Getting Started with Simulink

Overview of MATLAB Modeling/ Simulation Environment
MATLAB/Simulink Applications

- Mechanical System
- Automotive
- Controls
- Robotics
- Aerospace and Defense
- Communications
- Electronics and Signal Processing
- Medical Instrumentation
**Model-Based Design**

- Faster, more cost-effective development of *dynamic systems* (e.g. control systems, vehicles, etc.)
- A *system model* is at the center of the development process, from requirements development, through design, implementation, and testing.
- Model - an *executable specification* (*MATLAB* codes, or a block diagram and specified parameters) that is continually refined (*optimized*) throughout the development.
- Simulation – test whether the model works correctly, and obtain results.
- Software and hardware implementation – automatic code generation
MATLAB Codes – Simulink Block & Block Parameters

```matlab
component lossy_ultracapacitor
    % Lossy Ultracapacitor
    % Models an ultracapacitor with resistive losses.
    nodes
        p = foundation.electrical.electrical; % +top
        n = foundation.electrical.electrical; % -bottom
    end
    parameters
        C1 = [1, 'F']; % Nominal capacitance C1 at V=1
        C0 = [2, 'F/V']; % Rate of change of C with voltage V
        R = [2, 'Ohm']; % Effective series resistance
        R0 = [2, 'Ohm']; % Self-discharge resistance
        V0 = [0, 'V']; % Initial voltage
    end
    variables
        i = {v, 'A'}; % Current through variable
        v = {0, 'V'}; % Voltage across variable
        v1 = {0, 'V'}; % Internal variable
    end
    function setup
        if R <= [0, 'Ohm']
            error('Resistance must be greater than zero')
        end
        through(1, p.i, n.i); % Through variable 1
        across(1, v, p.v, n.v); % Across variable v
        v1 = V0;
    end
    equation
        i == (C0 + C1*v)*dv/dt + v/R0; % Equation 1
        v == v1 + i; % Equation 2
    end
end
```

\[ i = \left( C_0 + C_1 v \right) \frac{dv}{dt} + \frac{v}{r_d} \]
Modeling Process

On Paper:
1 Defining the System
2 Identifying System Components
3 Modeling the System with Equations

Using MATLAB/Simulink:
4 Building the Simulink Block Diagram
5 Running the Simulation
6 Validating the Simulation Results
The leading environment for technical computing

- The *de facto* industry-standard, high-level programming language for algorithm development
- Numeric computation
- Data analysis and visualization
- Toolboxes for signal and image processing, statistics, optimization, symbolic math, and other areas
- Foundation of MathWorks products

from Bryan Zocco & Doug Eastman’s Presentation
The leading environment for system-level modeling, simulation, and verification of communications and electronic systems

- **Multidomain** system-level design and verification
- Digital, analog, and mixed-signal simulation using discrete-time, continuous-time, state machine, and discrete event modeling
- Floating- and fixed-point **algorithm** development using MATLAB, Simulink blocks, or existing C code
- **Blocksets** for signal processing, video processing, communications, and RF
- **Open architecture** with links to third-party tools and development boards, and instrumentation
- C and HDL **code generation** for DSPs, embedded processors, and FPGAs

From Bryan Zocco & Doug Eastman’s Presentation
From Research to Development and Test

DATA ANALYSIS
- Data Analysis, Modelling & Visualization
- Mathematical Modeling
- Algorithm Development & Simulation

SYSTEM-LEVEL DESIGN
- System Modeling, Simulation and Partitioning
  - Environment Effects
  - Embedded Algorithms
  - System Components
- Other Design Flows

VALIDATION/VERIFICATION
- Test and Verification
- Hardware-in-the-Loop Test
- Embedded Software
- Embedded Hardware

IMPLEMENTATION

from Bryan Zocco & Doug Eastman's Presentation
Graphical Layout of Functional Modules
Complex System Model from Basic Building Blocks

Vehicle and Control
Simulink Library (blocks)
<table>
<thead>
<tr>
<th>Block Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commonly Used Blocks</td>
<td>Contains a group of the most commonly used blocks, such as the Constant, In1, Out1, Scope, and Sum blocks. Each of the blocks in this library are also included in other libraries.</td>
</tr>
<tr>
<td>Continuous</td>
<td>Contains blocks that model linear functions, such as the Derivative and Integrator blocks.</td>
</tr>
<tr>
<td>Discontinuities</td>
<td>Contains blocks with outputs that are discontinuous functions of their inputs, such as the Saturation block.</td>
</tr>
<tr>
<td>Discrete</td>
<td>Contains blocks that represent discrete time functions, such as the Unit Delay block.</td>
</tr>
<tr>
<td>Logic and Bit Operations</td>
<td>Contains blocks that perform logic or bit operations, such as the Logical Operator and Relational Operator blocks.</td>
</tr>
<tr>
<td>Sinks</td>
<td>Contains blocks that display or export output, such as the Out1 and Scope blocks.</td>
</tr>
<tr>
<td>Sources</td>
<td>Contains blocks that generate or import system inputs, such as the Constant, In1, and Sine Wave blocks.</td>
</tr>
<tr>
<td>Block Library</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LookUp Tables</td>
<td>Contains blocks that use lookup tables to determine their outputs from their inputs, such as the Cosine and Sine blocks.</td>
</tr>
<tr>
<td>Math Operations</td>
<td>Contains blocks that perform mathematical and logical functions, such as the Gain, Product, and Sum blocks.</td>
</tr>
<tr>
<td>Model Verification</td>
<td>Contains blocks that enable you to create self-validating models, such as the Check Input Resolution block.</td>
</tr>
<tr>
<td>Model-Wide Utilities</td>
<td>Contains blocks that provide information about the model, such as the Model Info block.</td>
</tr>
<tr>
<td>Ports &amp; Subsystems</td>
<td>Contains blocks that allow you to create subsystems, such as the In1, Out1, and Subsystem blocks.</td>
</tr>
<tr>
<td>Signal Attributes</td>
<td>Contains blocks that modify the attributes of signals, such as the Data Type Conversion block.</td>
</tr>
<tr>
<td>Signal Routing</td>
<td>Contains blocks that route signals from one point in a block diagram to another, such as the Mux and Switch blocks.</td>
</tr>
<tr>
<td>User-Defined Functions</td>
<td>Contains blocks that allow you to define custom functions, such as the Embedded MATLAB™ Function block.</td>
</tr>
<tr>
<td>Additional Math &amp; Discrete</td>
<td>Contains two additional libraries for mathematical and discrete function blocks.</td>
</tr>
</tbody>
</table>
## Key Multiphysics Modeling Toolbox

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateflow™</td>
<td>Design and simulate state machines and control logic.</td>
</tr>
<tr>
<td>SimMechanics™</td>
<td>Model and simulate mechanical systems.</td>
</tr>
<tr>
<td>SimPowerSystems™</td>
<td>Model and simulate electrical power systems.</td>
</tr>
<tr>
<td>Simulink® Control Design™</td>
<td>Design and analyze control systems in Simulink.</td>
</tr>
<tr>
<td>SimScape™</td>
<td>Provides expanded capabilities for modeling physical systems (mechanical, electrical, hydraulic, and other physical domains as physical networks)</td>
</tr>
<tr>
<td>SimDriveline™</td>
<td>Modeling and simulating the mechanics of driveline (drivetrain) systems</td>
</tr>
</tbody>
</table>
Modeling Dynamic Systems in Simulink

Modeling Approaches

First Principles Modeling

Simulink
- Simscape
- SimMechanics
- SimDriveline
- SimHydraulics
- SimPowerSystems

Data-Driven Modeling

Simulink Design Optimization
System Identification Toolbox
Neural Network Toolbox

Tools for Modeling Dynamic Systems

ADVISOR
PSAT/Autonomie
SimDriveline

Tools for Modeling Vehicle Powertrains

Modified from Bryan Zocco & Doug Eastman’s Presentation
SimDriveline™ Model

Diagram showing a driveline system with labels for engine, torque converter, transmission, drive shaft, rear differential, and tire components.
Simulink Online Help

- Simulink Getting Started Guide
- Simulink User’s Guide
- Simulink Reference
- Writing S-Functions
- Simulink Release Notes
- Other Posted References

Homework: build the Simulink models following the model building examples
Getting started with Simulink

An introductory tutorial

ES205 Analysis and Design of Engineering Systems
Rose-Hulman Institute of Technology

© R. Layton 2001
Launch Simulink

In the MATLAB command window,
at the >> prompt, type **simulink**
and press ← Enter
Create a new model

- Click the new-model icon in the upper left corner to start a new Simulink file.
- Select the Simulink icon to obtain elements of the model.
Your workspace

Library of elements

Model is created in this window
Save your model

- You might create a new folder, like the one shown below, called `simulink_files`
- Use the `.mdl` suffix when saving
Example 1: a simple model

- Build a Simulink model that solves the differential equation
  \[ \dot{x} = 3 \sin(2t) \]
- Initial condition \( x(0) = -1 \).
- First, sketch a simulation diagram of this mathematical model (equation) (3 min.)
Simulation diagram

- **Input is the forcing function** $3\sin(2t)$
- **Output is the solution of the differential equation** $x(t)$ where $x(0) = -1$

Now build this model in Simulink
Select an input block

Drag a *Sine Wave* block from the *Sources* library to the model window.
Select an operator block

Drag an *Integrator* block from the *Continuous* library to the model window
Select an output block

Drag a *Scope* block from the *Sinks* library to the model window
Connect blocks with signals

- Place your cursor on the output port (>) of the *Sine Wave* block.
- Drag from the *Sine Wave* **output** to the *Integrator* **input**.
- Drag from the *Integrator* **output** to the *Scope* **input**.

Arrows indicate the direction of the signal flow.
Select simulation parameters

Double-click on the *Sine Wave* block to set amplitude = 3 and freq = 2.

This produces the desired input of $3\sin(2t)$
Select simulation parameters

Double-click on the *Integrator* block to set initial condition $= -1$.

This sets our IC $x(0) = -1$. 
Select simulation parameters

Double-click on the *Scope* to view the simulation results
Run the simulation

In the model window, from the *Simulation* pull-down menu, select *Start*.

View the output $x(t)$ in the *Scope* window.
Simulation results

To verify that this plot represents the solution to the problem, solve the equation analytically.

The analytical result,

\[ x(t) = \frac{1}{2} - \frac{3}{2} \cos(2t) \]

matches the plot (the simulation result) exactly.
Example 2

- Build a Simulink model that solves the following differential equation (ODE)
  - 2nd-order mass-spring-damper system
  - zero ICs
  - input $f(t)$ is a step with magnitude 3
  - parameters: $m = 0.25$, $c = 0.5$, $k = 1$

$$m\ddot{x} + c\dot{x} + kx = f(t)$$
Create the simulation diagram

- On the following slides:
  - The simulation diagram for solving the ODE is created step by step.
  - After each step, elements are added to the Simulink model.

- Optional exercise: first, sketch the complete diagram (5 min.)

\[ m\ddot{x} + c\dot{x} + kx = f(t) \]
First, solve for the term with highest-order derivative

\[ m\ddot{x} = f(t) - c\dot{x} - kx \]

Make the left-hand side of this equation the output of a summing block
Drag a *Sum* block from the *Math* library.

Double-click to change the block parameters to *rectangular* and *+- -*.
Add a gain (multiplier) block to eliminate the coefficient and produce the highest-derivative alone.
Drag a \textit{Gain} block from the \textit{Math} library.

The gain is 4 since $\frac{1}{m}=4$.

Double-click to change the block parameters. Add a title.
Add integrators to obtain the desired output variable.

- $m\ddot{x}$
- $\frac{1}{m} \ddot{x}$
- $\frac{1}{s} \dot{x}$
- $\frac{1}{s} x$

Diagram:
- Summing block
- Integrators
- Outputs
Drag *Integrator* blocks from the *Continuous* library.

Initial Conditions (ICs) on the integrators are zero.

Add a scope from the *Sinks* library. Connect output ports to input ports. **Label the signals** by double-clicking on the leader line.
Connect to the integrated signals with gain blocks to create the terms on the right-hand side of the EOM.

\[ m\ddot{x} \xrightarrow{1/m} \frac{1}{s} \xrightarrow{1/s} x \]

Diagram:
- Summing block
- Gain blocks
- Input: \( m\ddot{x} \)
- Middle stage: \( \frac{1}{m} \)
- Next stage: \( \frac{1}{s} \)
- Final stage: \( \frac{1}{s} \)
- Output: \( x \)

Symbols:
- \( m \): Mass
- \( s \): Laplace variable
- \( c \): Gain
- \( k \): Gain
Drag new *Gain* blocks from the *Math* library.

To flip the gain block, select it and choose *Flip Block* in the *Format* pull-down menu.

- Double-click on gain blocks to set parameters.
- Connect from the gain block input backwards up to the branch point.
- Re-title the gain blocks.
Complete the model

- Bring all the signals and inputs to the summing block.
- Check signs on the summer.
Double-click on Step block to set parameters. For a step input of magnitude 3, set Final value to 3.
Final Simulink model
Run the simulation
Results

Underdamped response.
Overshoot of 0.5.
Final value of 3 (gain = 1).
Is this expected?
System design – adjust $m$, $c$, $k$ values to get different system response
Paper-and-pencil analysis based on the equations of motion

- **Standard form**
  
  \[ \frac{\ddot{x}}{k} + \frac{c}{k} \dot{x} + x = \frac{1}{k} f(t) \]

- **Nat’l freq.**
  
  \[ \omega_n = \sqrt{\frac{k}{m}} = 2.0 \]

- **Damping ratio**
  
  \[ \frac{2\zeta}{\omega_n} = \frac{c}{k} \rightarrow \zeta = 0.5 \]

- **Static gain**
  
  \[ K = \frac{1}{k} = 1 \]
Check simulation results

- Damping ratio of 0.5 is less than 1.
  - Expect the system to be underdamped.
  - Expect to see overshoot.

- Static gain is 1.
  - Expect output magnitude to equal input magnitude.
    - Input has magnitude 3, so does output.

- Simulation results conform to expectations.