MATLAB & Simulink Tutorial

16.06 Principles of Automatic Control & 16.07 Dynamics

Violeta Ivanova, Ph.D.
Educational Technology Consultant
MIT Academic Computing

violeta@mit.edu

Information Services and Technology

This Tutorial

- Class materials

  web.mit.edu/acmath/matlab/course16/
- Topics
  - MATLAB Review
  - Exercise 1: Matrices & ODEs
  - Introduction to Simulink
  - Exercise 2: Simulink Model

Other References

- Mathematical Tools at MIT

  web.mit.edu/ist/topics/math

  - MATLAB Mastery I (beginners' tutorial)
  - Introduction to MATLAB (IAP series)
- MATLAB Tutorial for Unified

  web.mit.edu/acmath/matlab/unified

MATLAB Review

Interface

Matrices & Vectors

Built-In Functions

Script & Function M-files

Differential Equations

What is MATLAB?

- Computational Software

  From The MathWorks: www.mathworks.com
- Algorithm Development Environment
- MATrix LABoratory

MATLAB @ MIT

- On Athena
  - 250 floating licenses (free)
- For student-owned computers
  - 300 floating licenses (free)

Starting MATLAB

- On Athena

```
athena% add matlab
athena% matlab &
```

```
>> desktop
```

- On laptops

  Desktop interface starts by default.

You must be running MATLAB now …

Help in MATLAB

- Command line help

```
>> help <command>
          e.g. help polyval
>> lookfor <keyword>
          e.g. lookfor polynomial
```

- Help Browser
  - Help->Help MATLAB

Variables

- Begin with an alphabetic character: a
- Case sensitive: `a, A`
- Data type detection: `a=5; a='ok'; a=1.3`
- Default output variable: `ans`
- Built-in constants: **`pi i j Inf`**
- **`clear`** removes variables
- **`who`** lists variables
- Special characters

  `[] () {} ; % : = . … @`

Operators

- Arithmetic operators

  `+   -   /   \   ^   .\   ./   .*   .^`

- Relational operators

  `<   >   <=   >=   ==   ~=`

- Logical operators

  `|   &   ||   &&   true   false`

- Operator precedence

  `()  {}  []` -> Arithmetic -> Relational -> Logical

- Do not use special characters, operators, or keywords in variable names.

Numeric Data Types

- Notation

```
>> x = 5;
>> y = 5.34;
>> z = 0.23e+3;
```

- Numeric manipulation

```
>> y = 5.3456; x = round(y);
>> format long
>> format compact
```

- Complex numbers

```
>> x = 5 + 6i
```

Vectors

- Row vector

```
>> R1 = [1 6 3 8 5]
>> R2 = [1 : 5]
>> R3 = [-pi : pi/3 : pi]
```

- Column vector

```
>> C1 = [1; 2; 3; 4; 5]
>> C2 = R2'
```

Matrices

- Creating a matrix

```
>> A = [1 2.5 5 0; 1 1.3 pi 4]
>> A = [R1; R2]
```

- Accessing elements

```
>> A(1,1); A(1:2, 2:4); A(:,2)
```

Input / Output

- Import Wizard for data import

  ```
  File->Import Data …
  ```

- File input with `load`

  ```
  B = load(‘datain.txt’)
  ```

- File output with `save`

  ```
  save(‘dataout’, ‘A’, ‘-ascii’)
  ```

Matrix Operations

- Operators *, /, and ^

  `>> Ainv = A^-1` Matrix math is default!

- Operators + and -

  >> X = [x_1 x_2 x_3];
  >> Y = [y_1 y_2 y_3];

  ```
  >> A = X + Y
     A =
  ```

  x_1+y_1 x_2+y_2 x_3+y_3

Element-Wise Operations

- Operators .*, ./, and .^

```
>> Z = [$z_1$ $z_2$ $z_3$]′
>> B = [Z.^2   Z  ones(3,1)]
B =
     $z_1^2$   $z_1$   1
     $z_2^2$   $z_2$   1
     $z_3^2$   $z_3$   1
```

Built-In Functions

- Matrices & vectors

```
>> [n, m]= size(A)
>> n = length(X)
>> M1 = ones(n, m)
>> M0 = zeros(n, m)
>> En = eye(n); N1 = diag(En)
>> [evals, evecs] = eig(A)
>> det(A); rank(A); trace(A)
```

- And many others …

```
>> y = exp(sin(x)+cos(t))
```

Polynomials

- Evaluating polynomials

$$y = p_1 x^n + p_2 x^{n-1} \ldots + p_n x + p_{n+1}$$

```
>> p = [p1 p2 … ]

>> t = [-3 : 0.1 : 3]

>> z = polyval(p, t)
```

- Curve fitting

```
>> X = [x1 x2 … xn]; Y = [y1 y2 … yn]

>> Pm = polyfit(X, Y, m)
```

Integration & Differentiation

- Polynomial integration

$$\int p_1 x^n + \ldots + p_n x + p_{n+1} dx = P_1 x^{n+1} + \ldots + P_{n+1} x + C$$

`>> P = polyint(p);` assumes C = 0

- Area under a curve from a to b

```
>> area = polyval(P,b) - polyval(P,a)
```

- Polynomial differentiation

```
>> P = [P1 P2 … Pn C]
>> p = polyder(P)
```

2D Linear Plots

- Command `plot`

```
>> plot (X, Y, ‘ro’)
>> plot (X, Y, ‘Color’, [0.5 0 0], …
             ‘Marker’, ‘o’, …
             ‘LineStyle’, ‘none’)
```

- Colors: `b, r, g, y, m, c, k, w`
- Markers: `o, *, ., +, x, d`
- Line styles: `-, --, -., :`

Multiple Graphs on One Plot

- Built-in function hold

```
>> p1 = plot(t, z, ‘r-’)
>> hold on
>> p2 = plot(t, -z, ‘b--’)
>> hold on
>> p3 = plot(T, Z, ‘go’)
>> hold off
```

Subplots on One Figure

- Built-in function `subplot`

```
>> s1 = subplot(1, 3, 1)
>> p1 = plot(t, z, ‘r-’)
>> s2 = subplot(1, 3, 2)
>> p2 = plot(t, -z, ‘b--’)
>> s3 = subplot(1, 3, 3)
>> p3 = plot(T, Z, ‘go’)
```

Customizing Graphs

- Annotating graphs

```
>> plot (t, z, ‘r-’)
>> legend (‘z=f(t)’)
>> title (‘Position vs. Time’)
>> xlabel (‘Time’)
>> ylabel (‘Position’)
```

- Plot Edit mode: icon ↖ in Figure editor
- Property Editor: `View->Property Editor`
- Saving figures: `File->Save As`

M-File Programming

- Script M-Files
  - Automate a series of steps.
  - Share workspace with other scripts and the command line interface.
- Function M-Files
  - Extend the MATLAB language.
  - Can accept input arguments and return output arguments.
  - Store variables in internal workspace.

A MATLAB Program

- Always has one script M-File
- Uses built-in functions as well as new functions defined in function M-files
- Saved as `<filename>.m`
- To run: filename only (no `.m` extension)

  ```
  >> <filename>
  ```
- Created in Editor / Debugger

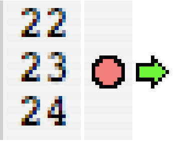
M-File Editor / Debugger

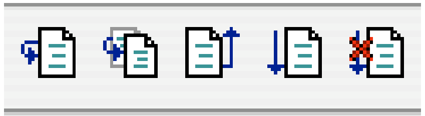
- Create or open M-file in editor

  ```
  >> edit <filename>.m
  ```

- Type or copy commands
- Use **%** for comments
- Use **;** to suppress output at runtime
- Debugging mode

  ```
  k >>
  ```





Variable Types

- Local (default)
  - Every function has its own local variables.
  - Scripts share local variables with functions they call and with the base workspace.
- Global

  ```
  global speedoflight
  ```

  - Shared by functions, scripts, and base workspace.
- Persistent

  ```
  persistent R, C
  ```

  - Can be declared and used only in functions.

Program Flow Control

- **if**, **elseif** and **else**

  Example:

  ```
  if         planet == 1, G = 9.814;
  elseif     planet == 2, G = 3.688;
  else       G  = input(‘Gravity: ’);
  end
  ```

- **switch** and **case**
- **for**
- **while**

Function M-File Example

See file: `odeLanderVelocity.m`

```
function DV = odeLanderVelocity(t, V)
% ODELANDERVELOCITY defines dV/dt for a Mars lander.
% This is help text for “help odeLanderVelocity”.

% The function’s body is below.
Gm = 3.688;
global K M
DV = Gm - K/M * V^2;
return
```

Differential Equations

- Ordinary Differential Equations

$$y' = f(t,y)$$

- Differential-Algebraic Expressions

$$M(t,y)y' = f(t,y)$$

- Solvers for ODEs and DAEs

```
>> ode45; ode23; ode113 …
```

ODE and DAE Solvers

```
>> [T, Y] = solver(odefun, tspan, Y0)
```

- Syntax:
  - `solver`: `ode45`, `ode23`, etc.
  - `odefun`: function handle
  - `tspan`: interval of integration vector

    ```
    >> tspan = [t0 : tstep : tfinal]
    ```
  - `Y0`: vector of initial conditions
  - `[T, Y]`: numerical solution in two vectors

ODE Example

- Problem: $$\frac{dv(t)}{dt} = g - \frac{k}{m}v^2$$
- Solution:

```
global K M
K = 1.2;          % drag coefficient
M = 150;          % mass (kg)
V0 = 67.056;      % velocity at t0 (m/s)

tspan = [0 : 0.05 : 6];
[t, V] = ode45(@odeLanderVelocity, ...
                tspan, V0)
```

Symbolic Math Toolbox

- Incorporates symbolic computations into MATLAB’s numerical environment
- Functions access the Maple kernel
- Constructs for symbolic values & expressions

```
>> x = sym(‘x’)
>> f = sym(‘cos(t)’)
>> syms a b c
```

Laplace Transforms

- Definition: $F(s) = L\{f(t)\} = \int_{0}^{\infty} e^{-st} f(t)\,dt$
- Examples:
  - Laplace transform of $f(t) = \sin(t)$

```
>> f = sym('sin(t)')
>> F = laplace(f)
```

  - Inverse Laplace transform of $G(s) = \frac{0.1}{0.1s + 1}$

```
>> G = sym('0.1/(0.1*s+1)')
>> g = ilaplace(G)
```

Transfer Functions

- System of linear differential equations
- State Space model

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

X, u & Y: state, input & output vectors

A, B & C: state, input & output matrices

D: usually zero (feedthrough) matrix

- Transfer function

$$H(s) = \frac{Num(s)}{Den(s)} = C(sI - A)^{-1}B = D$$

```
>> [Num, Den] = ss2tf(A, B, C, D)
```

Exercise 1: Matrices & ODEs

- 1-A: Mars Lander Velocity
  - Function file: `odeLanderVelocity.m`
  - Script file: `MarsLander.m`

- 1-B: F-8 Longitudinal Time Response
  - Function file: `LongTimeResponse.m`
  - Script file: `f8long.m`

Follow instructions in exercise handout …

Introduction to Simulink

Interface

Models

Blocks

Simulations

What is Simulink?

- Software for modeling, simulating, and analyzing dynamic systems
- Tool for model-based design
- MATLAB Toolbox -> access to all MATLAB functions

Simulink @ MIT

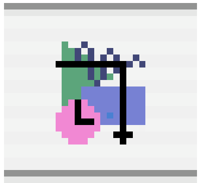
- Comes with MATLAB
- On Athena
  - 50 floating licenses (free)
- For student-owned computers
  - 50 floating licenses (free)
  - Student MATLAB Lite includes MATLAB, Simulink, Control System, Optimization, Signal Processing, Symbolic Math, Statistics

Starting Simulink

- *Run MATLAB first …*
- Type in the Control Line Window

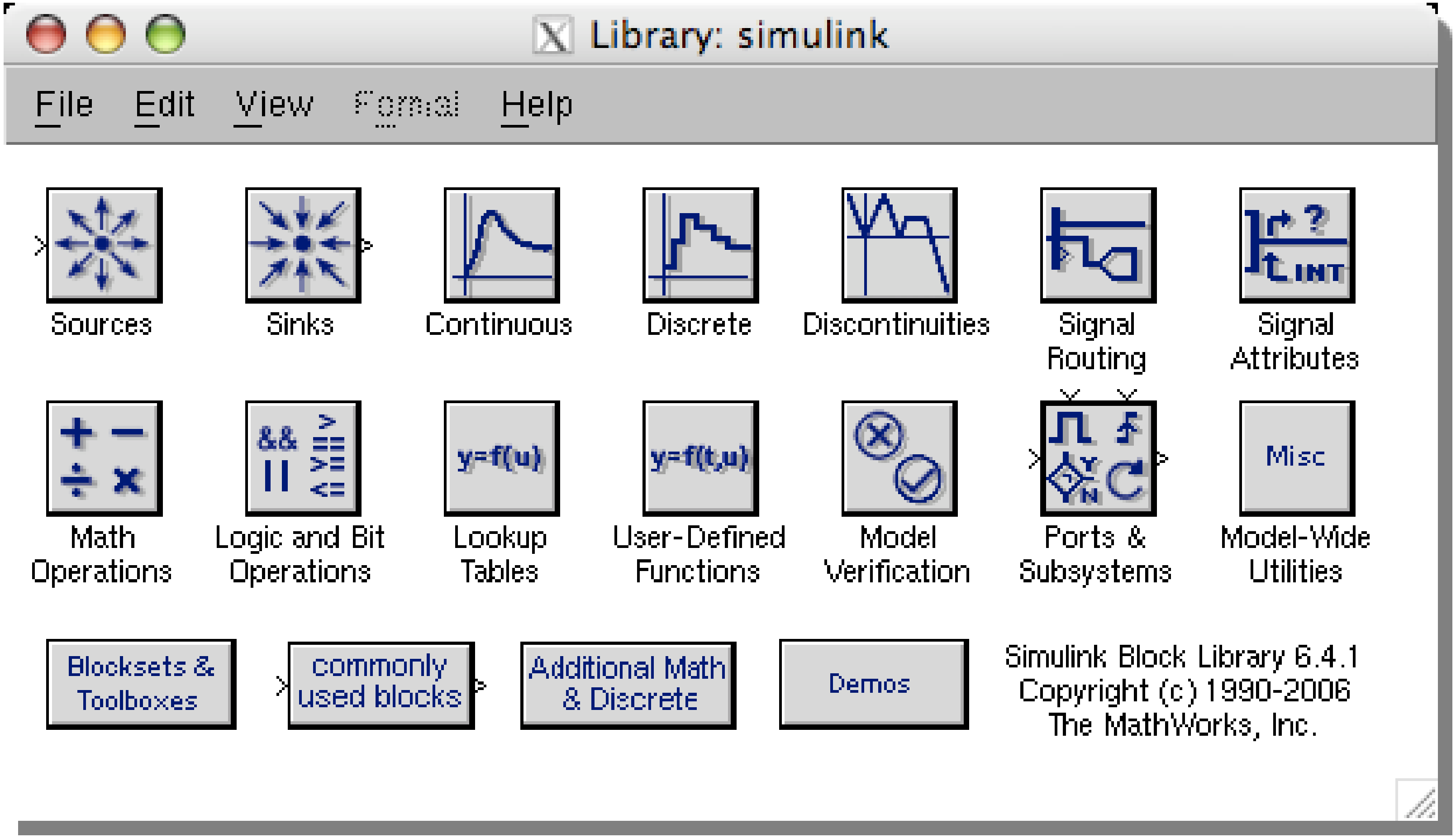
  `>>  simulink`

  or …
- Click on the Simulink icon in the MATLAB toolbar



You must be running Simulink now …

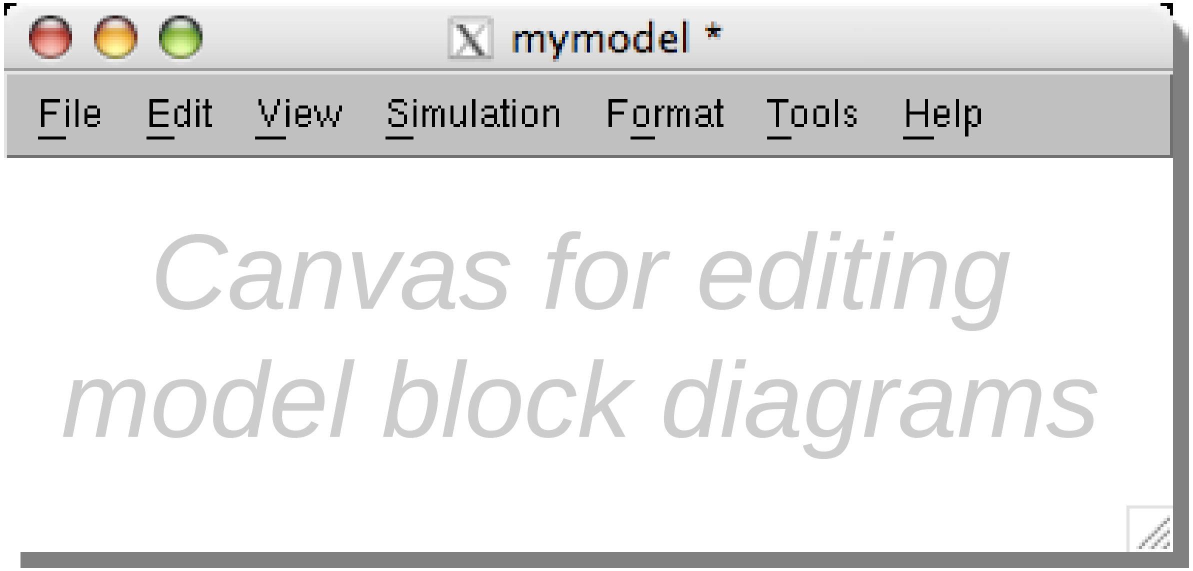
Simulink Libraries



Simulink Block Library 6.4.1
Copyright (c) 1990-2006
The MathWorks, Inc.

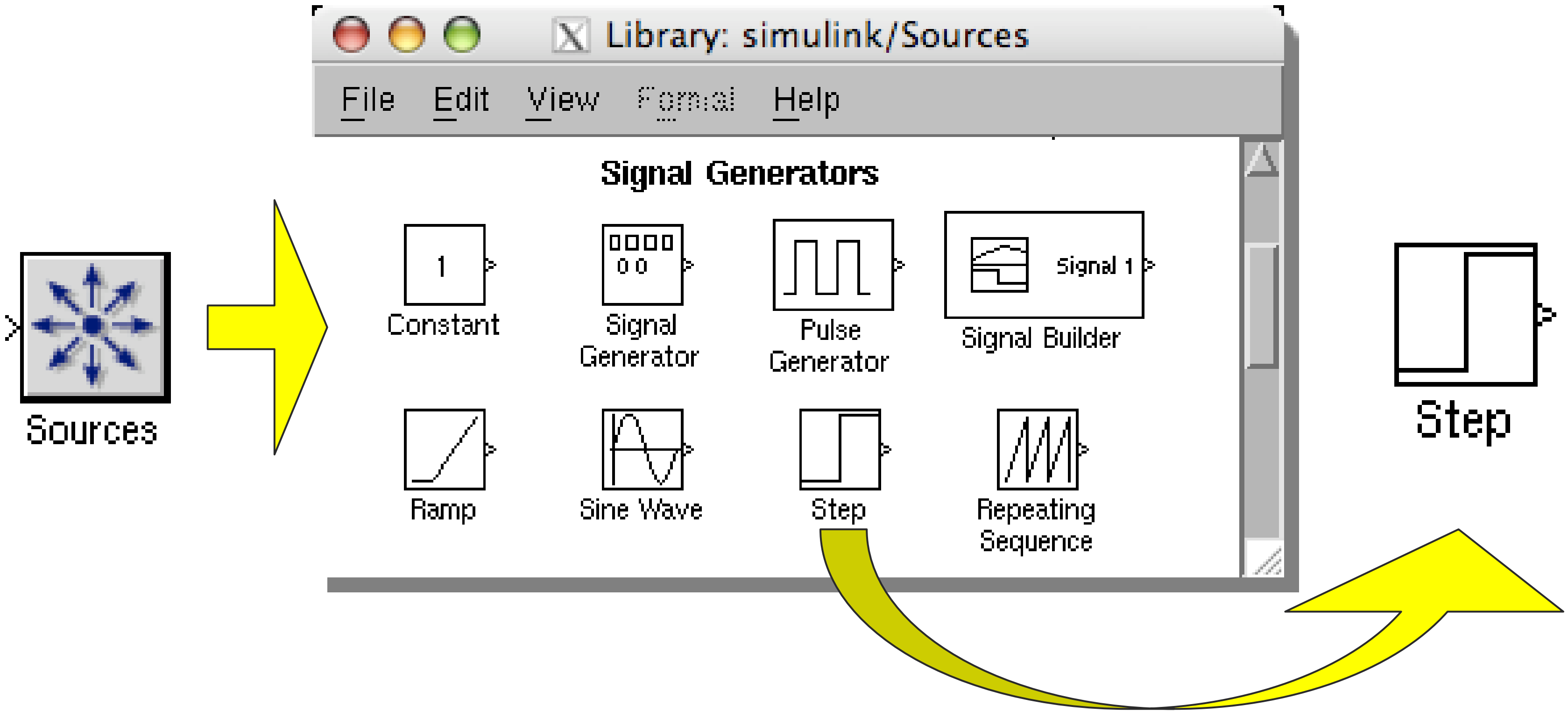
Model Editor

- Creating a model: `File->New->Model`
- Saving a model: `File->Save As` `<modelname>.mdl`



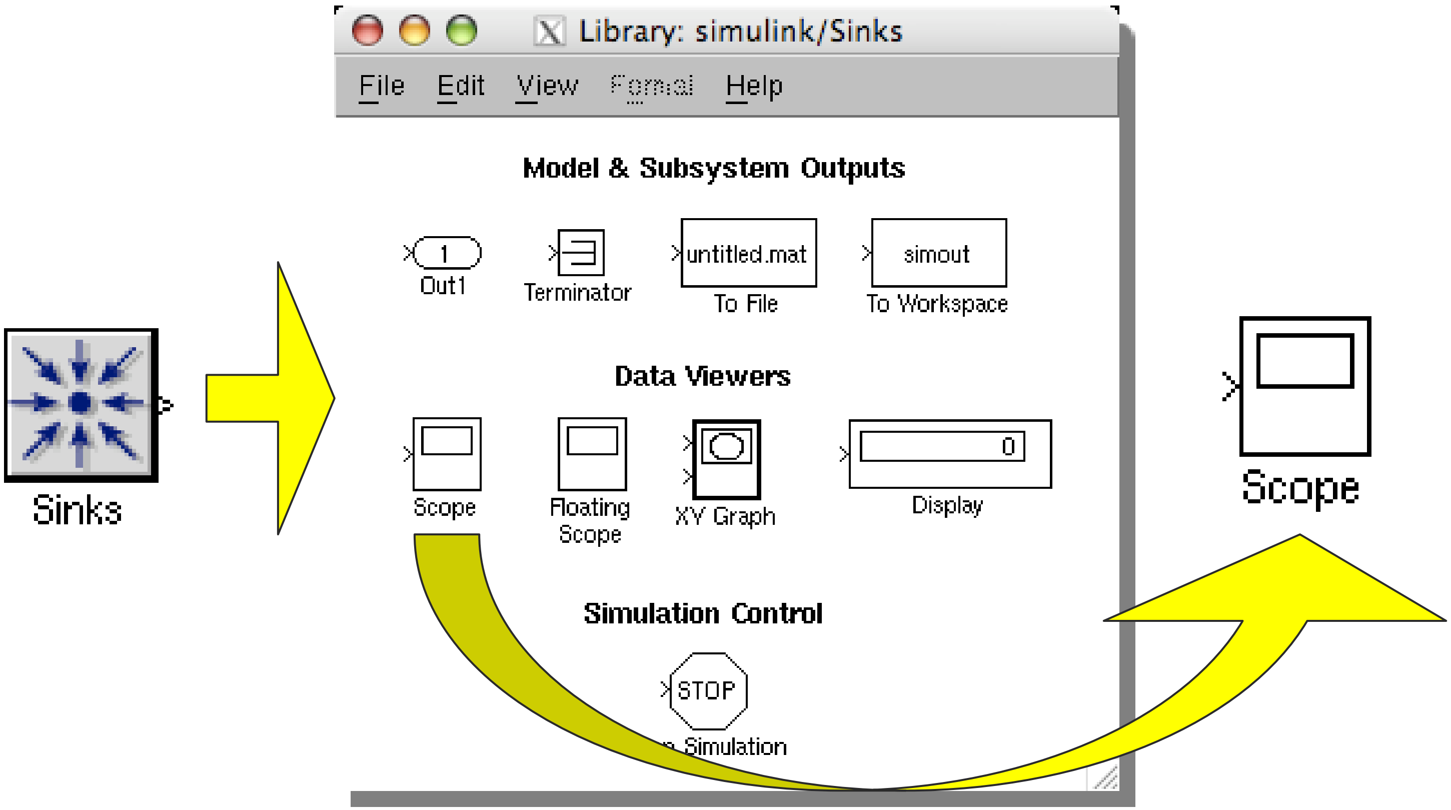
Model Blocks: Sources

- Example: Step Function



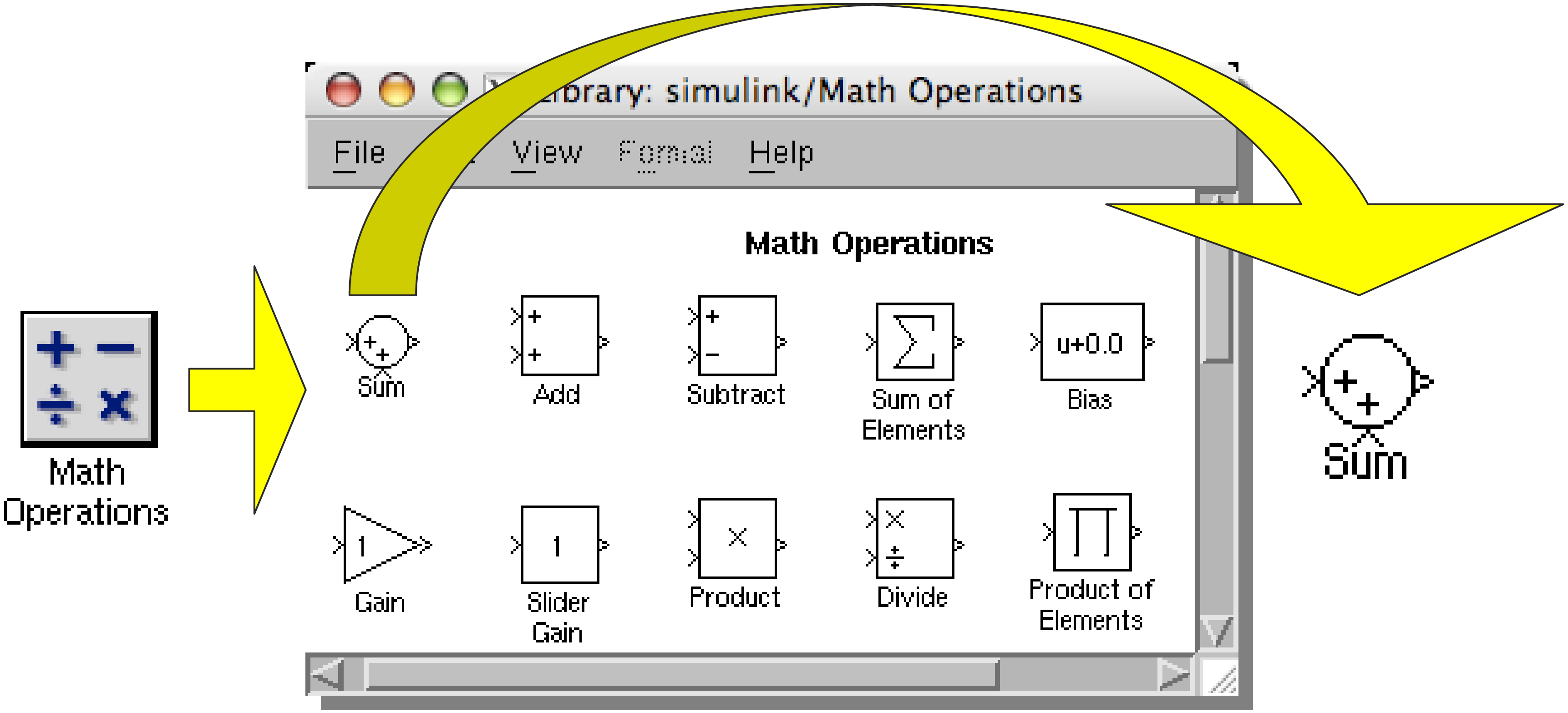
Model Blocks: Sinks

- Example: Scope



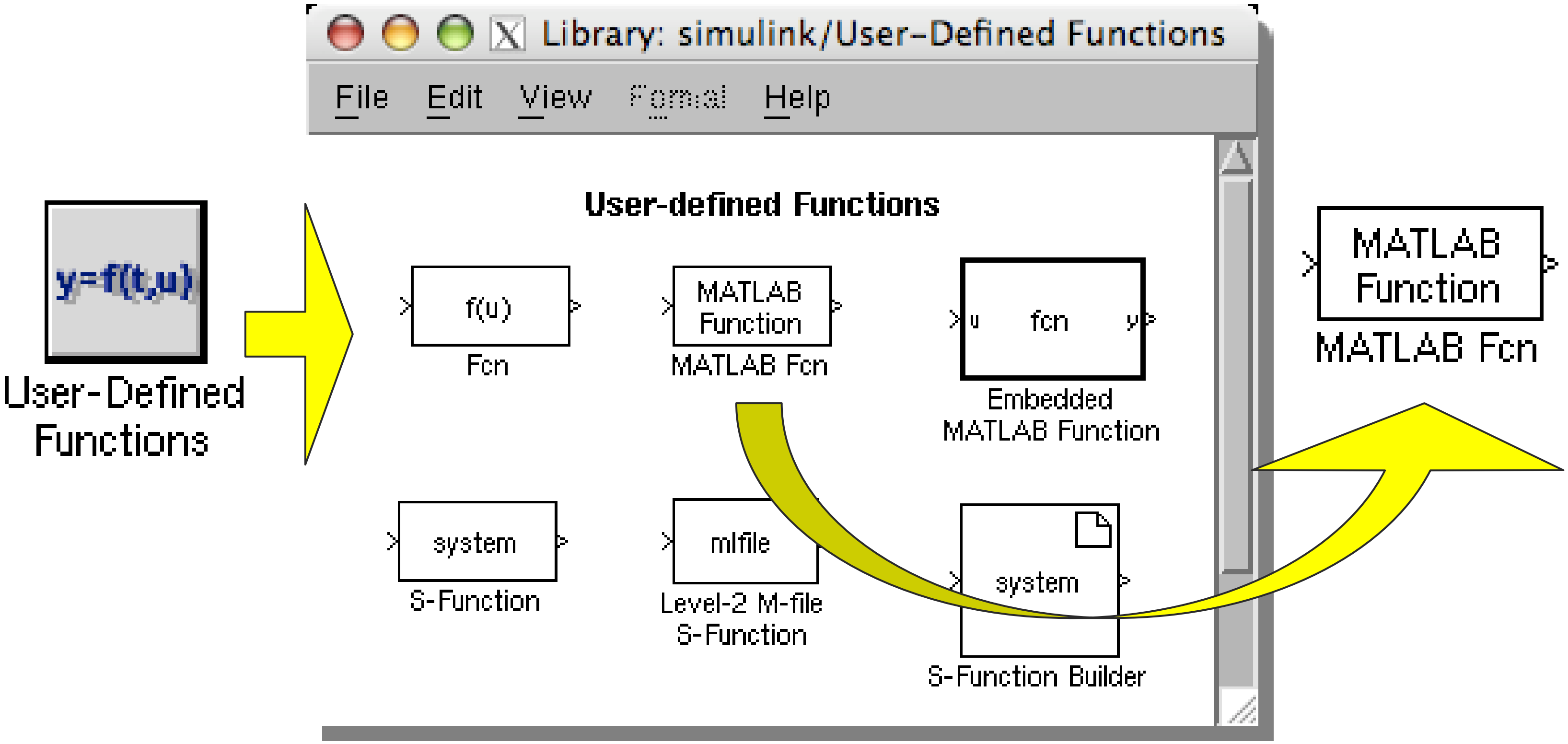
Model Blocks: Math Operations

- Example: Sum



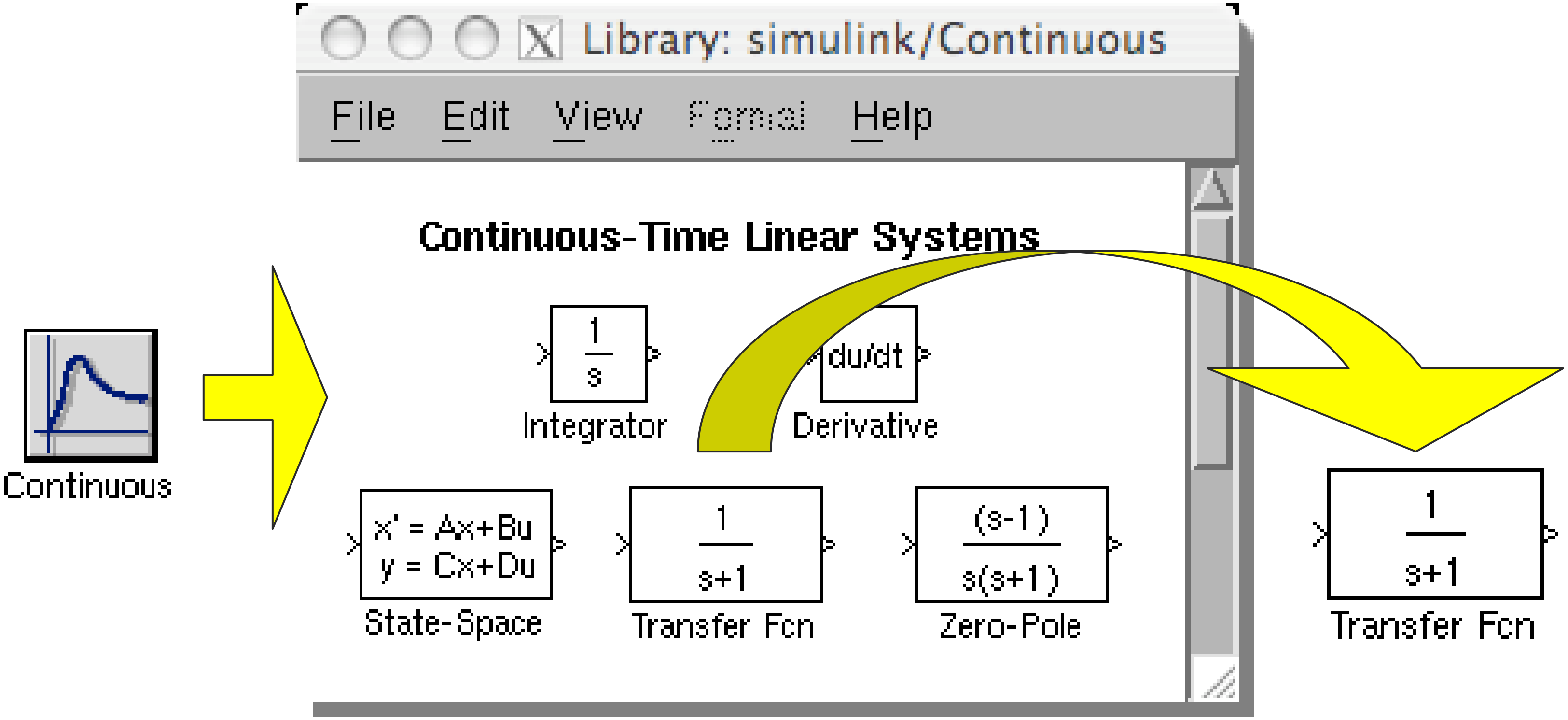
Model Blocks: User-Defined

- Example: MATLAB function



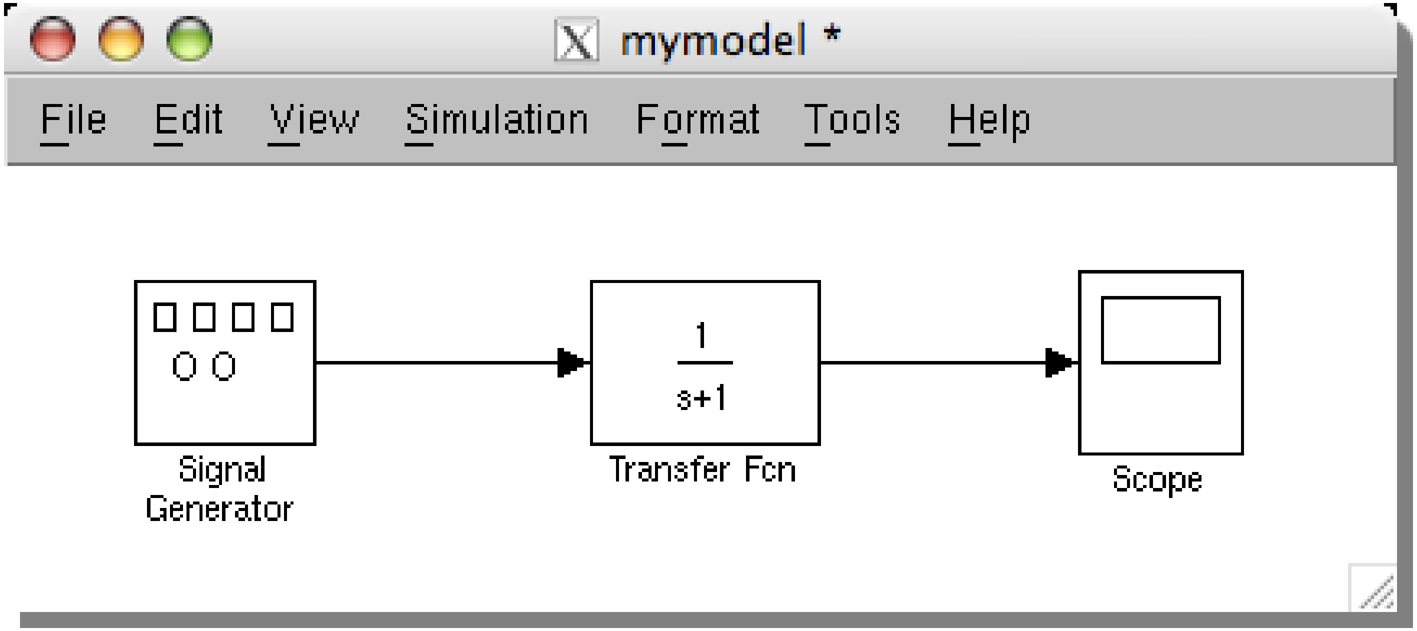
Model Blocks: Continuous State

- Example: Transfer Function

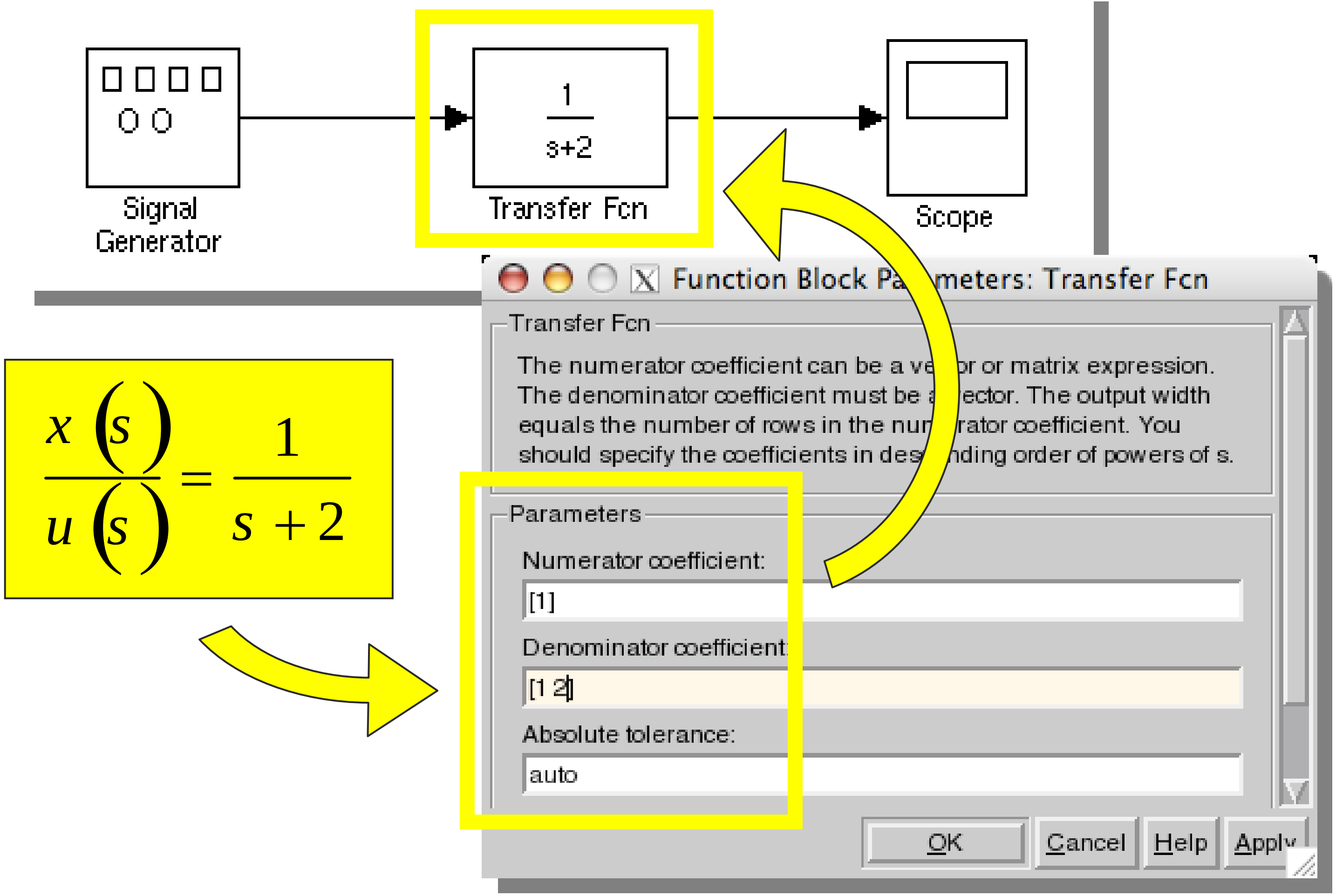


Modeling: Block Diagram

- Example: Continuous System



Modeling: Block Parameters



$$\frac{x(s)}{u(s)} = \frac{1}{s+2}$$

Function Block Pa meters: Transfer Fcn

Transfer Fcn

The numerator coefficient can be a ve or or matrix expression. The denominator coefficient must be a ector. The output width equals the number of rows in the nu rator coefficient. You should specify the coefficients in des nding order of powers of s.

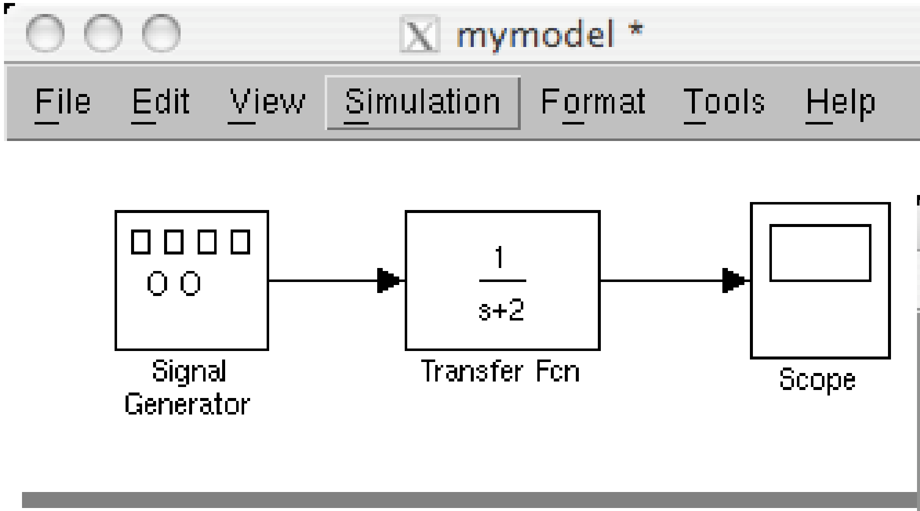
Parameters

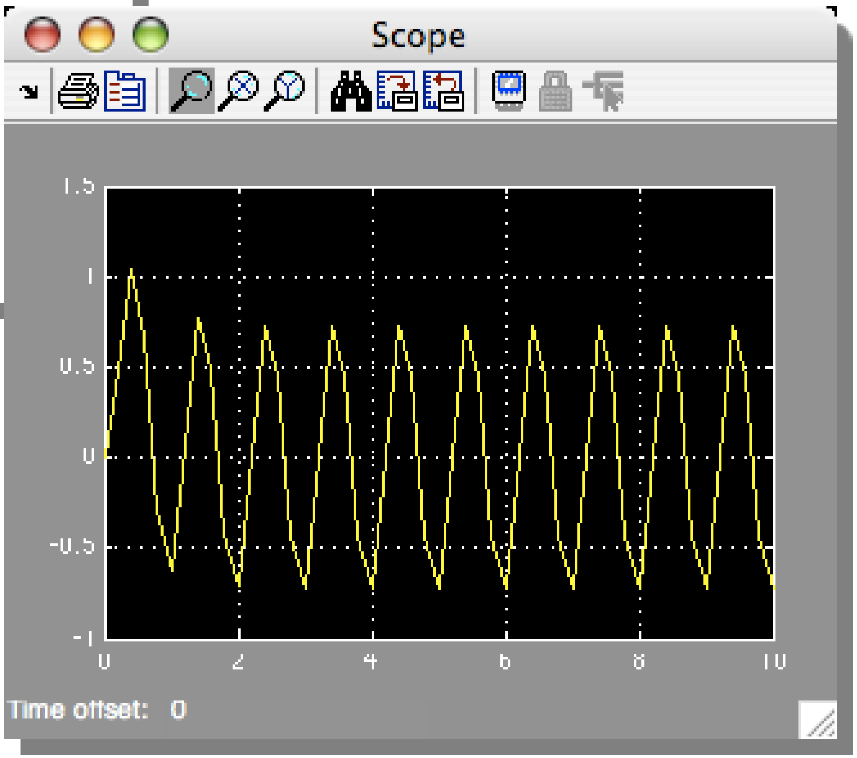
Numerator coefficient: [1]

Denominator coefficient: [1 2]

Absolute tolerance: auto

Running Simulations





- Configuration parameters
  `Simulation -> Set …`
- Run Simulation
  `Simulation -> Start`

Exercise 2: Simulink Model

- F-8 Controller Design
  - Simulink file: `f8.mdl`

Follow instructions in exercise handout …

Resources

- web.mit.edu/ist/topics/math
- web.mit.edu/acmath/matlab/course16
- 16.06 TA: Tom Gray
- 16.07 TA: Shannon Cheng

Questions?