

# An Introduction to Matlab and Simulink for SOFC Modeling and Controls

Stewart Moorehead

# Introduction

- ▶ The goal of this seminar is to introduce you to the basics of the Matlab and Simulink software packages and how they can be used for fuel cell system modeling and control.
- ▶ Seminar Outline
  - Matlab Introduction
  - Simulink Introduction
  - Fuel Cell Systems Modeling – SOFC based APU (example application)
  - Fuel Cell Systems Control

# Introduction to Matlab

- ▶ Matlab stands for MATrix LABoratory and is a programming language/environment designed for mathematics, particularly matrices.
- ▶ Matlab is an interpreted language.
- ▶ Has built in functions for matrices, complex numbers, graphing, polynomials and ODEs.
- ▶ Commands can be entered on the command line or through M-file scripts.

# Help Commands

- ▶ For help finding a command use type `help` at the command line :

`help` : brings up a list of help topics

`help topic` : lists all the commands in that topic

`help command` : brings up a description on how to use the command.

- ▶ If you cannot remember the command name use `lookfor` at the command line :

`lookfor keyword` : lists all commands that have *keyword* in the help description.

# Matrices

- ▶ Assign a matrix :

$$A = [1\ 2\ 3;4\ 5\ 6;7\ 8\ 9]$$

- ▶ Spaces separate columns, semi-colons (;) separate rows.

- ▶ A is therefore :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- ▶ Use the `whos` command to see what variables are defined.

# Accessing Matrices

- ▶ To access a matrix element specify the row and column :

$$x = A(2,3) \quad \longrightarrow \quad x = 6$$

- ▶ To access a submatrix or vector, use the colon

$$X = A(1:2, 2:3) \quad \longrightarrow \quad X = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

or

$$X = A(2,:) \quad \longrightarrow \quad X = [4 \ 5 \ 6]$$

# M-File Programming Example

```
A = [5 2 1;3 8 2;4 7 9];
```

```
B = A^2;
```

% ^ denotes exponent, so this is A squared

```
C = A^-1;
```

% And this is matrix inversion.

```
A(4,:) = [10 11 12];
```

%we can dynamically increase the size of  
%matrices

```
for i=1:100
```

```
    d(i) = 0;
```

```
    for j=1:4
```

```
        d(i) = (A(j,1) + i)^2 + d(i);
```

```
    end
```

```
end
```

# A few tricks to improve efficiency...

```
A = [5 2 1;3 8 2;4 7 9];
```

```
B = A^2;           % ^ denotes exponent, so this is A squared  
C = A^-1;         % And this is matrix inversion.
```

```
A(4,:) = [10 11 12];    %we can dynamically increase the size of matrices
```

```
%For greater efficiency allocate enough space for d all at once  
d = zeros(1,100);
```

```
for i=1:100  
    %We can further increase efficiency by eliminating one of the for  
    %loops and use the : operator instead  
    d(i) = sum((A(:,1) + i).^2);
```

```
end
```

# Polynomials and Functions

```
%An example to illustrate the use of functions and polynomials
%p is a vector of length N+1 whose elements are the coefficients of an Nth
%order polynomial.
%m is the order of a polynomial to fit to noise corrupted data made with p
%v is the mth order polynomial fit.
```

```
function [v] = example3(p, m)
```

```
d = zeros(20);
```

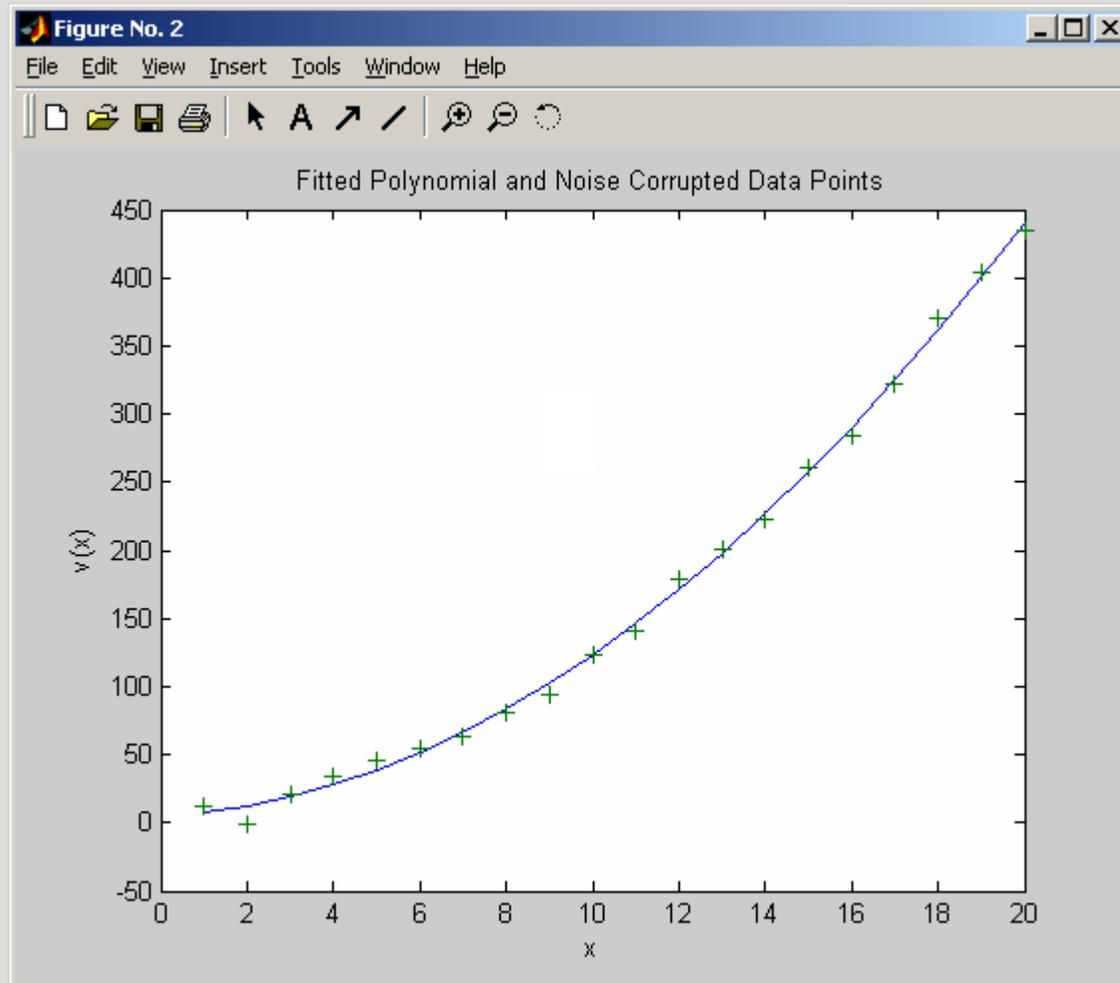
```
%create a vector of data points using p and a random number
for i=1:20
    d(i) = polyval(p, i) + (rand(1)-0.5)*20; %add random noise
end
```

```
%Graph the polynomial and the data points
figure(1);
plot(1:20, polyval(p, 1:20), 1:20, d(1:20), '+');
title('Original Polynomial and Noise Corrupted Data Points');
xlabel('x');
ylabel('p(x)');
```

```
%fit an mth order polynomial to the data in d
v = polyfit(1:20, d(1:20), m);
```

```
%Graph the fitted polynomial and data points
figure(2);
plot(1:20, polyval(v, 1:20), 1:20, d(1:20), '+');
title('Fitted Polynomial and Noise Corrupted Data Points');
xlabel('x');
ylabel('v(x)');
```

# Figure showing polynomial curve fit



# Functions

- ▶ M-File scripts read and place variables in the global workspace.
- ▶ Functions use local variables. Therefore they do not change your workspace variables, nor are they affected by the state of the current workspace.

# Toolboxes

- ▶ Matlab comes with many very useful built-in functions.
- ▶ However, for some specialty tasks these functions may not be sufficient.
- ▶ You can purchase specialty Toolboxes which are libraries of functions useful in specific fields.
- ▶ For example the Control Toolbox has commands to do Bode plots, Root Locus, Nyquist and controller design.
- ▶ Common toolboxes are: Control, Signal Processing and Statistics.

# The MathWorks Products and Prices • North America • August 2003

Please contact your sales representative for pricing on enterprise-based license options.

## MATLAB® 1

	Individual
MATLAB 1	\$1,900
MATLAB COM Builder 3, 28	\$3,000
MATLAB Compiler 2	\$2,700
MATLAB Report Generator	\$500
MATLAB Web Server 4	\$2,000
MATLAB Runtime Server Dev. Kit 4a, 21	\$5,000
MATLAB Runtime Distribution 4a, 21	See MATLAB Runtime Server price list.
MATLAB Excel Builder 3, 28	\$4,000
Excel Link 3	\$200

## MATLAB Toolboxes

### Math and Analysis

Optimization Toolbox	\$900
Statistics Toolbox	\$600
Neural Network Toolbox	\$900
Symbolic Math Toolbox 9	\$600
Extended Symbolic Math Toolbox 9, 12	\$900
Partial Differential Equation Toolbox	\$900
Mapping Toolbox	\$900
Spline Toolbox	\$400
Curve Fitting Toolbox 33	\$500

### Data Acquisition and Import

Data Acquisition Toolbox 3	\$900
Instrument Control Toolbox 3a	\$600
NEW Image Acquisition Toolbox 3	\$900
Database Toolbox 9	\$1,000

### Signal and Image Processing

Signal Processing Toolbox	\$800
Image Processing Toolbox 14	\$900
Communications Toolbox 6	\$800
System Identification Toolbox	\$900
Wavelet Toolbox 14, 15	\$900
Filter Design Toolbox 6	\$1,000
MATLAB Link for Code Composer Studio® 3, 6	\$1,000

### Control Design and Analysis

Control System Toolbox	\$1,000
Fuzzy Logic Toolbox	\$900
Robust Control Toolbox 16	\$900
μ-Analysis and Synthesis Toolbox 18	\$900
LMI Control Toolbox 18	\$900
Model Predictive Control Toolbox 18	\$900
Model-Based Calibration Toolbox 3, 13, 26, 29	\$7,000

## MATLAB Toolboxes

continued

### Finance and Economics

Financial Toolbox 13	\$900
NEW Fixed-Income Toolbox 10, 13	\$1,000
Financial Time Series Toolbox 10, 13	\$600
GARCH Toolbox 13	\$1,000
Datafeed Toolbox 3	\$1,000
Financial Derivatives Toolbox 10, 13	\$1,000

## Simulink®

Simulink 5	\$2,800
Simulink Performance Tools	\$500
Stateflow®	\$2,800
Stateflow Coder 8	\$2,800
Real-Time Workshop®	\$7,500
Real-Time Workshop Embedded Coder 7	\$5,000
Real-Time Windows Target 3, 7	\$2,000
xPC Target 3, 7	\$4,000
xPC Target Embedded Option 3, 7, 19	\$4,000
xPC TargetBox™ 19	See xPC TargetBox price list.
Simulink Report Generator 11	\$500
Requirements Management Interface	\$600
SimMechanics 30	\$4,000
SimPowerSystems	\$2,000
Virtual Reality Toolbox 27, 30	\$1,000

## Embedded Targets 3, 7, 26

Embedded Target for Infineon C166® Microcontrollers 32	\$4,000
NEW Embedded Target for Motorola® HC12 32	\$4,000
Embedded Target for Motorola® MPC555 31, 32	\$4,000
NEW Embedded Target for OSEK/VDX® 32	\$4,000
Embedded Target for TI® C6000™ DSP 6, 20, 23	\$4,000

## Simulink Blocksets 26

DSP Blockset 6	\$1,000
Fixed-Point Blockset	\$2,000
Dials & Gauges Blockset 3	\$700
Communications Blockset 6, 20, 22	\$1,000
CDMA Reference Blockset 6, 20, 22, 24	\$2,000
Nonlinear Control Design Blockset	\$1,000
Aerospace Blockset 16	\$1,000

## Footnotes

- 1: Prerequisite for all other products
- 2: Includes MATLAB Compiler, MATLAB C/C++ Math Library, and MATLAB C/C++ Graphics Library. Not all products are eligible for distribution; see the price list addendum.
- 3: Available only for Windows
- 3a: Available only for Windows, Solaris, or Linux
- 4: Available only for Windows NT, Solaris, or Linux
- 4a: Consult the price list addendum for products eligible for distribution. Contact The MathWorks for quantity prices and toolbox distribution pricing.
- 5: Prerequisite for all Simulink products
- 6: Requires Signal Processing Toolbox
- 7: Requires Real-Time Workshop
- 8: Requires Stateflow
- 9: Not available for HP 700/HP-UX10
- 10: Requires Financial Toolbox
- 11: Requires MATLAB Report Generator
- 12: Includes Symbolic Math Toolbox functionality
- 13: Requires Statistics and Optimization Toolboxes
- 14: Signal Processing Toolbox recommended
- 15: Image Processing Toolbox recommended
- 16: Requires Control System Toolbox
- 18: Control System Toolbox recommended
- 19: Requires xPC Target
- 20: Requires DSP Blockset
- 21: Not available for Macintosh OS X
- 22: Requires Communications Toolbox
- 23: Requires MATLAB Link for Code Composer Studio. Fixed-Point Blockset recommended.
- 24: Requires Communications Blockset
- 26: Requires Simulink
- 27: Requires Simulink for blockset portion of product
- 28: Requires MATLAB Compiler
- 29: Requires Extended Symbolic Math Toolbox
- 30: Not available for HP 700, IBM RS, and HP-UX
- 31: Requires Stateflow and Stateflow Coder (only for CCP-related blocks)
- 32: Requires Real-Time Workshop Embedded Coder
- 33: Command line only for HP700/HP-UX10, HP-UX11, and IBM RS

Products are available on Windows, UNIX, Linux, and Macintosh OS X unless otherwise indicated.

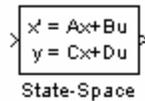
Prices are per unit, listed in U.S. dollars, valid for program installation and use in the U.S. or Canada only, and are subject to change without notice.

# Simulink

- ▶ Simulink is a software package for modeling, simulating and analyzing dynamic systems.
- ▶ Simulink can model linear and non-linear, continuous and discrete time systems.
- ▶ Simulink has an easy to use GUI which lets you build models as block diagrams.
- ▶ Simulink is part of Matlab and can access all of Matlab's functions.

# Some Simulink Blocks

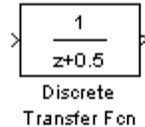
## Continuous



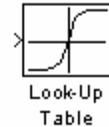
## Non-Linear



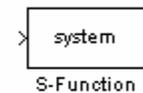
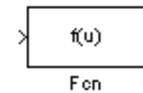
## Discrete



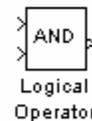
## Lookup Tables



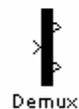
## Functions



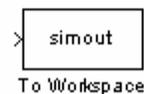
## Math



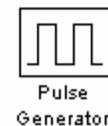
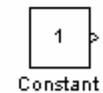
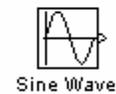
## Signals



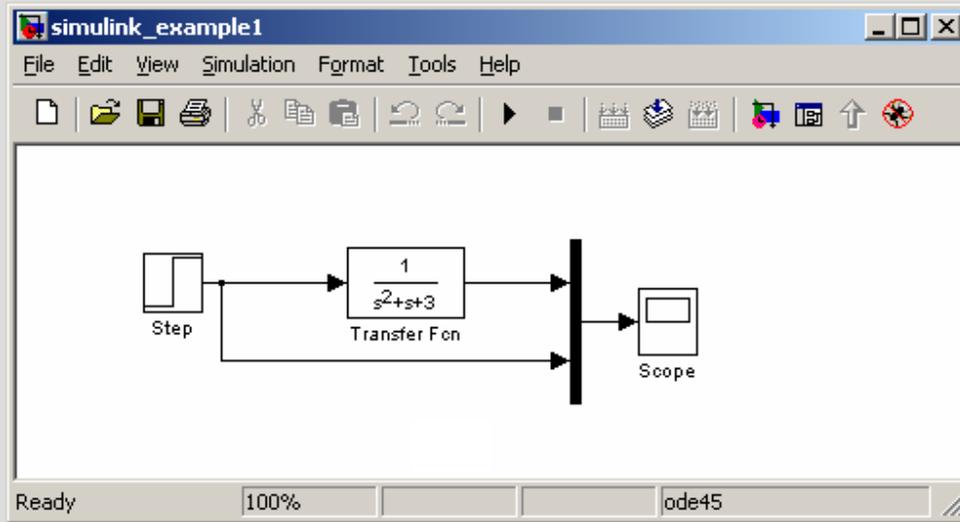
## Outputs



## Inputs

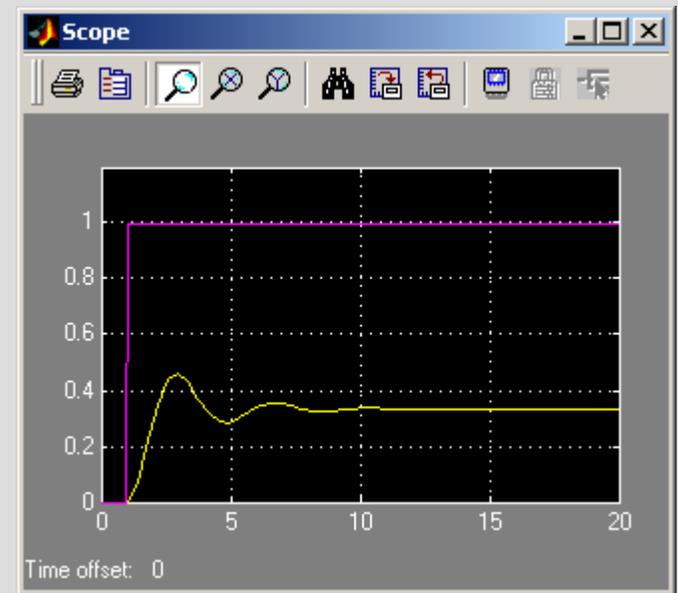


# Example 1 : 2<sup>nd</sup> Order System

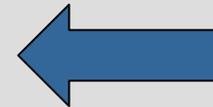
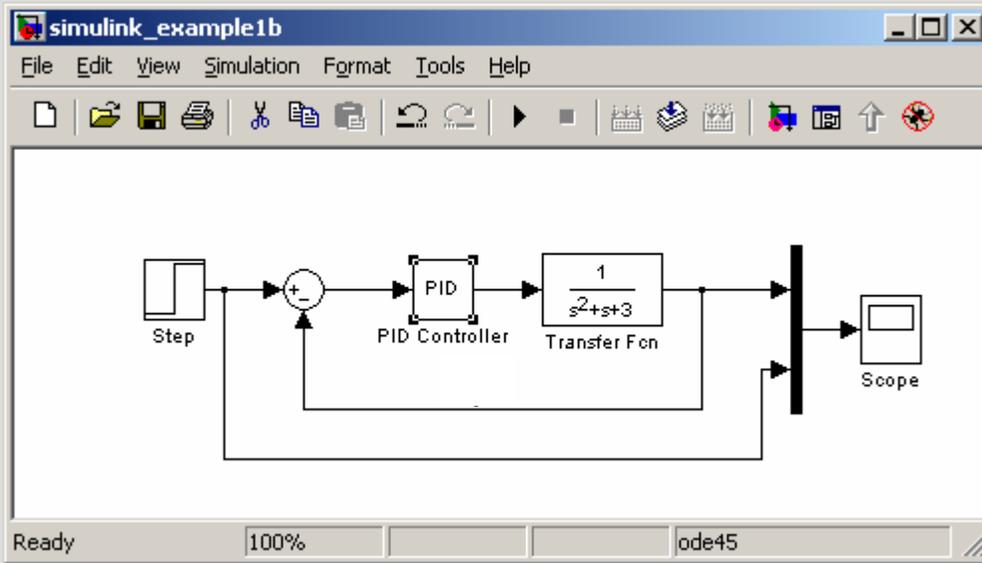


← Model

Input Signal (Purple)  
Model Output (Yellow)

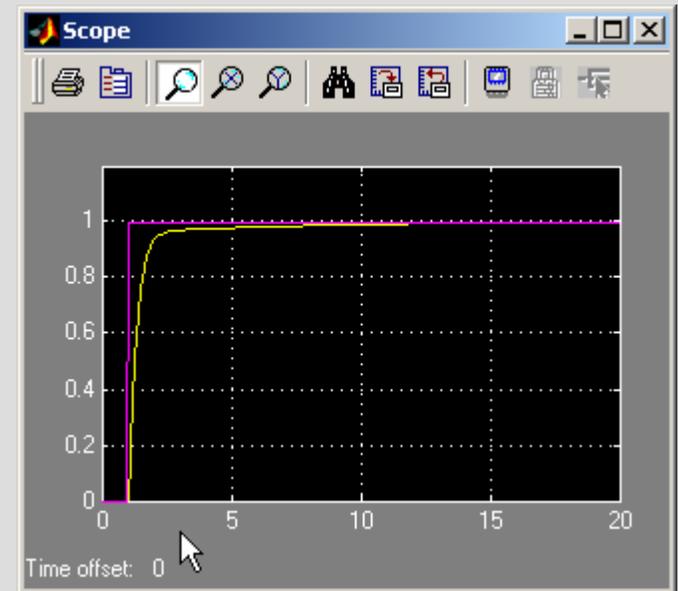


# Add PID Control to Model

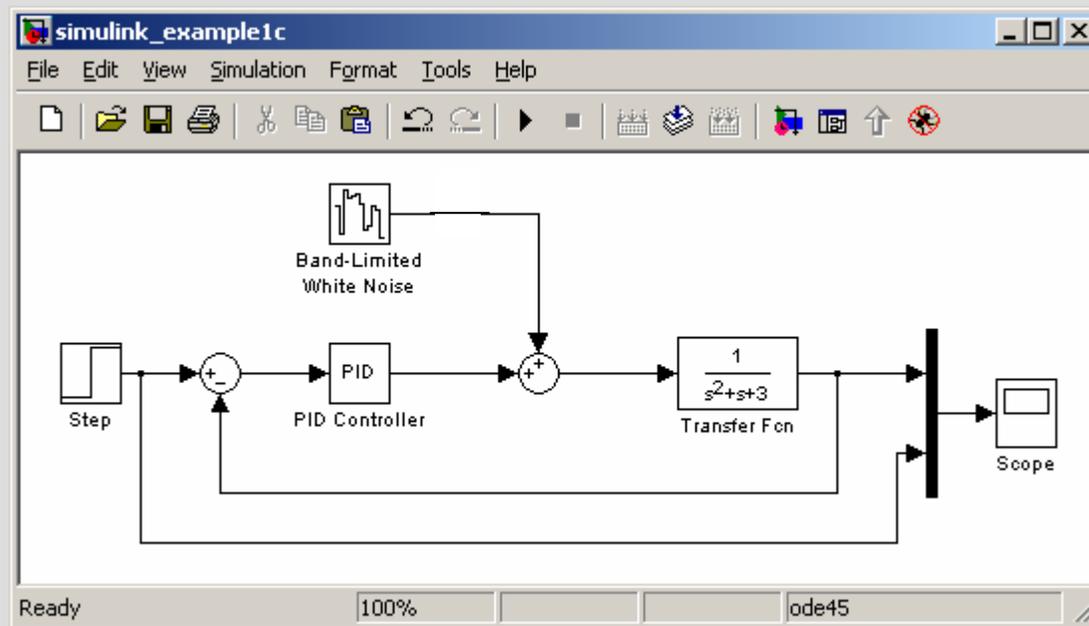


Model

Input Signal (Purple)  
Model Output (Yellow)

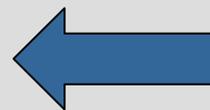
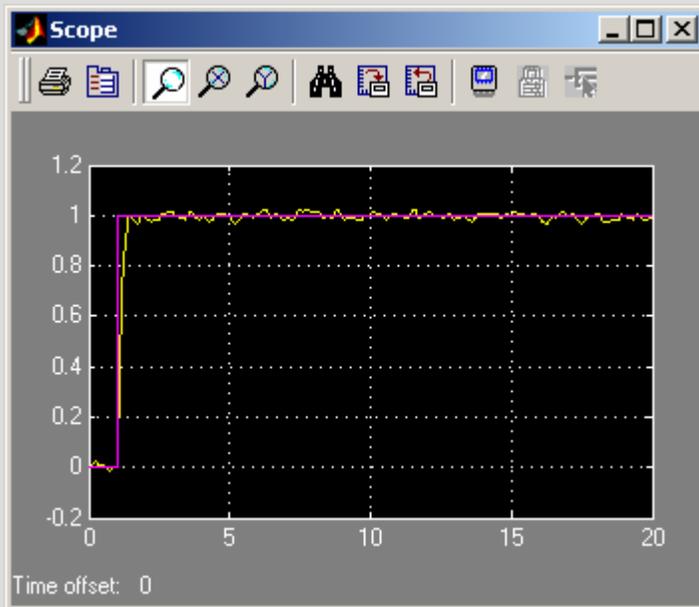
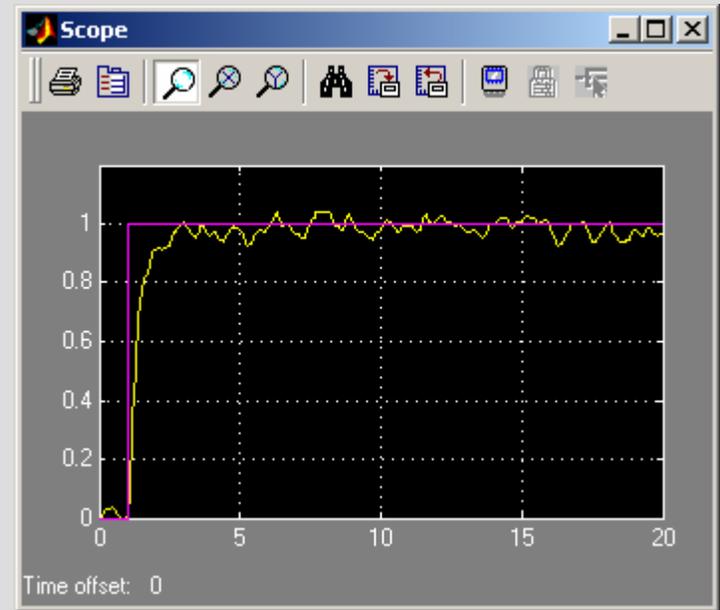


# Add noise ...



# Reduce Noise Effects with Controller

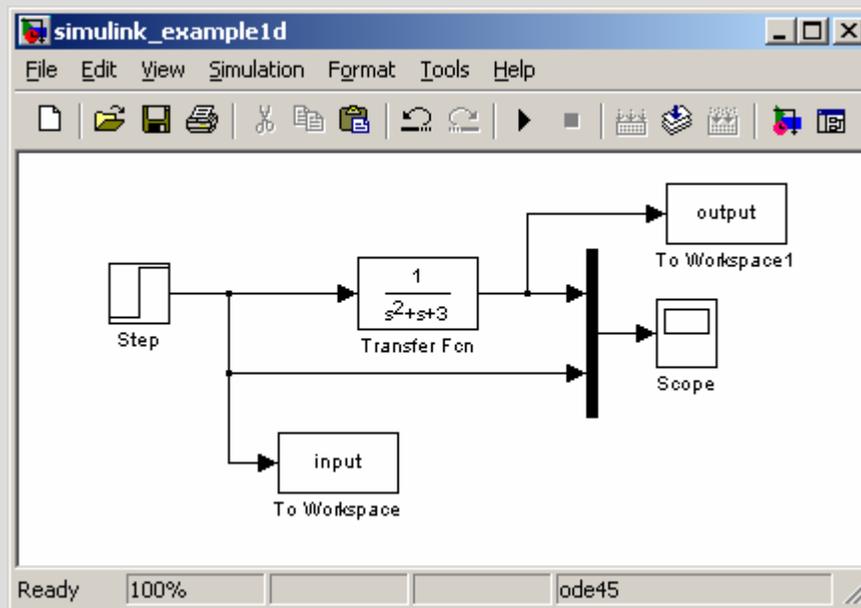
With original gains of  
 $P=30$ ,  $I=5$ ,  $D=10$



And increased gains  
 $P=100$ ,  $I=15$ ,  $D=15$

# Saving Data to Matlab Workspace

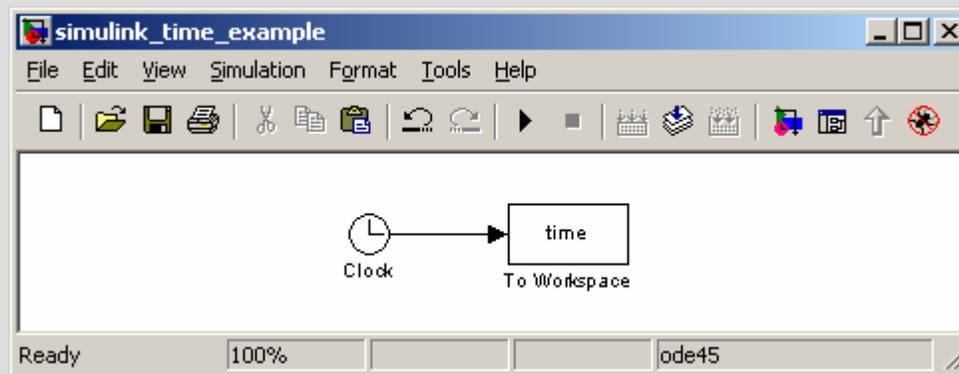
- ▶ Use the “To Workspace” block to send simulation data to a variable in the Matlab workspace.



- ▶ Saves simulation data to variables called input and output.

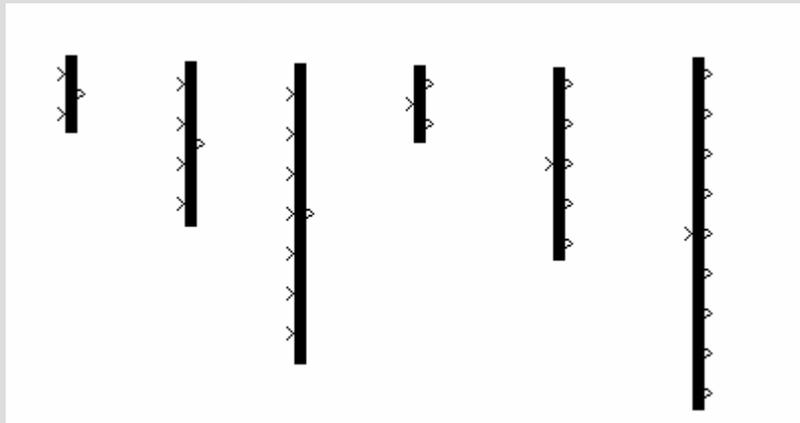
# Saving Time

- ▶ Simulink uses a variable time step in the simulation which makes it difficult to determine the time, in seconds, of a data point.
- ▶ Use the “Clock” block, to save the time at each time step.



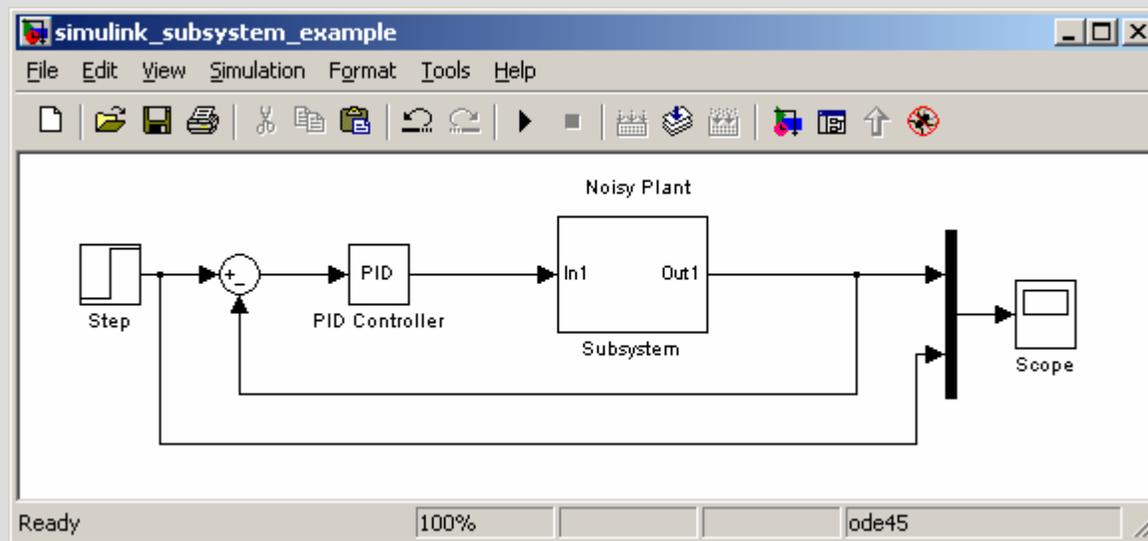
# Managing Signals – Muxs and Demuxs

- ▶ On complex models the number of wires linking blocks can grow very large, impairing readability.
- ▶ Use the “mux” block to combine any number of wires into a single bus.
- ▶ Many Simulink blocks can accept buses as inputs, treating them as a vector.
- ▶ Use the “demux” block to break a bus down into single wires again.



# Subsystems

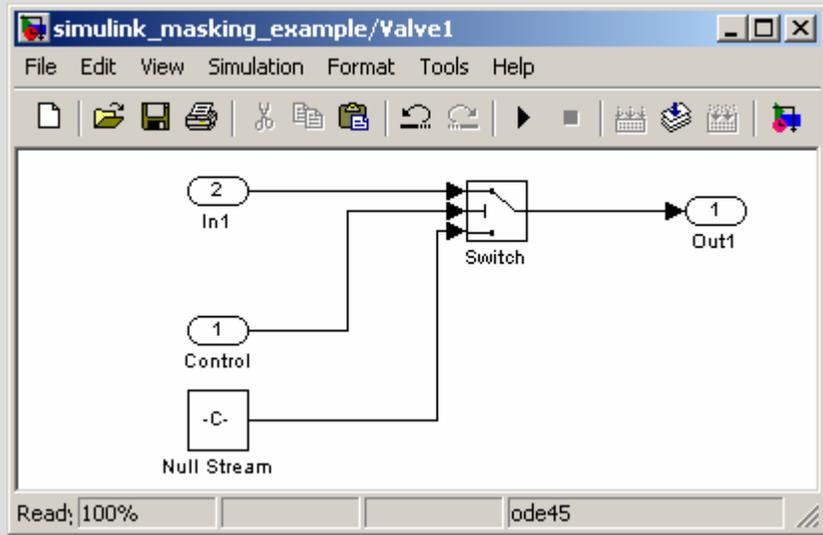
- ▶ Collections of blocks can be combined into a “Subsystem” block.
- ▶ Here the noise source and plant blocks from the previous example are placed into a subsystem called “Noisy Plant”



# Masking

- ▶ Normally when a subsystem is double clicked a new window appears showing the blocks inside.
- ▶ By Masking the subsystem, we can make double clicking pull up a new window with instructions and user input parameters.
- ▶ Masking also lets us specify how the block icon should be drawn.

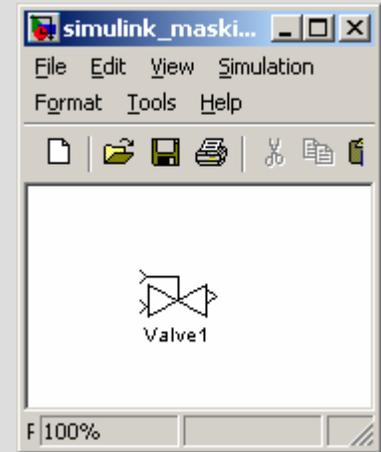
# Masking Example



Valve blocks are combined into a subsystem.



Masking allows the icon to be altered.



Double clicking on the Valve1 icon will bring up this information block rather than the inner components.



The dialog box is titled 'Block Parameters: Valve1'. It contains a text area with the following text: 'Valve (mask)' followed by two lines: 'If control input is 1, input stream is passed unaltered to output. (Valve open)' and 'If control input is 0, output is a zero stream. (Valve closed)'. At the bottom, there are four buttons: 'OK', 'Cancel', 'Help', and 'Apply'.

# Masking Example 2

Masks can also request user input which is used by the model components inside the masked block.



**Block Parameters: SOFC** [x]

Subsystem (mask)

The SOFC block models a Solid Oxide Fuel Cell which produces electricity when Hydrogen or Carbon Monoxide and Oxygen gases are passed through it. A single SOFC is referred to as a cell, much like a battery cell. To get a desired output voltage multiple cells are combined to create a stack. The parameters allow you to set the number of SOFC cells in the stack and the temperature of the stack at the start of simulation.

Parameters

Initial Stack Temperature (C)

Number of cells in stack

OK Cancel Help Apply

# M Functions

- ▶ The “MATLAB Fcn” block lets you include a Matlab M-File function to the Simulink model.
- ▶ The M-File must be a function with 1 input vector and 1 output vector. For example :

```
function v=matlab_func(u)
```

- ▶ If you have more than one input, use the “mux” blocks to create a single bus.
- ▶ “MATLAB Fcn” blocks can be convenient for complex equations but are slow to execute.

# S Functions

- ▶ Similar to M-functions, S-Functions provide a method of programming dynamic equations easily.
- ▶ An S-Function is similar to an M-file with several functions contained in the same file.
- ▶ During simulation, Simulink calls particular functions of the S-Function at the appropriate time. For example the `mdlInitializeSizes` function is called at the start.
- ▶ The S-Function can save states between calls. Just like State Space models, the states are used to compute the outputs and the derivatives of the states are computed to update the states at each time step.

# Example S-Function File

```
function
    [sys,x0,str,ts]=limintm(t,x,u,flag,lb,ub,xi)
%LIMINTM Limited integrator implementation.
% Example M-file S-function implementing a
% continuous limited integrator
% where the output is bounded by lower bound
% (LB) and upper bound (UB)
% with initial conditions (XI).
%
% Copyright 1990-2002 The MathWorks, Inc.
% $Revision: 1.17 $

switch flag
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 0
        [sys,x0,str,ts] =
            mdlInitializeSizes(lb,ub,xi);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Derivatives %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 1
        sys = mdlDerivatives(t,x,u,lb,ub);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Update and Terminate %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    case {2,9}
        sys = []; % do nothing

    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Output %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 3
        sys = mdlOutputs(t,x,u);
    otherwise
        error(['unhandled flag = ',num2str(flag)]);
end
% end limintm

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlInitializeSizes
% Return the sizes, initial conditions, and
% sample times for the S-function.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sys,x0,str,ts] =
    mdlInitializeSizes(lb,ub,xi)

    sizes = simsizes;
    sizes.NumContStates = 1;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 1;
    sizes.NumInputs = 1;
    sizes.DirFeedthrough = 0;
    sizes.NumSampleTimes = 1;

    sys = simsizes(sizes);
    str = [];
    x0 = xi;
    ts = [0 0]; % sample time: [period, offset]

% end mdlInitializeSizes
```

# Example S-Function Continued

```
%=====
% mdlDerivatives
% Compute derivatives for continuous states.
%=====
function sys = mdlDerivatives(t,x,u,lb,ub)

if (x <= lb & u < 0) | (x>= ub & u>0 )
    sys = 0;
else
    sys = u;
end

% end mdlDerivatives

%=====
% mdlOutputs
% Return the output vector for the S-function
%=====
function sys = mdlOutputs(t,x,u)

sys = x;

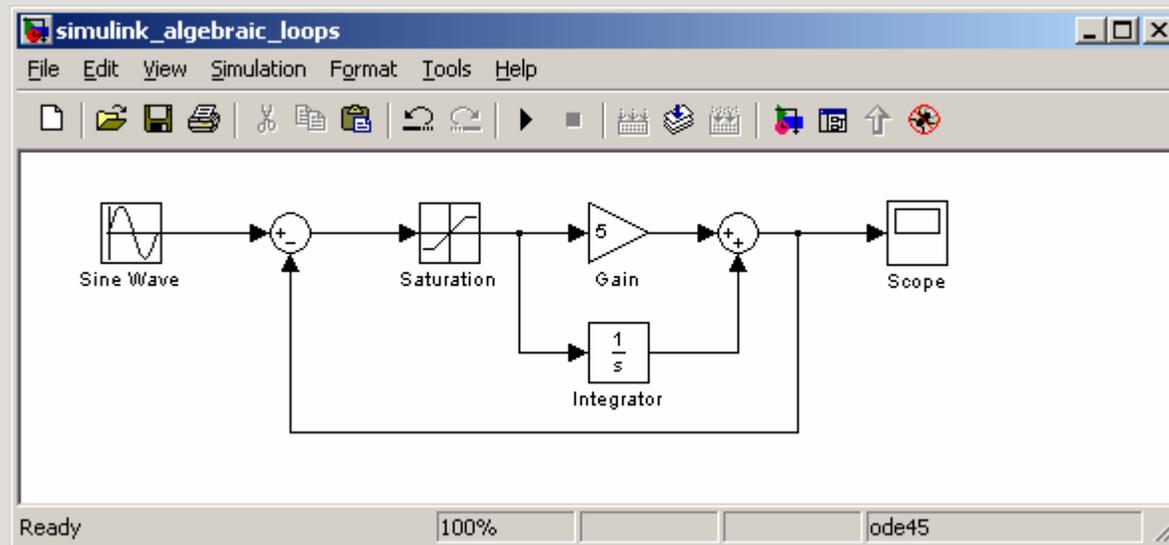
% end mdlOutputs
```

- ▶ mdlDerivatives computes the derivative of the state,  $x$ . Simulink does the numerical integration to update state value.
- ▶ mdlOutputs computes the output variables, in this case it is simply the state,  $x$ .

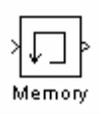
# Algebraic Loops

- ▶ Problems can occur in Simulink when the input of a block depends on the output of that block at the same time.
- ▶ This is called an Algebraic Loop which means that the model contains a loop with no dynamic or delay components in it.
- ▶ Simulink may have problems solving these algebraic loops, especially if they are nonlinear.
- ▶ Algebraic loops are solved using Newton-Raphson which can dramatically slow down simulation time.

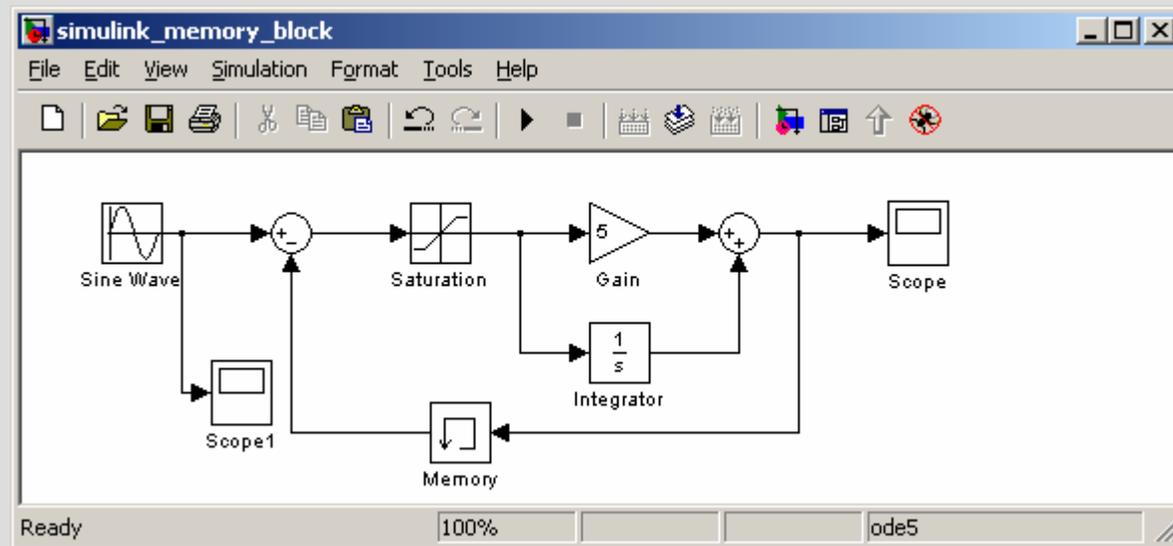
# Algebraic Loop Example



# Elimination of Algebraic Loops

- ▶ Algebraic loops can be eliminated in several ways :
  - Reformulate equations to remove the loop.
  - Add some dynamics to the loop.
  - Use the memory block. 
- ▶ The memory block delays the signal by one time step. This effectively removes the algebraic loop.
- ▶ However, the memory block may adversely affect the accuracy of the results.

# Example of Memory Block Usage



# Advisor

- ▶ NREL has developed an Advanced Vehicle Simulator in Simulink called Advisor.
- ▶ You can download Advisor at :  
<http://www.ctts.nrel.gov/analysis/>
- ▶ Advisor has many model blocks for common vehicle parts, including electric and hybrid vehicles.

# (Example Application) SOFC Auxiliary Power Units (APU)

- ▶ Project goal is to develop system models and control techniques for an SOFC based APU for long haul trucks.
- ▶ Purpose of project is to improve SOFC APU efficiency and durability through better control techniques.
- ▶ Using Simulink to implement the models and control.

# APU Model

- ▶ System model consists of controller, electrical system and APU.
- ▶ SOFC model is based on Larry Chick's model from SECA. We extended the thermal aspects to deal with heat up phase and are adding dynamic components to the fuel utilization.
- ▶ Reformer model uses a diesel approximate as fuel.
- ▶ Electrical system modeling models the power conversion electronics as well as the electrical loads such as air conditioning.

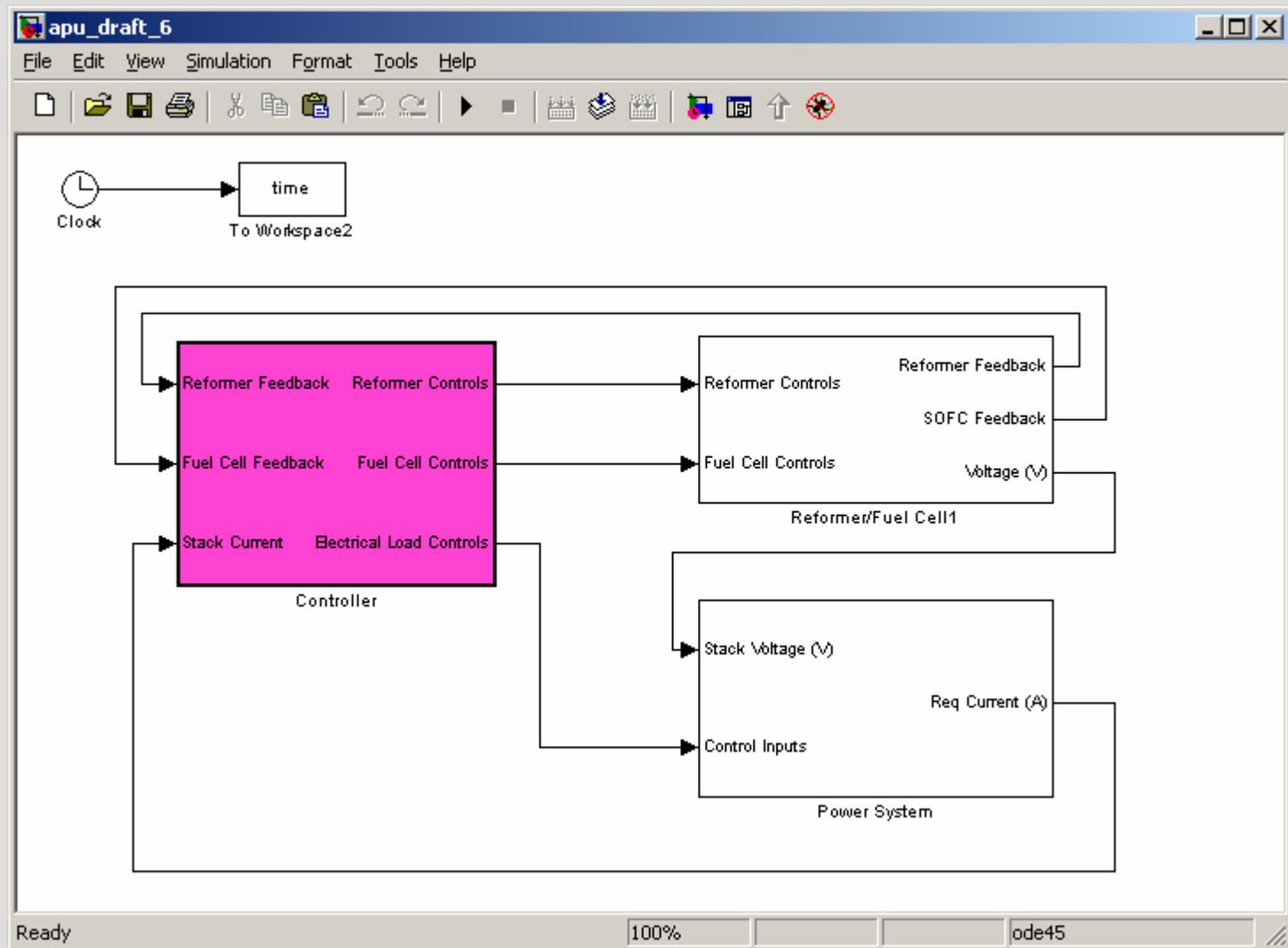
# Model Components – Fluid Stream

- ▶ A central part of the APU model is the fluid stream.
- ▶ The fluid stream is a bus containing the required information about a fluid stream such as the anode input to the SOFC.
- ▶ The fluid stream bus has 9 elements :

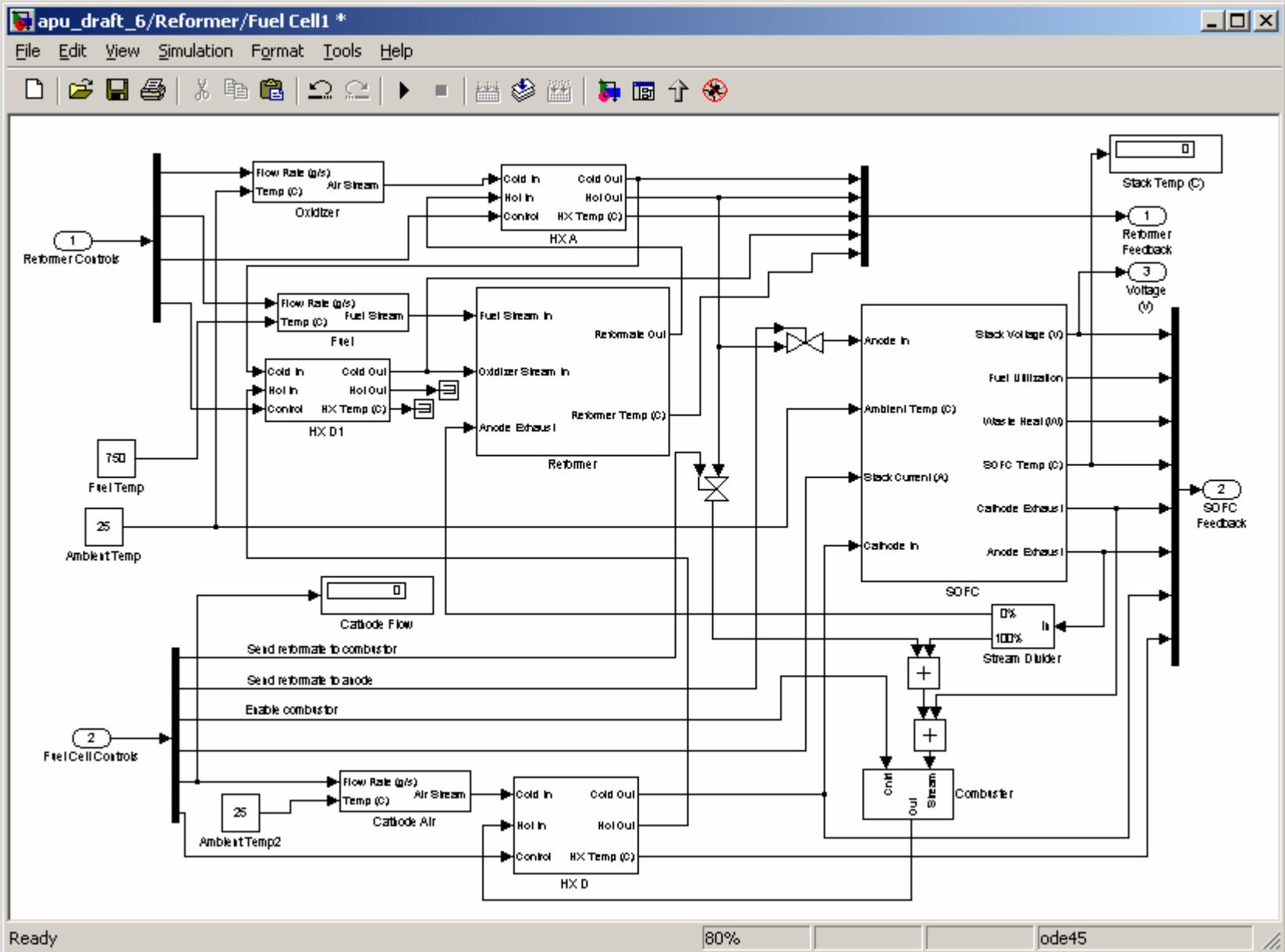
1. Flow Rate (g/s)
2. Temperature (deg C)
- 3-9. Molar Fraction Composition  
(elements 3-9 sum to one)

- Molar Fraction Composition
3. N<sub>2</sub>
  4. H<sub>2</sub>
  5. H<sub>2</sub>O
  6. CO
  7. CO<sub>2</sub>
  8. Fuel (Diesel, gasoline or CH<sub>4</sub>)
  9. O<sub>2</sub>

# System Model



# APU Model



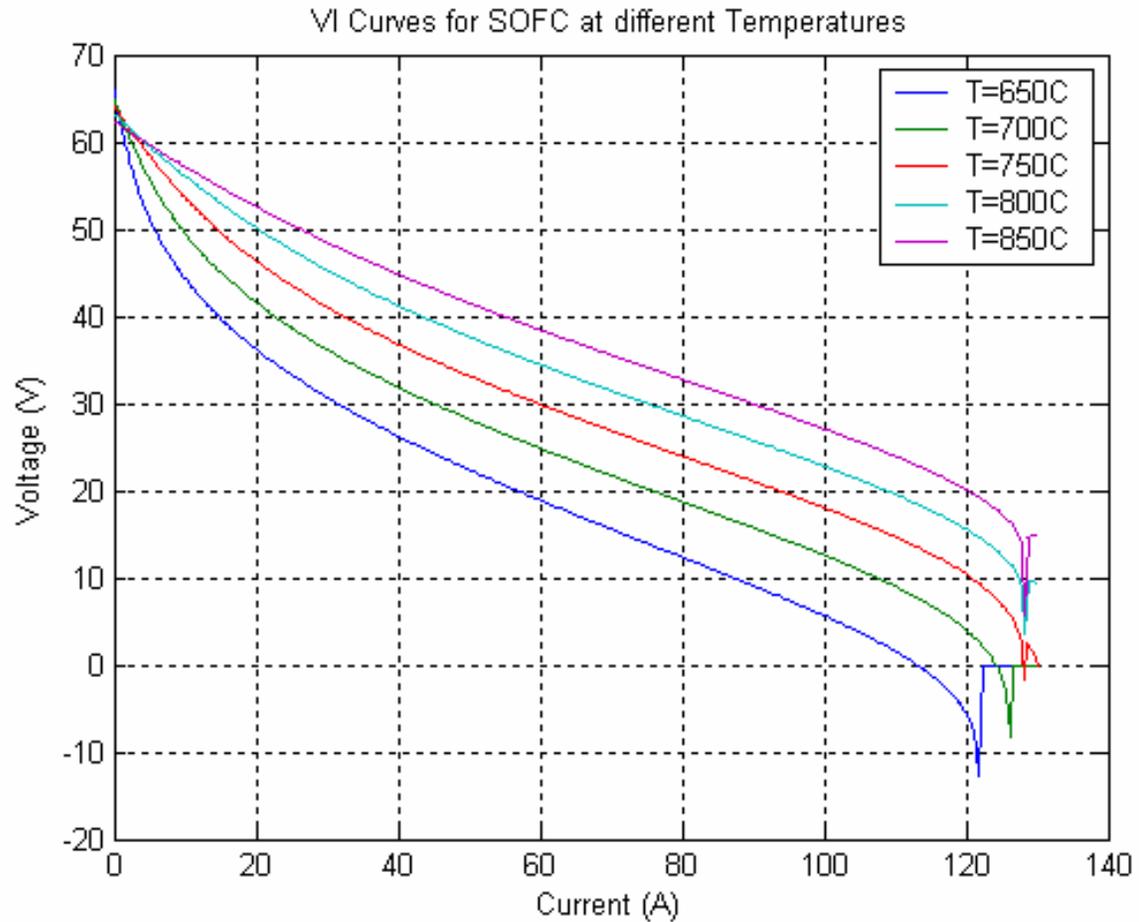
# Reformer

- ▶ Use single hydrocarbon approximate ( $C_{12.95}H_{24.38}$ ).
- ▶ Lookup tables used to compute composition of reformat based on temperature and O:C ratio.
- ▶ Output temperature computed using enthalpies and heats of formation.
- ▶ The reformation process considered is a Partial Oxidation – this negates the need for water storage in the APU.
- ▶ Anode gas re-circulation can provide some steam, greatly improving the reformation process.

# SOFC

- ▶ SOFC model is based on Larry Chick's model which was presented this morning.
- ▶ Implemented in Simulink as an S-Function.
- ▶ Stack block was masked, allowing the user to easily change number of cells in stack and initial temperature. Other parameters can be changed by editing the S-function.
- ▶ Model maintains stack temperature as a state and computes the heat transfers from stack to exhaust gases.

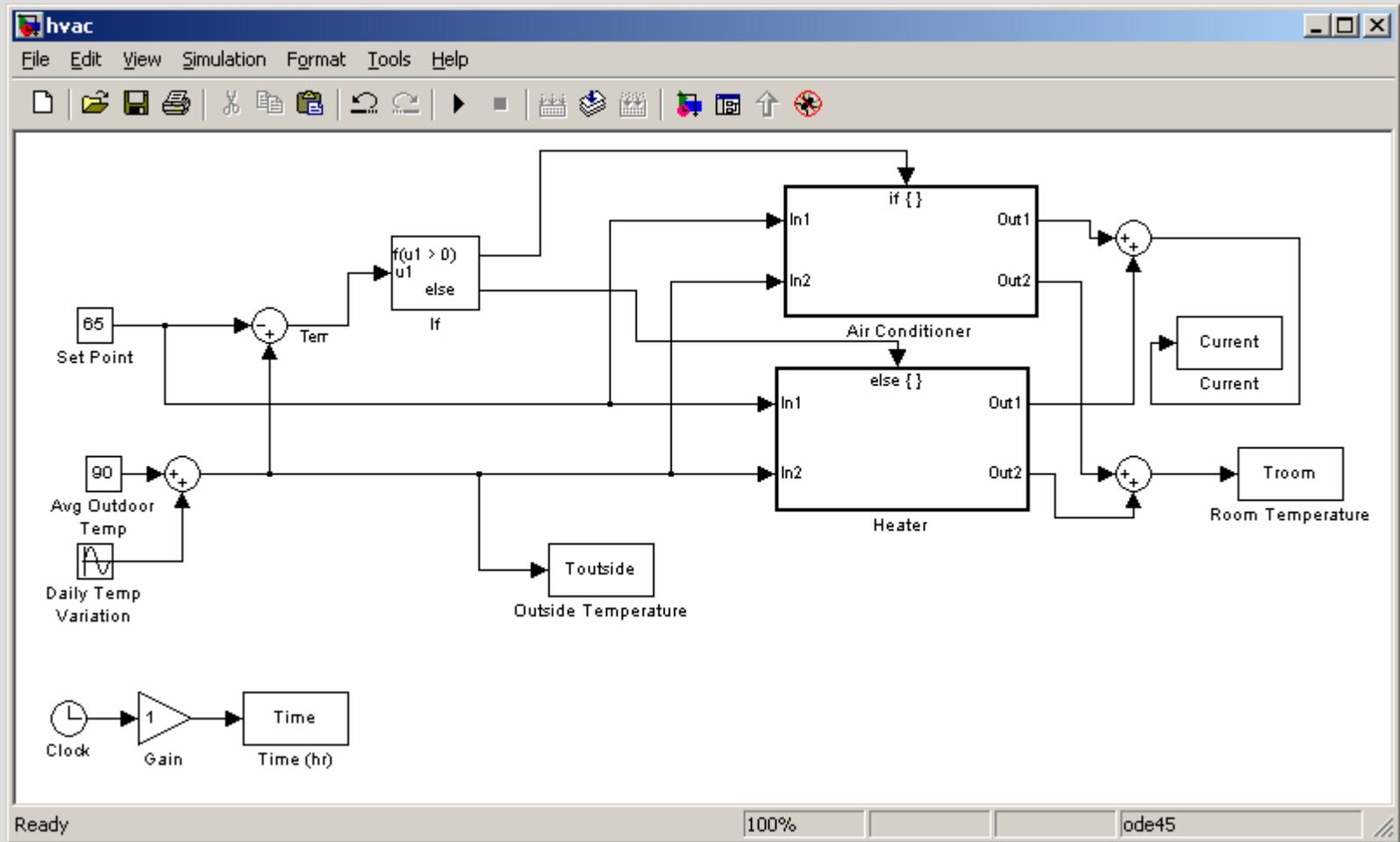
# VI Curves



# Electrical Load Modeling

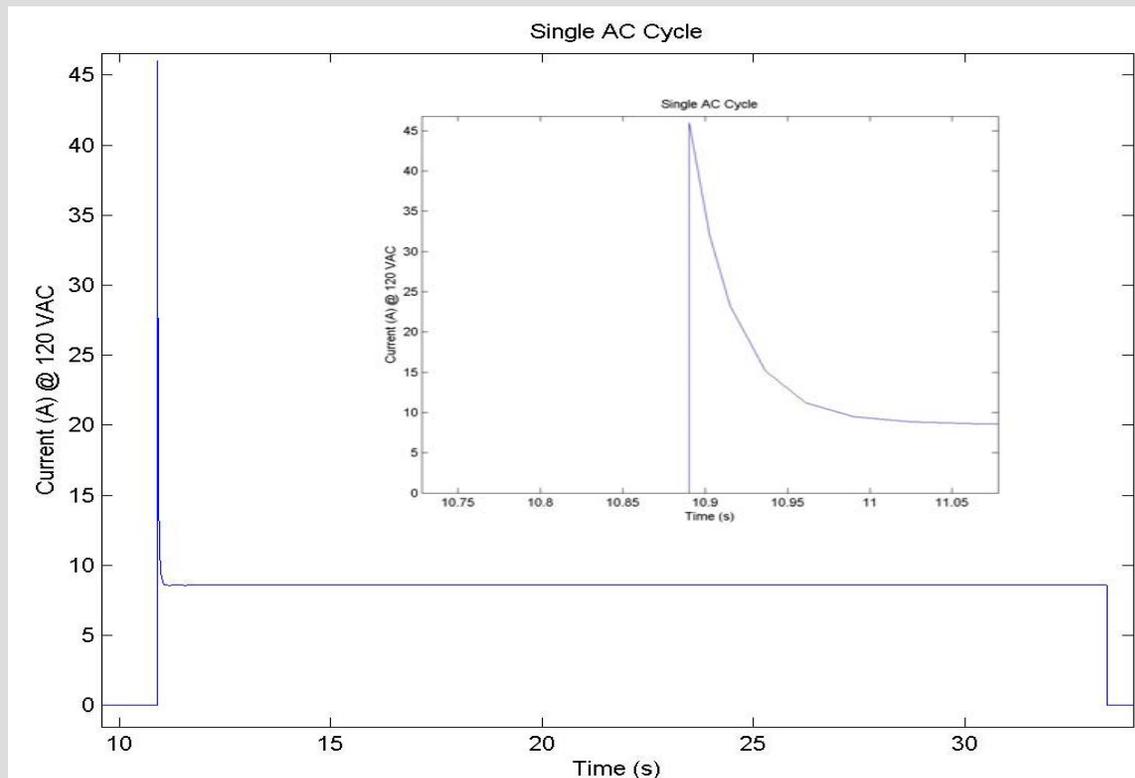
- ▶ An important part of the APU system is the electrical load and power conversion electronics.
- ▶ Working with University of Illinois, Chicago to develop very efficient power converters and models in Simulink.
- ▶ A major electrical load for an APU is the heating and air conditioning of the cab/living quarters.

# Heating/Air Conditioning



# Air Conditioner Current Requirements

- ▶ Model shows the inrush current as compressor turns on.



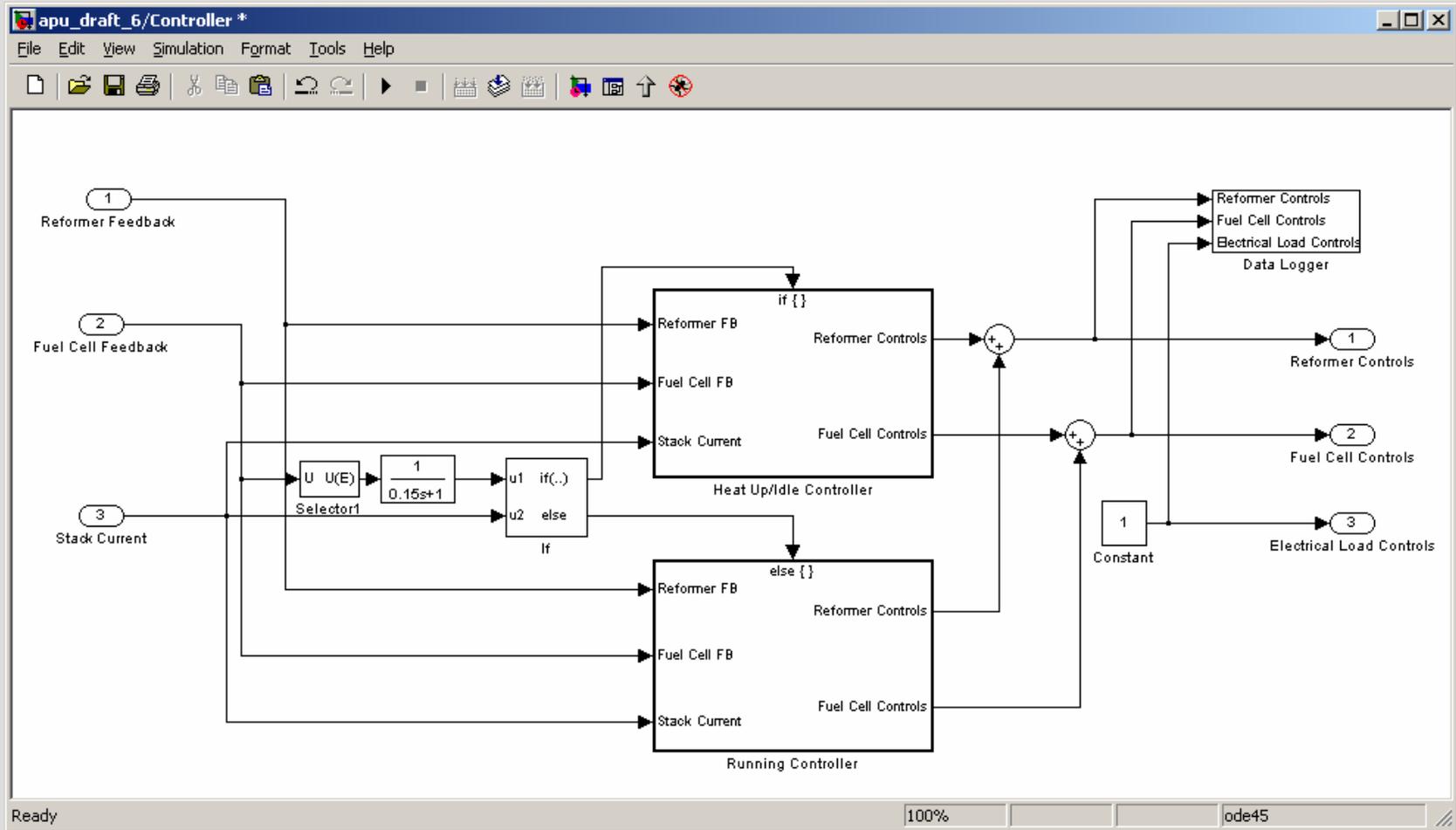
# APU Control

- ▶ The APU system is very complex and as electrical load demands change, the operating point of the reformer/fuel cell must be changed appropriately.
- ▶ An APU controller must control variables such as: fuel flow rate, reformate composition, cathode flow rate and temperatures throughout the system.
- ▶ Many of these variables are dependent on each other, and the controller must respond to potentially fast load changes.

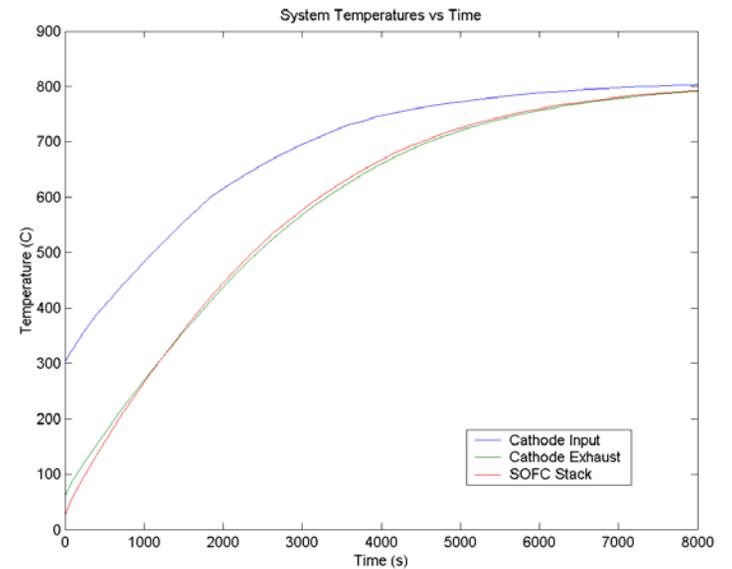
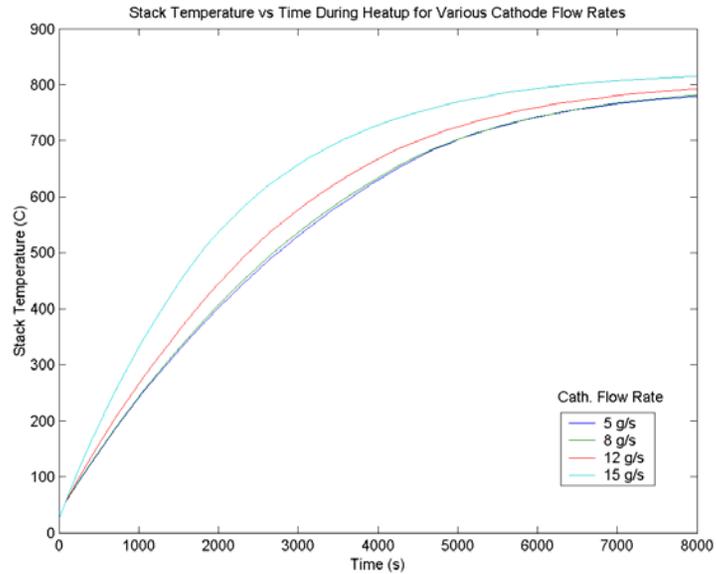
# Initial Control

- ▶ Using a two phase controller – heat up/idle and operating.
- ▶ Heat up/idle controller
  - Uses a lookup table based on empirical data to specify desired cathode inlet temp based on stack temp and cathode flow rate.
  - Uses a PID controller to actuate HX bypass valve and control cathode temperature.
- ▶ Operating Controller
  - PID control of anode flow rate based on fuel utilization.

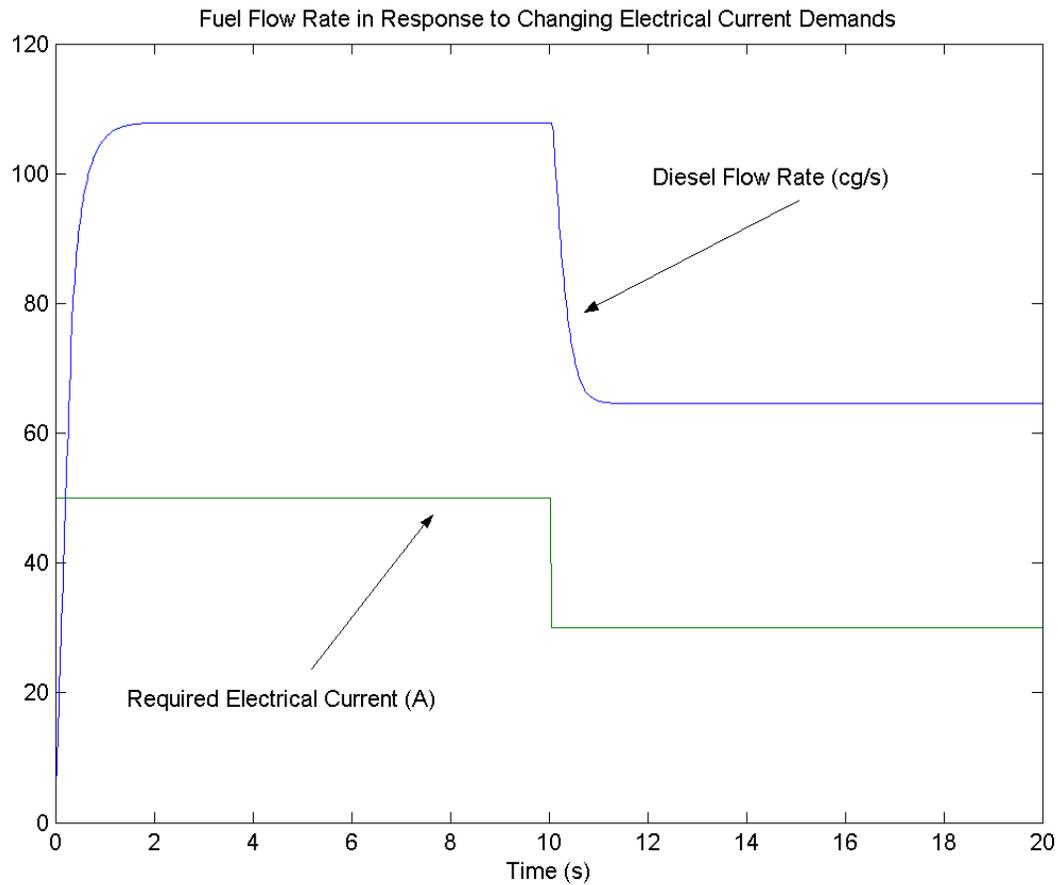
# Initial Control Strategy



# Heat Up Control



# Fuel Utilization Control



# Summary

- ▶ Matlab and Simulink are powerful simulation environments for dynamic systems.
- ▶ Simulink's graphical user interface makes it easy to use and models easy to create and change.
- ▶ At PNNL we are using Simulink to develop control strategies and model SOFC based APUs for long haul trucks.