Simulink: Basics

A Brief Introduction to Simulink

Bruno Abreu Calfa

Last Update: January 31, 2011

Table of Contents

Outline

Contents

1 What is Simulink?

Simulation and Model-Based Design

- Simulink is a graphical environment for simulation and model-based design for dynamic systems.
- Based on block diagrams and can be used for many applications:
  - Communications
  - Control
  - Signal Processing
  - Video Processing
  - Image Processing
- Interoperability with MATLAB
- Used in industry and academia

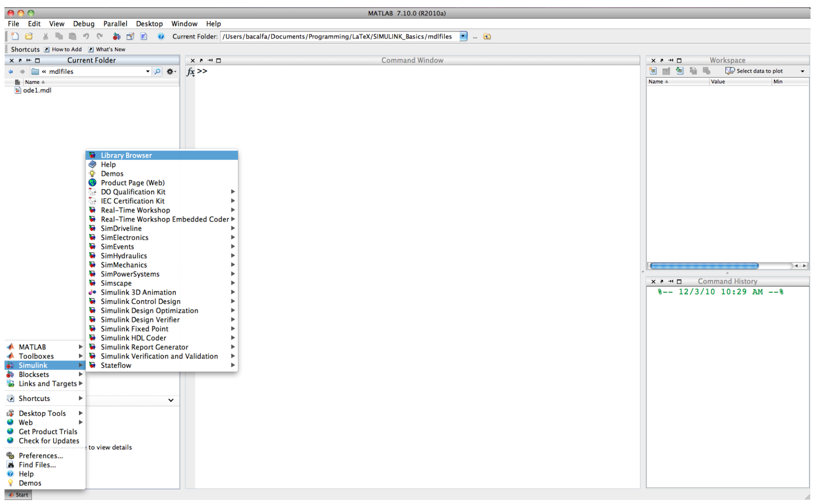
Help with Simulink?

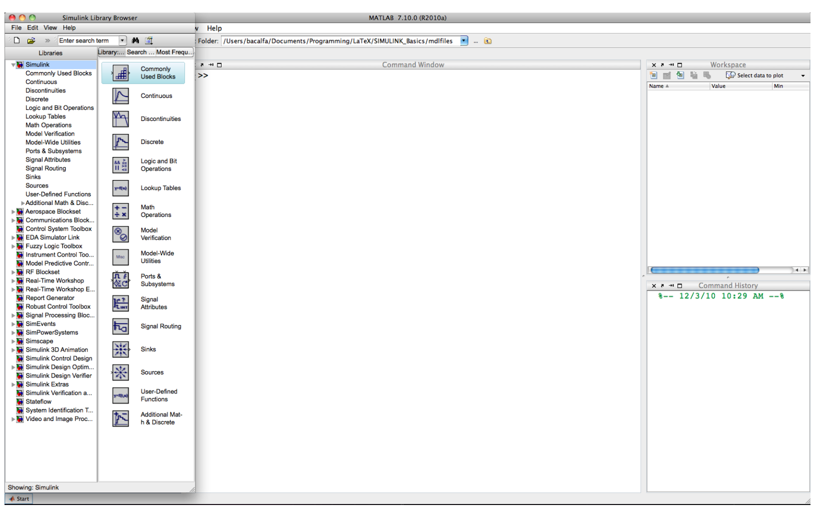
- Simulink's Help
- Google
- A book about Simulink

2 Simulink Windows

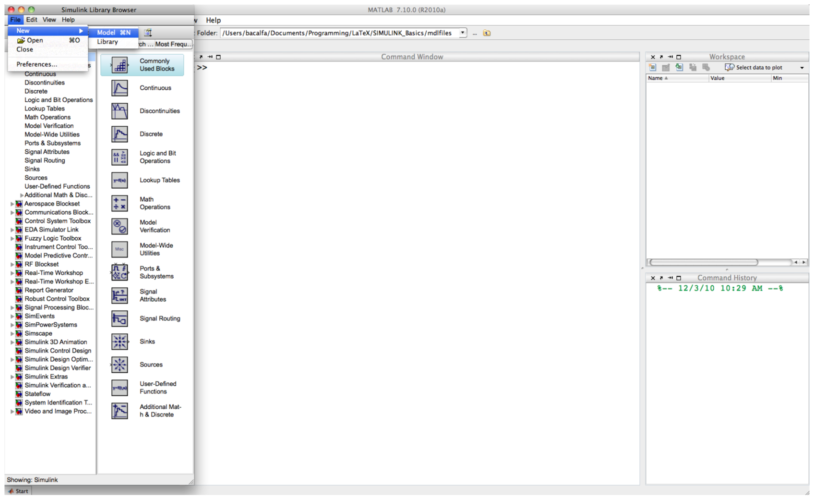
Library Browser

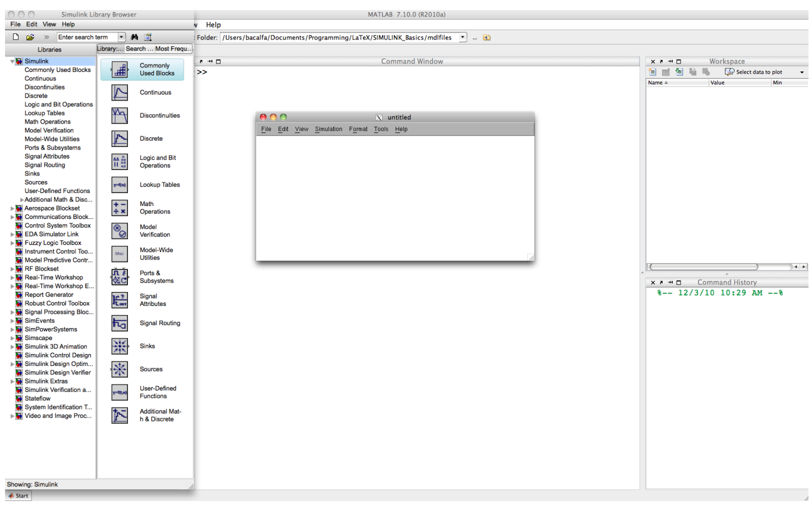
- Type "`simulink`" in Command Window or go to: Start → Simulink → Library Browser





Create New Model





3 How to Build Models

Adding and Connecting Blocks

- Pay special attention to the subcategories of "Simulink"
  - Continuous (*e.g.*: Integrator, PID Controller, Transfer Fcn)
  - Math Operations (*e.g.*: Gain, Sum)
  - Sinks (*e.g.*: Scope, To File, To Workspace)
  - Sources (*e.g.*: Clock, Constant, Step)

– User-Defined Functions (*e.g.*: S-Function)

- Click on a block, drag and drop it to the Model window.
- Add more blocks and connect their inputs and outputs (**Hint**: to connect blocks faster, hold the keyboard "control" key and click on them)

4 Examples

Solving ODEs: Intro

- Simulink can be used to solve Initial Value Problems (IVPs) of Ordinary Differential Equations (ODEs)
- First-Order ODE

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t))$$
$$\mathbf{y}(0) = \mathbf{y}_0$$

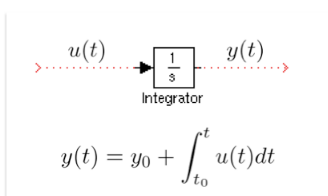
- Second-Order ODE

$$\ddot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \dot{\mathbf{y}}(t))$$
$$\mathbf{y}(0) = \mathbf{y}_0$$
$$\dot{\mathbf{y}}(0) = \dot{\mathbf{y}}_0$$

- Solving ODEs is done by *integration*, so use the "Integrator" block from the "Continuous" subcategory

Solving ODEs: Reasoning

- The "Integrator" block performs the following operation to the signal:



Solving ODEs: Example 1

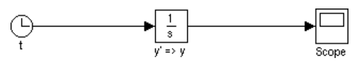
- The first example is to solve the following IVP:

$$\dot{y}(t) = t$$
$$y(0) = 0$$

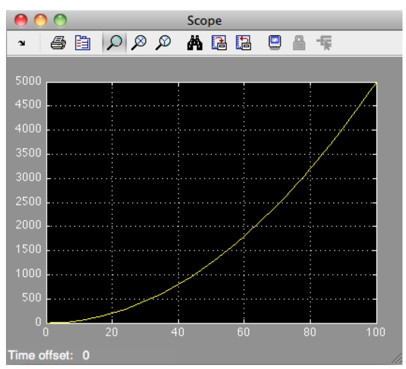
- The exact solution to that is:

$$y(t) = \frac{1}{2}t^2$$

- In Simulink, we need the following blocks:
  - **Clock**: represents time t
  - **Integrator**: represents the integration of the RHS of the ODE
  - **Scope**: plots the result
- It looks like this:



- To solve, go to: Simulation → Start or use the keyboard shortcut `control` + `T`
- The solution is:



- The model is in the file `ode1.mdl`
- **Hint**: To change the simulation time, go to: Simulation → Configuration Parameters... and adjust the "Start time" and "Stop time" accordingly.

Solving ODEs: Example 2

- Solve the following IVP:

$$\dot{y}(t) + 3y(t) = \sin 0.5t$$
$$y(0) = 1$$

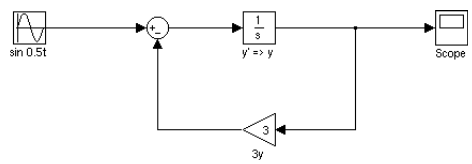
- The exact solution to that is:

$$y(t) = -\frac{2}{37}\cos 0.5t + \frac{12}{37}\sin 0.5t + \frac{39}{37}e^{-3t}$$

- Before representing the ODE in Simulink, first rewrite it to:

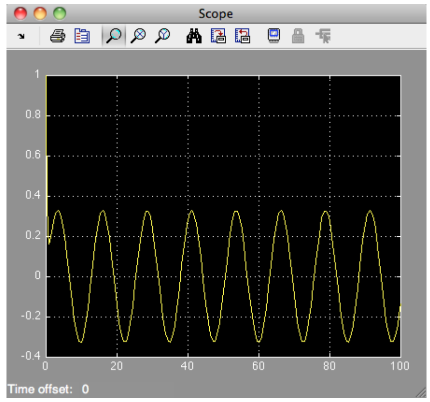
$$\dot{y}(t) = \sin 0.5t - 3y(t)$$

- We want to integrate the RHS of the rearranged ODE. Notice that now we have a sine function and a negative *gain* in $y(t)$.
- We can use the "Sine Wave" block under the Sinks subcategory to generate the first term in the RHS
- We then have to "Sum" the sinusoidal signal with the negative of $3y(t)$, which we get from multiplying the $y(t)$ by a "Gain" of 3
- In Simulink, it looks like this:



- To flip a block, right click it then go to: Format → Flip Block or use the

  keyboard shortcut `control` + `I`
- To create a node in the signal, hold the `control` key, click on it and then connect it to a block input
- To change the signs in the "Sum" block, double click it and change the order/number of signs accordingly
- To account for the initial condition (IC) valued at 1, double click the "Integrator" block and change it to 1
- The solution is:



- The model is in the file `ode2.mdl`

Solving ODEs: Example 3

- Solve the following IVP:

$$
\begin{aligned}
&2\ddot{y}(t)+3\dot{y}(t)+4y(t)=H(t)\\
&y(0)=0\\
&\dot{y}(0)=1
\end{aligned}
$$

  where $H(t)$ is the Heaviside (Step) function

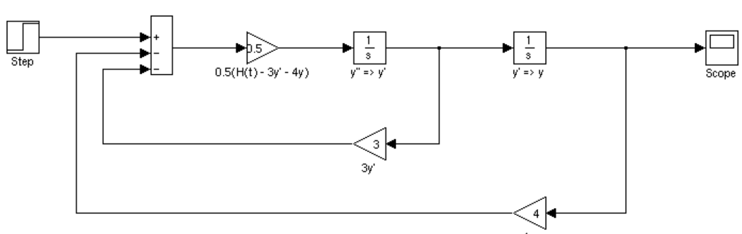
- The exact solution to that is:

$$
y(t)=\frac{4\sqrt{23}}{23}e^{-\frac{3}{4}t}\sin\left(\frac{1}{4}\sqrt{23t}\right)-\frac{3\sqrt{23}}{92}e^{-\frac{3}{4}t}\sin\left(\frac{1}{4}\sqrt{23t}\right)H(t)-
\frac{1}{4}H(t)\left[-1+e^{-\frac{3}{4}t}\cos\left(\frac{1}{4}\sqrt{23t}\right)\right]
$$

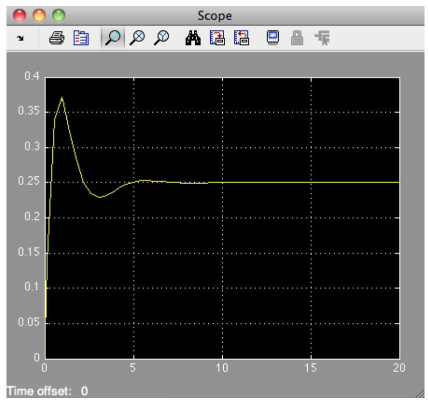
- Again, let us rewrite the ODE as:

$$\ddot{y}(t)=\frac{1}{2}\left[H(t)-3\dot{y}(t)-4y(t)\right]$$

- We need *two* "Integrators" now
- The step function is represented by the "Step" block under Sources subcategory
- It looks like this:



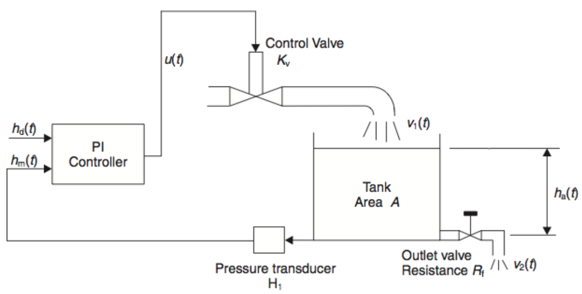
- The solution is:



- The model is in the file `ode3.mdl`

Chemical Process Control: Liquid-Level Tank

- The system is depicted below (Burns, 2001):



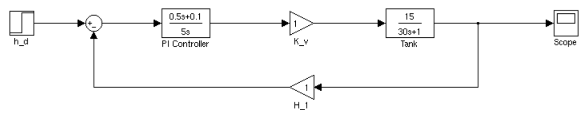
- The dynamic model of the tank is given by the following ODE:

$$A\dot{h}_a = v_1(t) - v_2(t) = K_v u(t) - \frac{h_a(t)}{R_f}$$

- Taking the Laplace Transform of both sides of the ODE yields:

$$H_a(s) = \frac{R_f}{1 + AR_f s} V_1(s)$$

- Investigate the time response for a step change in $h_d(t)$ from 0 to $4\,\mathrm{m}$
- In Simulink, it looks like this:



- The Laplace Transform of the PI controller is:

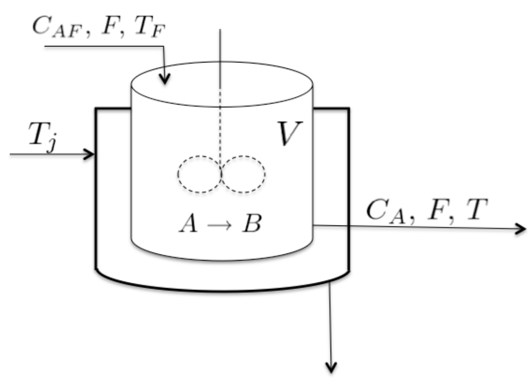
$$U(s) = K_P \left(1 + \frac{1}{\tau_I s}\right) E(s)$$

where K_P is the Proportional gain and τ_I is the Integral Time

- The model is in the file `tank.mdl`

Chemical Process Control: Non-Isothermal CSTR

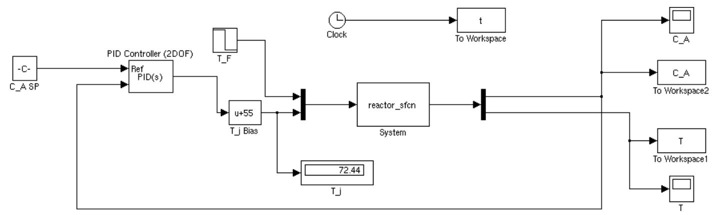
- The system is depicted below:



- The dynamic model of the system is given by the following ODEs:

$$\dot{C_A} = \frac{F}{V}(C_{AF} - C_A) - k_0 \exp\left(-\frac{E_a}{R(T+460)}\right) C_A$$
$$\dot{T} = \frac{F}{V}(T_F - T) - \frac{\Delta H}{\rho C_P}\left[k_0 \exp\left(-\frac{E_a}{R(T+460)}\right) C_A\right] - \frac{UA}{\rho C_P V}(T - T_j)$$

- **Controlled Variable**: C_A **Manipulated Variable**: T_j **Load (Disturbance)**: T_F
- Given a step disturbance in T_F, we wish to control C_A by changing T_j
- In Simulink, it looks like this:



- The system is represented by an “S-Function” block (file `reactor_sfcn.m`)
- The main script and the model are in the files `cstrmain.m` and `cstr.mdl`, respectively

5 References

References

References

R.S. Burns. *Advanced Control Engineering*. Butterworth-Heinemann, 2001.