example`

# “MATLAB Function” Blocks



# Getting help



# Print Dialog Box

The image shows the MATLAB Simulink interface with the 'twogain\_grouped/Subsystem - Simulink academic use' window open. The 'File' menu is selected, and the 'Print' option is highlighted. A red arrow points from the text 'Select printed systems' to the 'Print Model' dialog box, which is overlaid on the main window. The 'Print Model' dialog box contains settings for printing a subsystem. It includes fields for 'Printer' (set to 'Adobe PDF'), 'Print To File' (set to 'ers\Dogan\Documents\MATLAB\Subsystem.pdf'), orientation options ('Portrait' and 'Landscape' radio buttons), and system selection options ('Current System', 'Current system and above', 'Current system and below', and 'All systems' radio buttons). Below these are checkboxes for 'Enable tiled printing', 'Print sample time legend', 'Include print log', 'Look under mask dialog', 'Expand unique library links', and 'Frame' (set to 'R2015a\toolbox\simulink\simulink\sldefaultframe.fig'). At the bottom are 'Print using system dialog...', 'Print', and 'Cancel' buttons.

twogain\_grouped/Subsystem - Simulink academic use

File Edit View Display Diagram Simulation Analysis Code Tools >

New Open... Ctrl+O

Close

Save Ctrl+S

Save As...

Source Control

Export Model to Reports

Model Properties

Print > Print Ctrl+P

Simulink Preferences

Stateflow Preferences

Exit MATLAB Ctrl+Q

>

Ready 100% ode45

Select printed systems

Print Model

Printer: Adobe PDF

Print To File: ers\Dogan\Documents\MATLAB\Subsystem.pdf ...

Orientation:  Portrait  Landscape

Options:

- Current System
- Current system and above
- Current system and below
- All systems

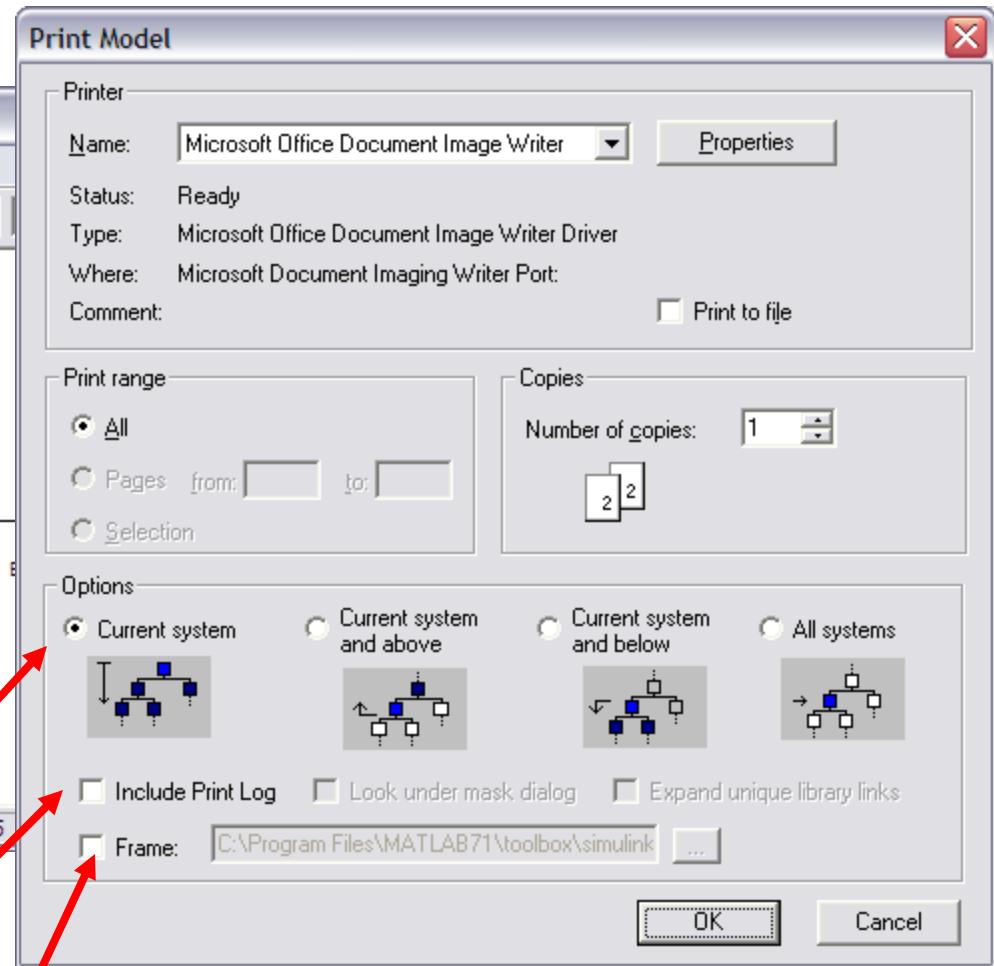
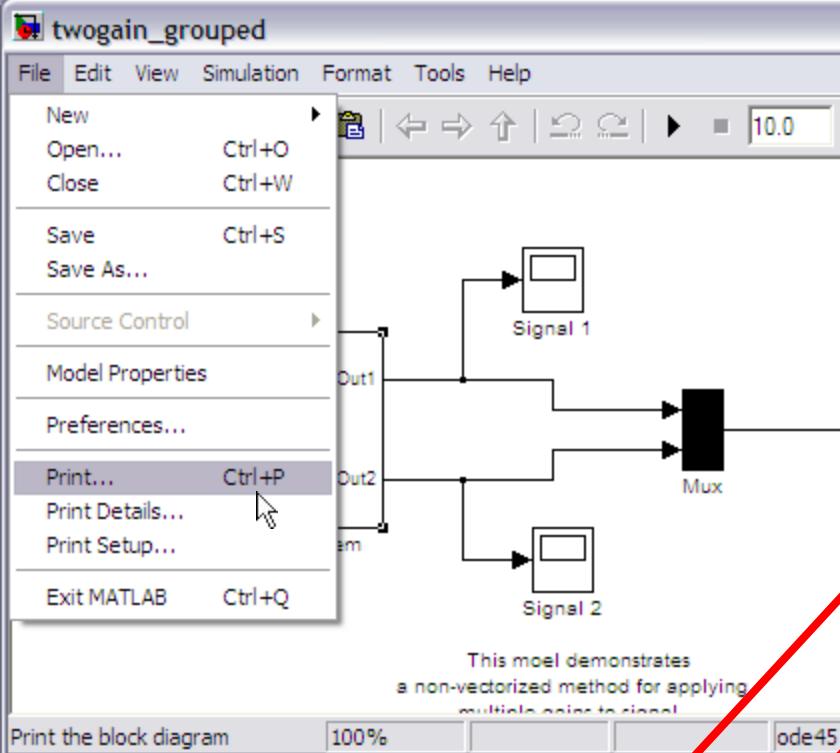
Enable tiled printing  Print sample time legend  Include print log

Look under mask dialog  Expand unique library links

Frame: R2015a\toolbox\simulink\simulink\sldefaultframe.fig ...

Print using system dialog... Print Cancel

# Print Dialog Box



Select printed systems

Include a table of contents

Include a frame

# Tips for building a model

---

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

- Minimize line crossings
- Remove titles when not needed
- Show dialog parameters in title
- Use Goto/From Block wisely
- Size blocks appropriately
- Use subsystems appropriately

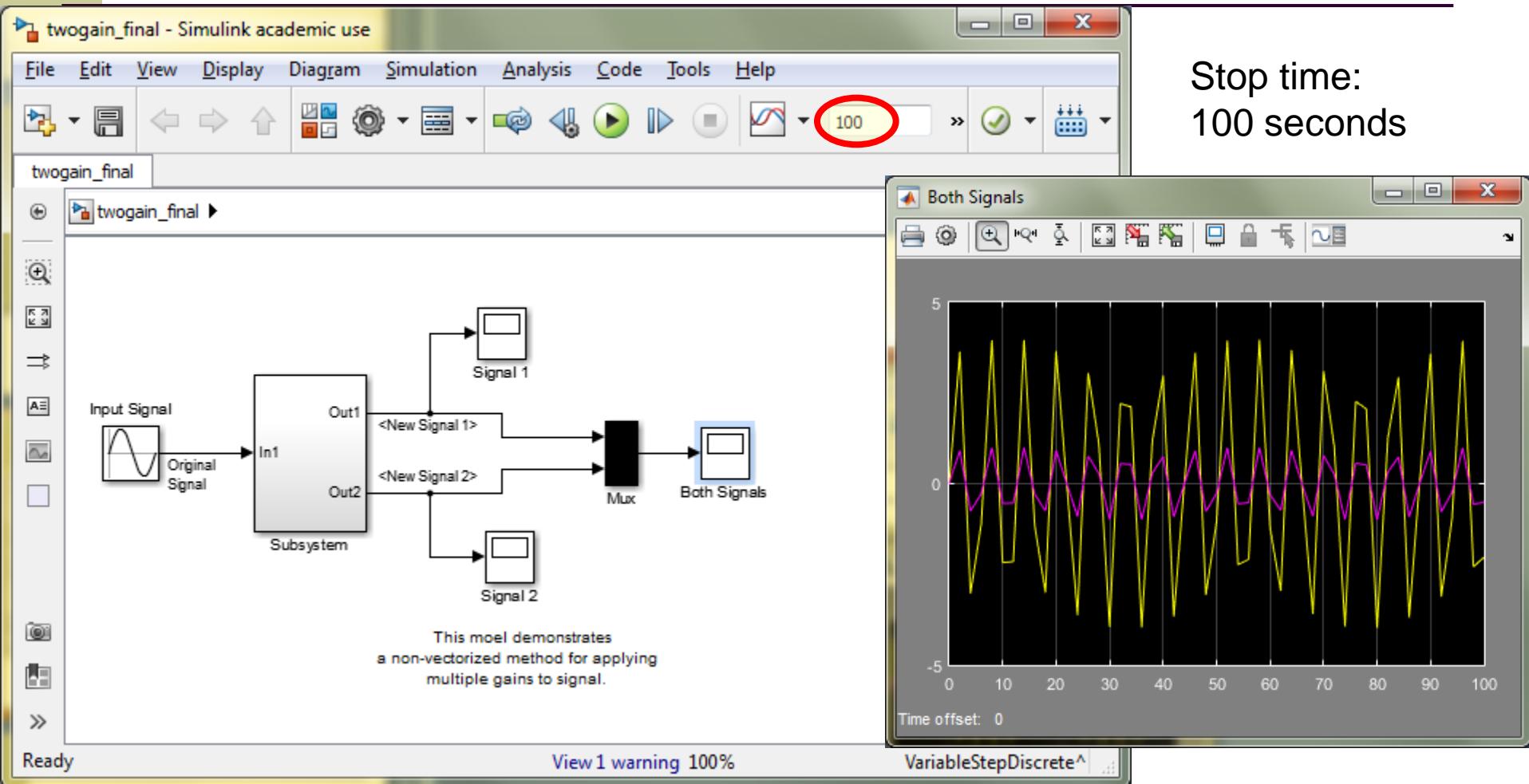
# Creating a model summary

---

You should be able to do the following

- Open a new model window
- Navigate Simulink block libraries
- Annotate and label blocks and signals
- Edit models
- Change block parameters and characteristics
- Create and manage subsystems
- Save and print your models
- Get help on blocks
- Build a sensible model

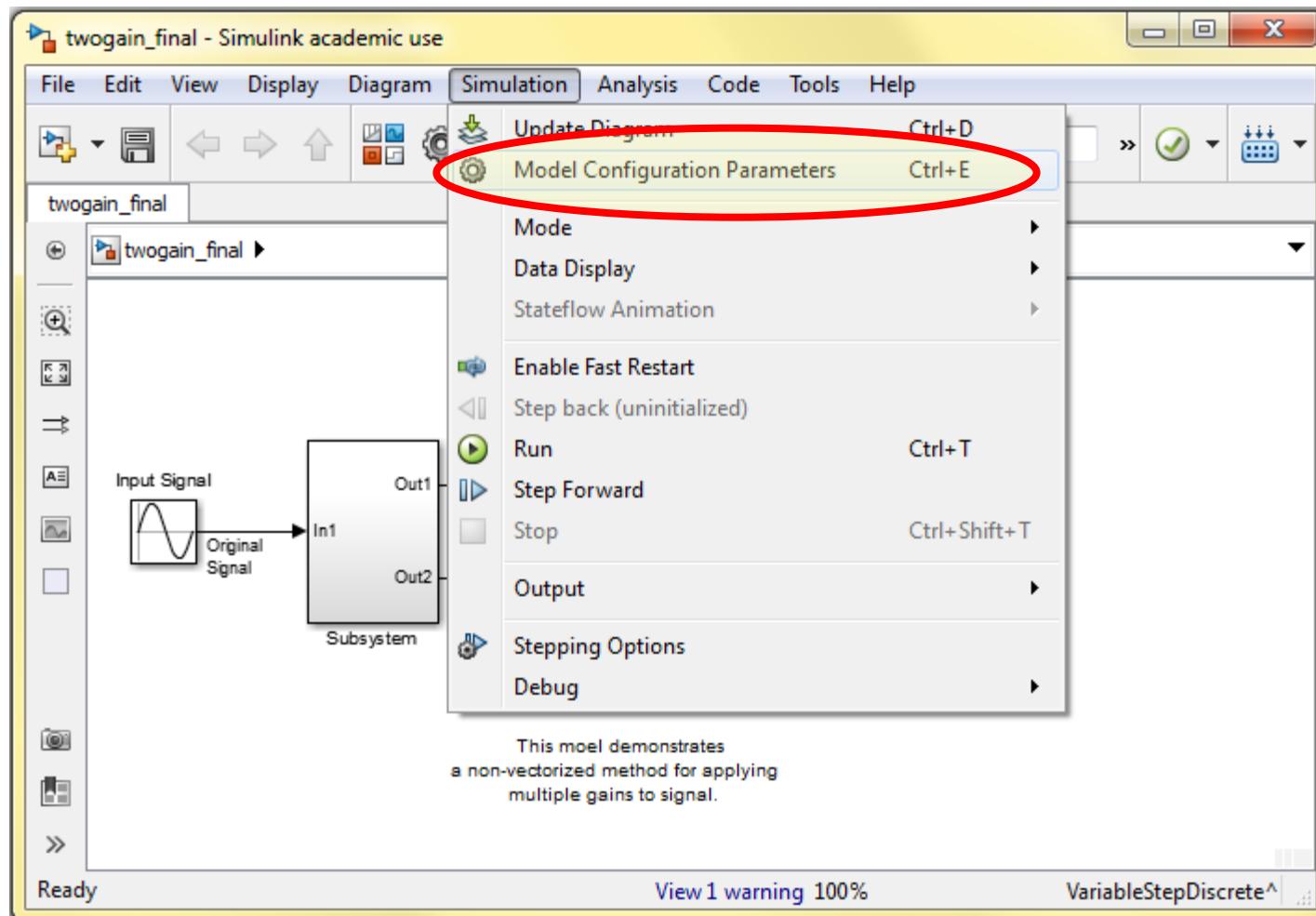
# Simulating



```
>> twogain_final
```

Final simulation: jagged!

# Setting simulation parameters



```
>> twogain_final
```

# Setting simulation parameters

Configuration Parameters: twogain\_final/Configuration (Active)

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation

Simulation time

Start time: 0.0 Stop time: 100

Solver options

Type: Variable-step Solver: discrete (no continuous states)

Max step size: auto

Tasking and sample time options

Tasking mode for periodic sample times: Auto

Automatically handle rate transition for data transfer

Higher priority value indicates higher task priority

Zero-crossing options

Zero-crossing control: Use local settings Algorithm: Nonadaptive

Time tolerance:  $10*128*eps$  Signal threshold: auto

Number of consecutive zero crossings: 1000

Auto in this case means:  
 $h_{max} = (t_{stop} - t_{start})/50$

# Setting simulation parameters

Configuration Parameters: twogain\_final/Configuration (Active)

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation

Simulation time

Start time: 0.0 Stop time: 100

Solver options

Type: Variable-step (highlighted with a red circle)

Solver: discrete (no continuous states) (highlighted with a red circle)

Max step size: 0.1 (highlighted with a red circle and has an arrow pointing to it from the text "Max.Step size = 0.1")

Tasking and sample time options

Tasking mode for periodic sample times:

- Automatically handle rate transition for data transfer
- Higher priority value indicates higher task priority

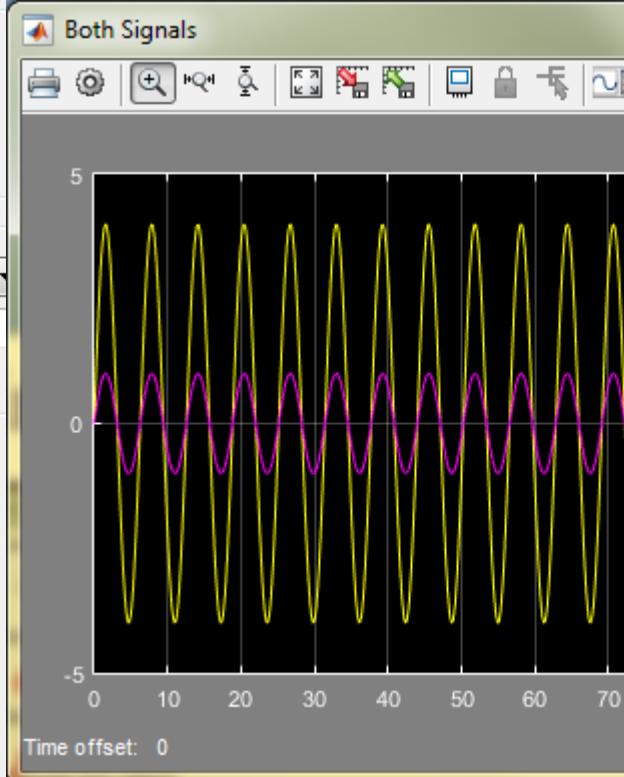
Zero-crossing options

Zero-crossing control: Use local settings

Time tolerance:  $10 * 128 * \text{eps}$

Number of consecutive zero crossings:

Max.Step size = 0.1

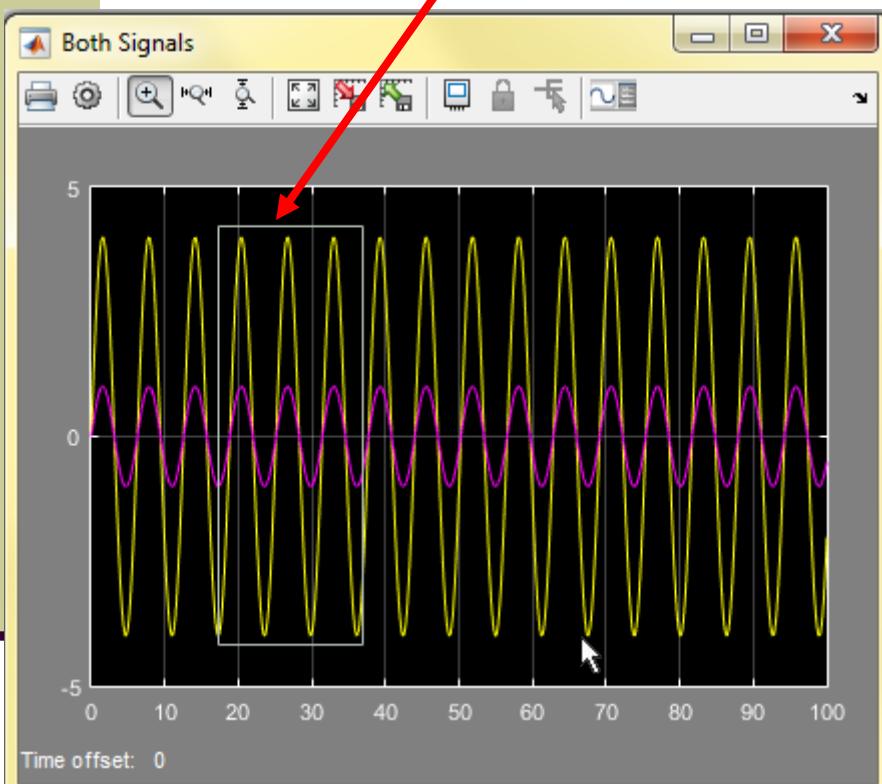


>> twogain\_final

# Simulating

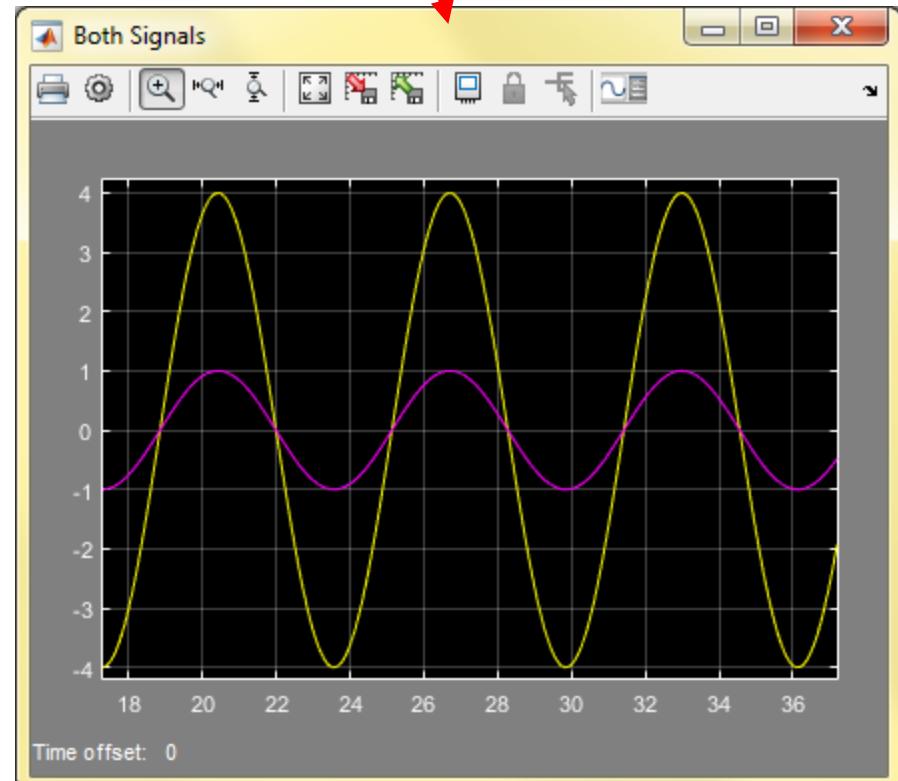
Second Simulation: Much Better...

You can even zoom in to see.



```
>> twogain_final
```

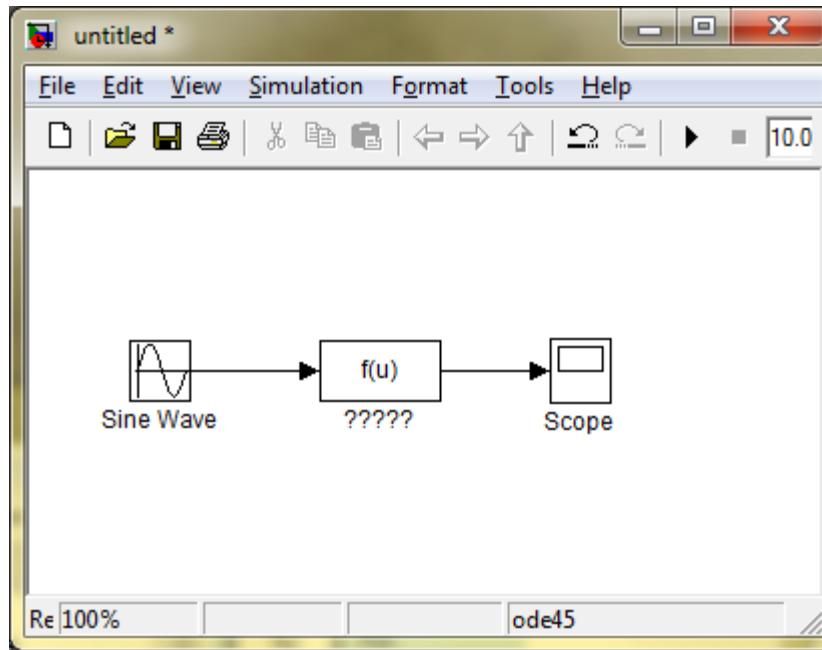
If you're happy with these setting you can save or print them



# Example: Temperature Conversion

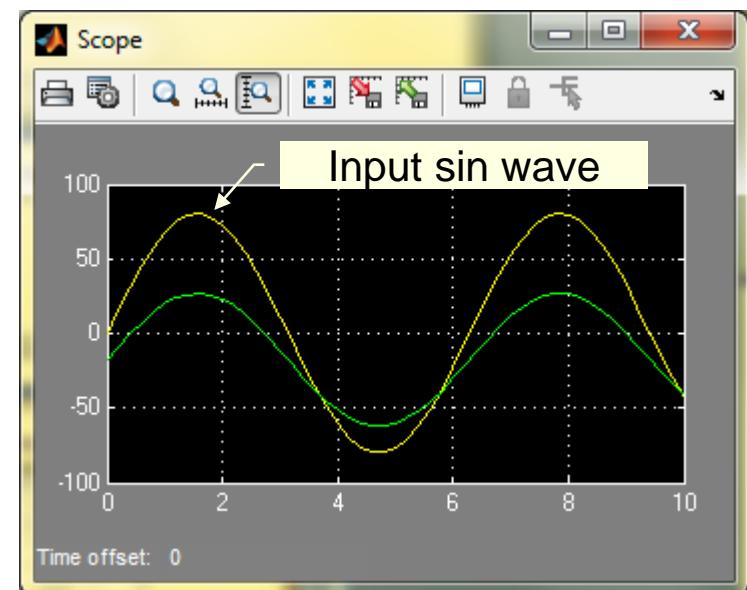
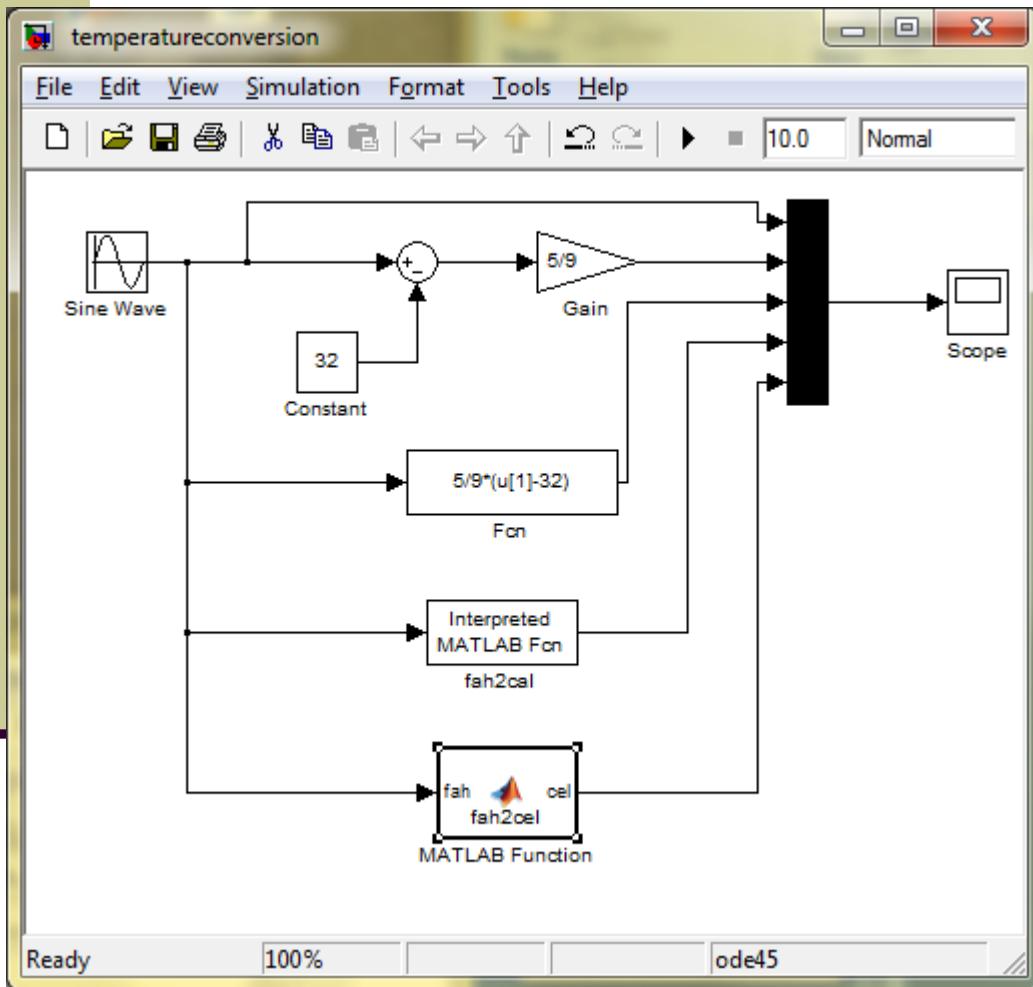
---

$$C = (5/9)*(F-32)$$



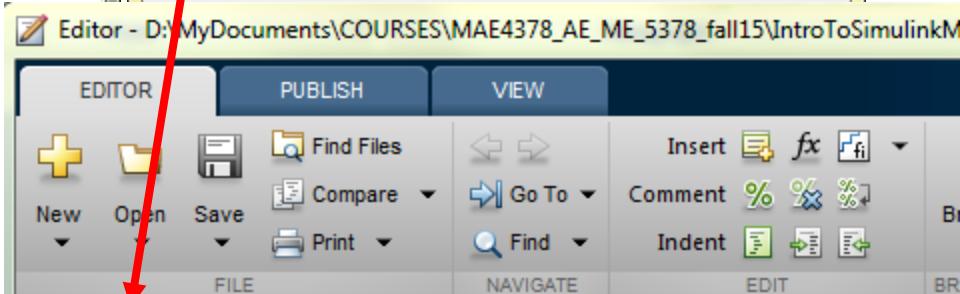
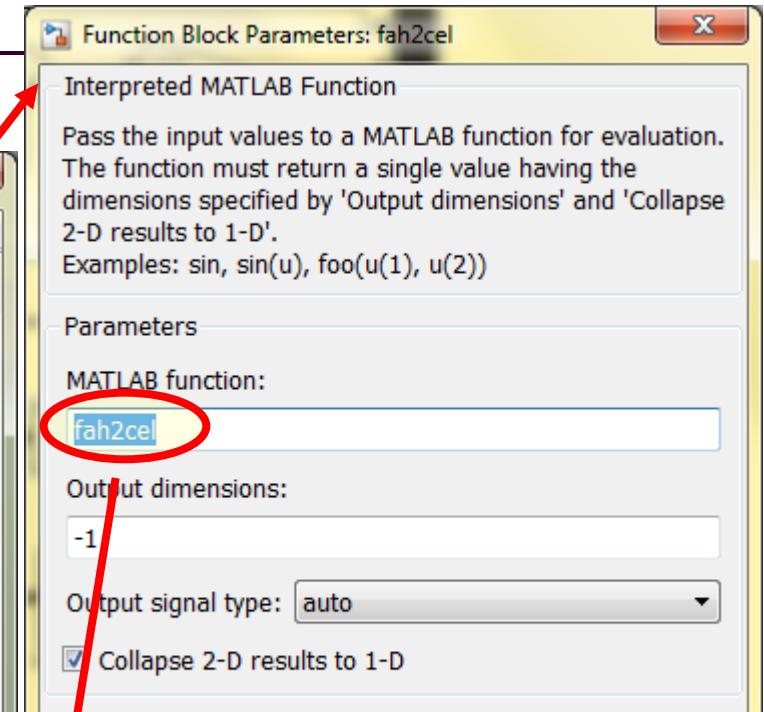
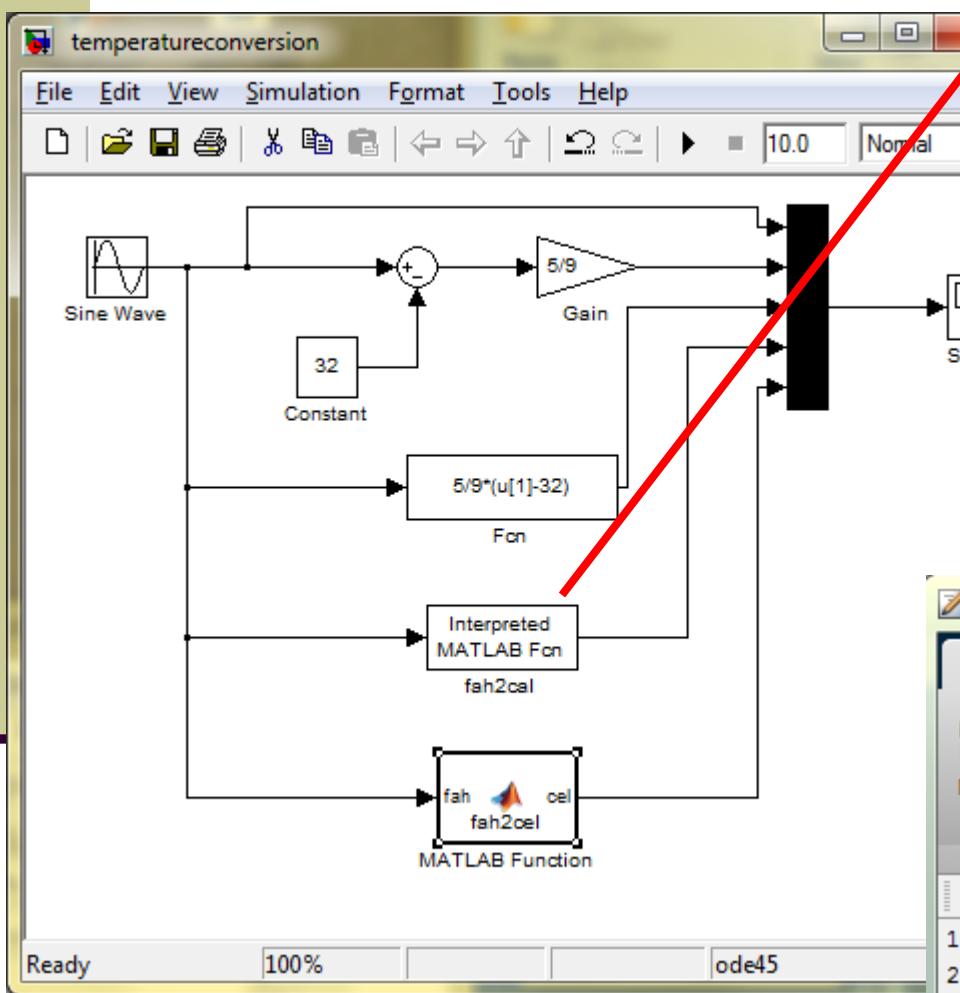
- 1) Determine what blocks are needed to convert the temperature in Simulink?
- 2) Using a sin wave as the input (set amplitude to 80 degF.), compare F vs C on the scope

# Temperature Conversion



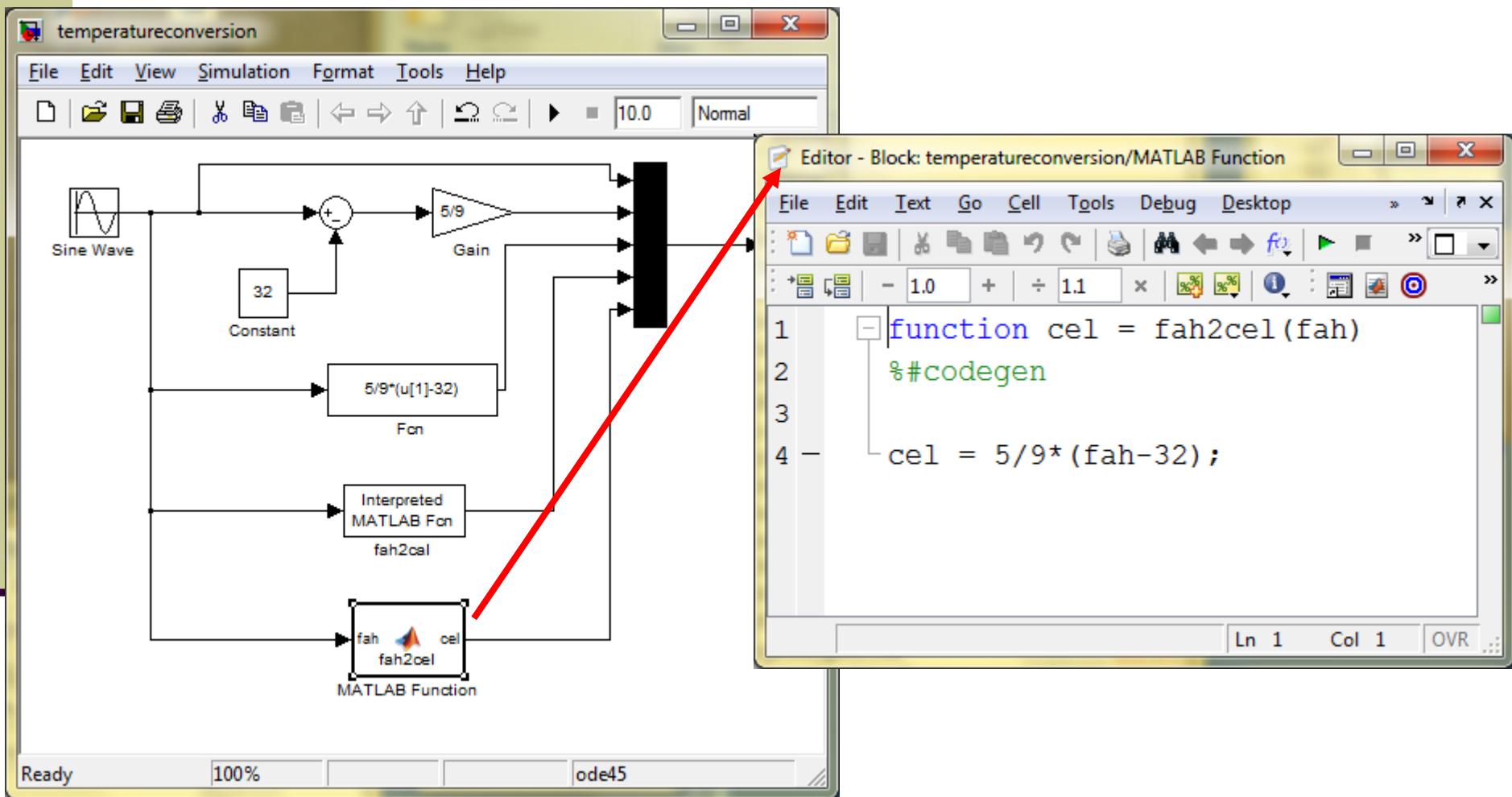
```
>> temperatureconversion
```

# Temperature Conversion



```
>> temperatureconversion
```

# Temperature Conversion



```
>> temperatureconversion
```

# Getting data out of model

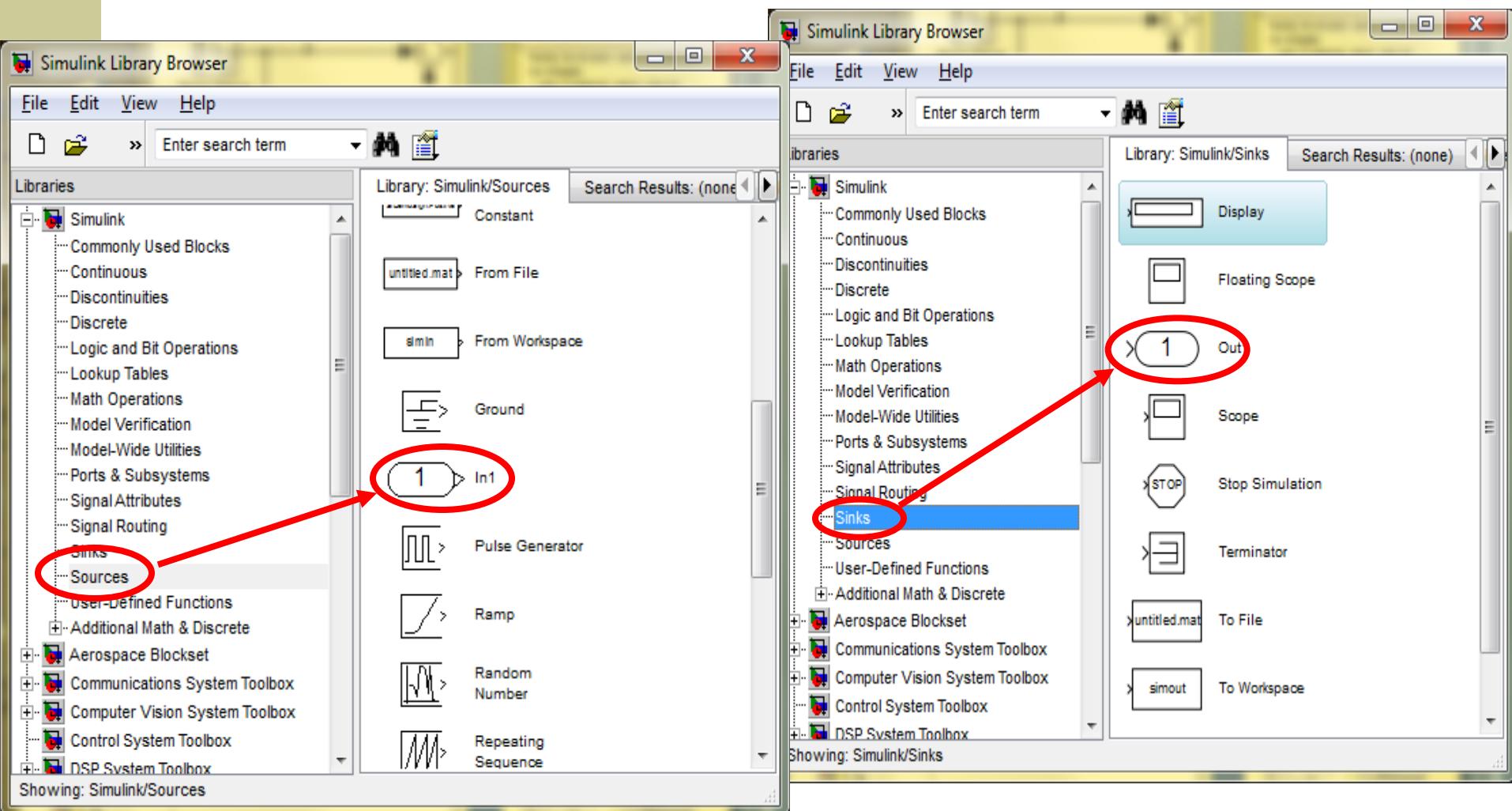
---

Pausing the simulation sends all stored data to the workspace

- ❑ Useful for analyzing results while the simulation is “running”
- ❑ Useful for storing information during different parts of the simulation

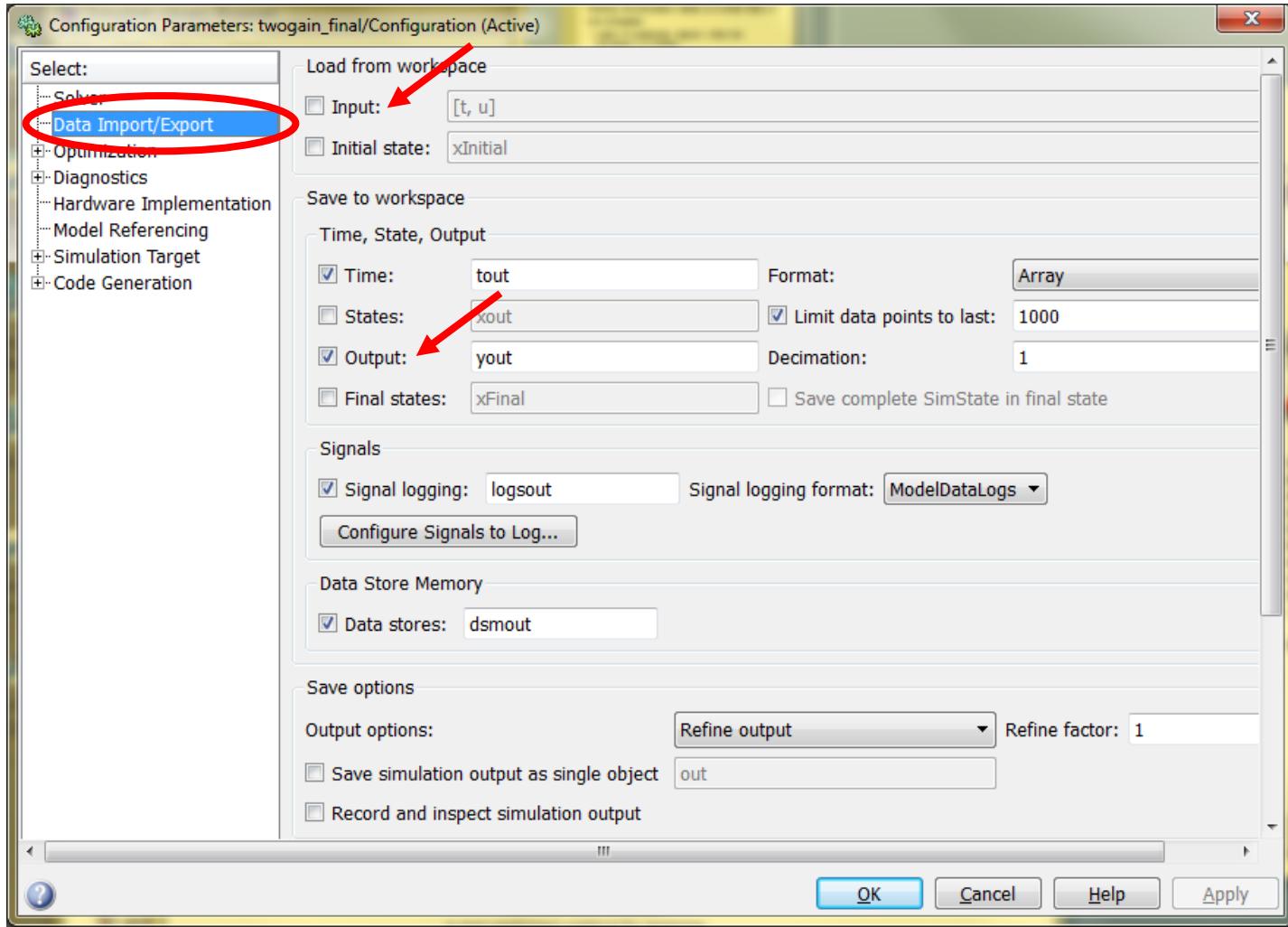
# Getting data into and out of models

Applies to Imports and Outports on top level only

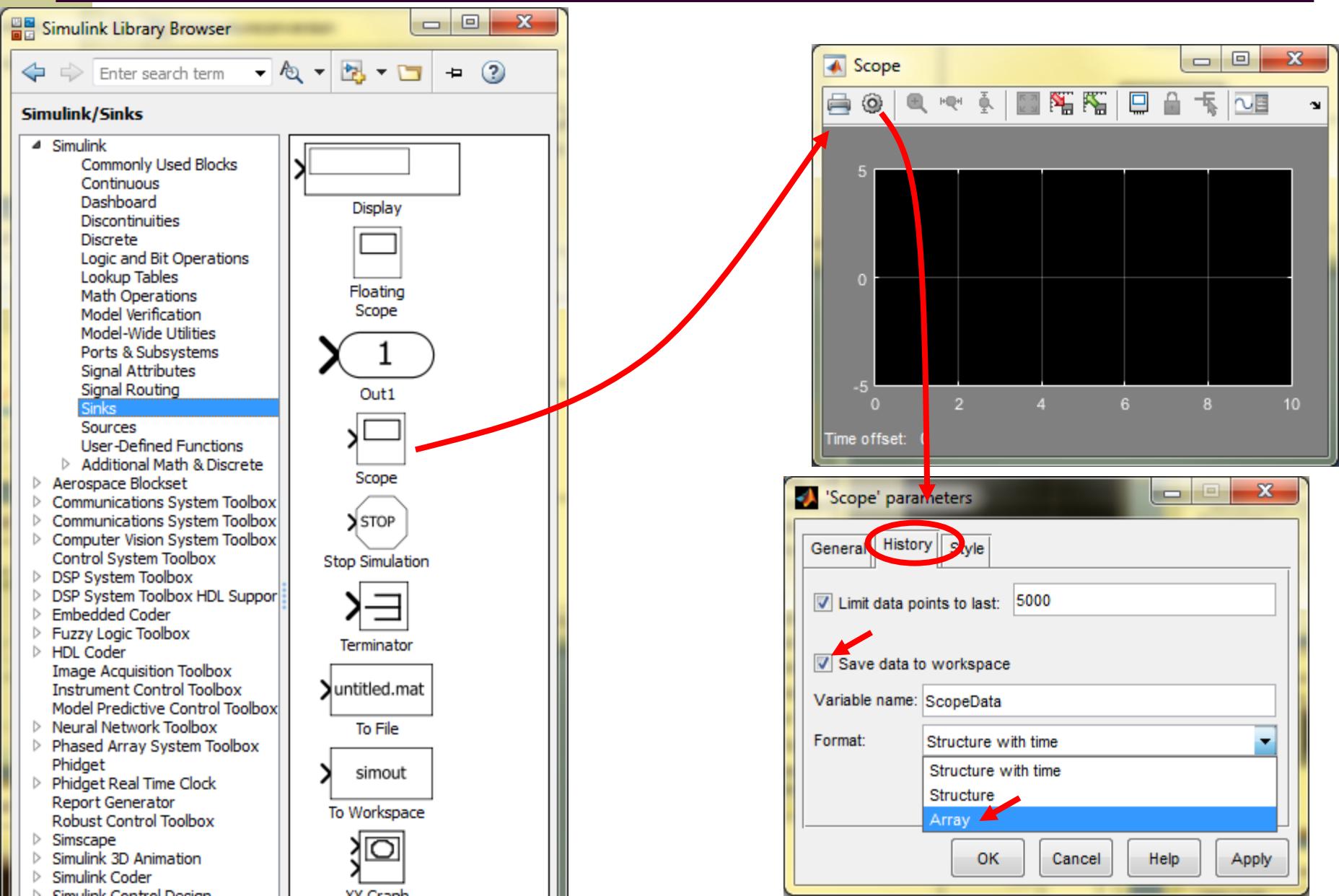


# Getting data into and out of models

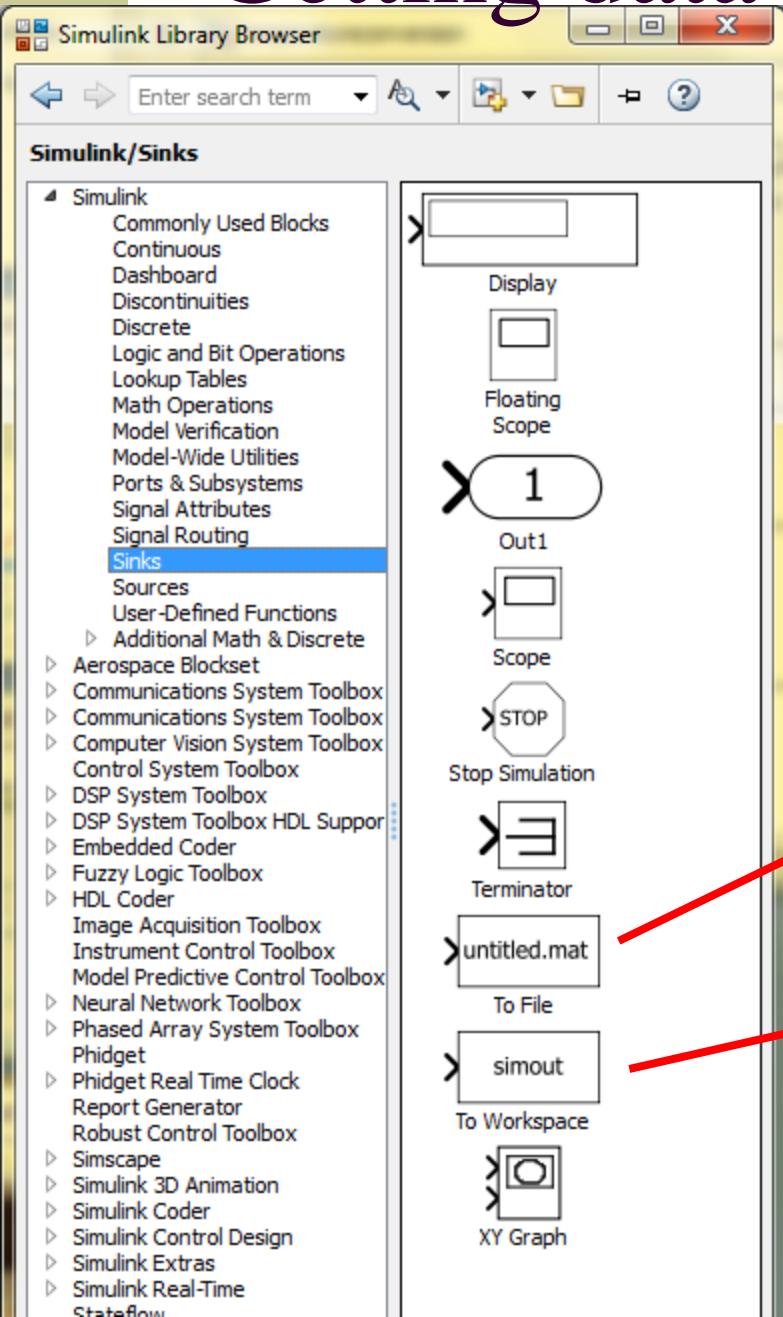
Applies to Imports and Outports on top level only



# Getting data into and out of models



# Getting data into and out of models



to write signal history to a file

to write signal history to Workspace