Introduction of MATLAB and Simulink

Neeraj Kumar Singh and Marc Pantel

nsingh@enseeiht.fr
INPT-ENSEEIHT/IRIT
University of Toulouse, France

September 2023

Outline

MATLAB and Simulink

Application

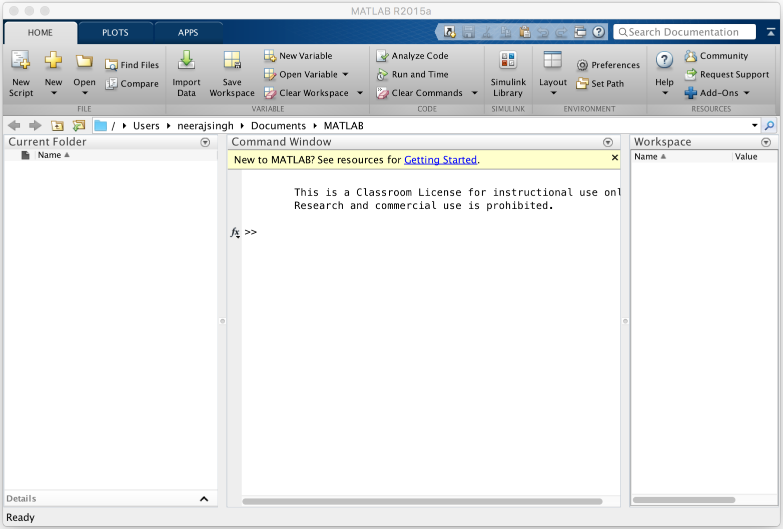
- Use of MATLAB and Simulink in Industry.

What is MATLAB?

- MATrix LABoratory
- Computational software
  - The MathWorks: `www.mathworks.com`
- Algorithm development environment
- Graphical modelling and simulation language
- Solving complex non-trivial mathematical operations, such as ODEs, root identification, and eigenvalue calculation
- Domain specific toolboxes and blocksets
- Alternative software
  - SciLab `http://www.scilab.org/`
  - GNU Octave `http://www.gnu.org/software/octave/`

MATLAB Interface

(1) Workspace (2) Current Directory
(3) Command History (4) Command Window



New to MATLAB? See resources for Getting Started.

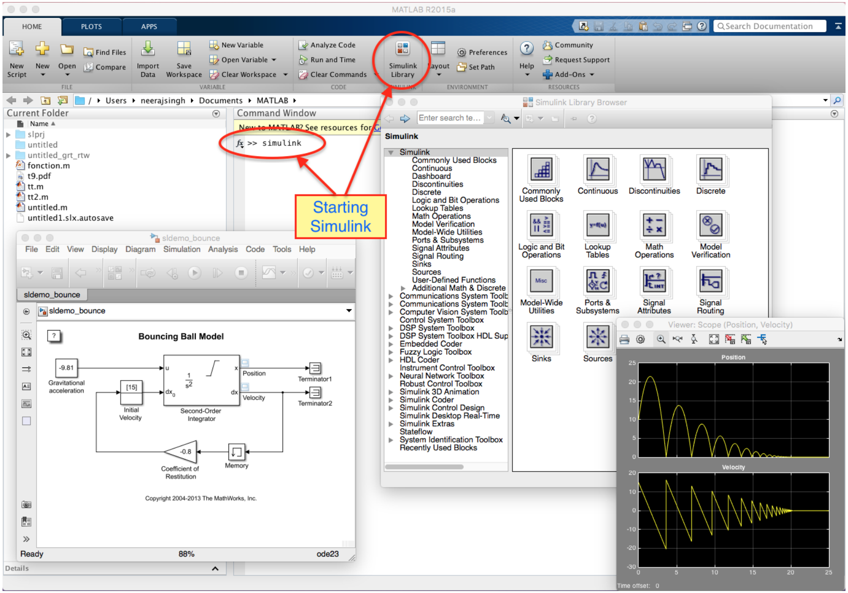
```
        This is a Classroom License for instructional use onl
        Research and commercial use is prohibited.

>>
```

Simulink

What is Simulink?

- Block based system modelling and simulation
- A collection of standard toolboxes and libraries (Machine learning, signal processing, image processing etc.)
- Direct interaction with hardware and real-time systems
- Multi-domain modelling using signal flow diagrams, state machines and physical modelling
- Heterogeneous programming environment (such as C, C++ and FORTRAN) using S-Function blocks
- Automatic code generation for deployment

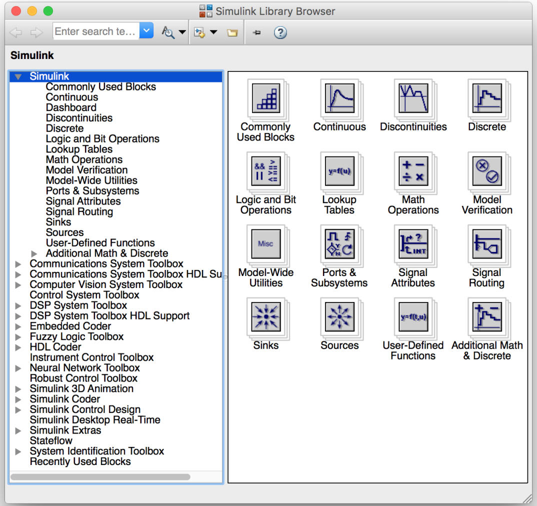
Launching Simulink



```
>> simulink
```

Simulink Library Browser

(1) Search Block (2) Block List
(3) New Simulink Model (4) Help

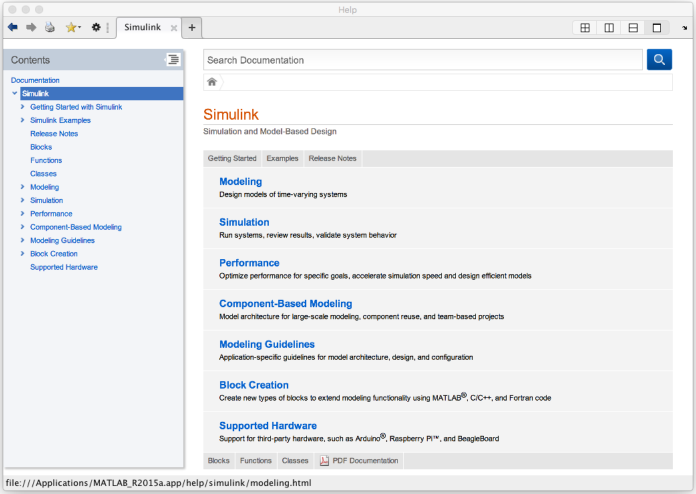

Simulink Library Browser

Enter search te...

Simulink

- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
- Communications System Toolbox
- Communications System Toolbox HDL Su
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder
- Fuzzy Logic Toolbox
- HDL Coder
- Instrument Control Toolbox
- Neural Network Toolbox
- Robust Control Toolbox
- Simulink 3D Animation
- Simulink Coder
- Simulink Control Design
- Simulink Desktop Real-Time
- Simulink Extras
- Stateflow
- System Identification Toolbox
- Recently Used Blocks

Simulink Help



Help

Simulink

Contents

Documentation

- Simulink
  - Getting Started with Simulink
  - Simulink Examples
  - Release Notes
  - Blocks
  - Functions
  - Classes
  - Modeling
  - Simulation
  - Performance
  - Component-Based Modeling
  - Modeling Guidelines
  - Block Creation
  - Supported Hardware

Search Documentation

Simulink

Simulation and Model-Based Design

Getting Started | Examples | Release Notes

Modeling
Design models of time-varying systems

Simulation
Run systems, review results, validate system behavior

Performance
Optimize performance for specific goals, accelerate simulation speed and design efficient models

Component-Based Modeling
Model architecture for large-scale modeling, component reuse, and team-based projects

Modeling Guidelines
Application-specific guidelines for model architecture, design, and configuration

Block Creation
Create new types of blocks to extend modeling functionality using MATLAB®, C/C++, and Fortran code

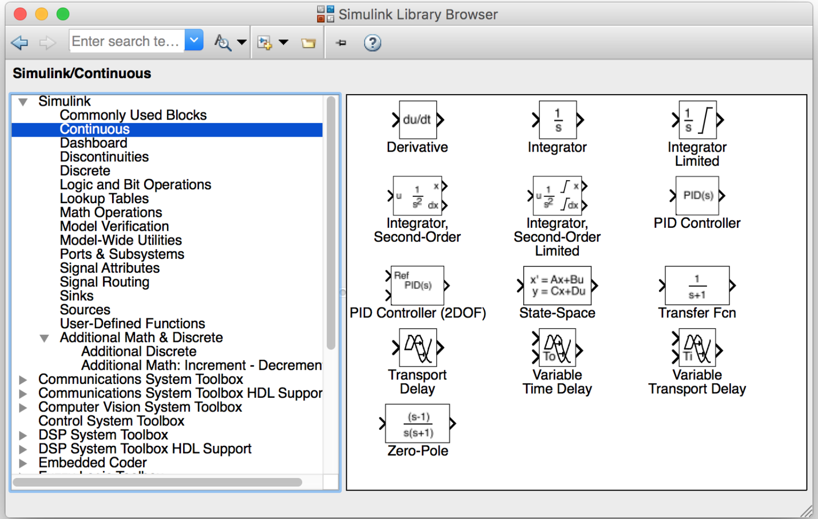
Supported Hardware
Support for third-party hardware, such as Arduino®, Raspberry Pi™, and BeagleBoard

Blocks | Functions | Classes | PDF Documentation

file:///Applications/MATLAB_R2015a.app/help/simulink/modeling.html

Use the “Help” button in the library browser for finding tutorials, demos, information on available blocks, and so on.

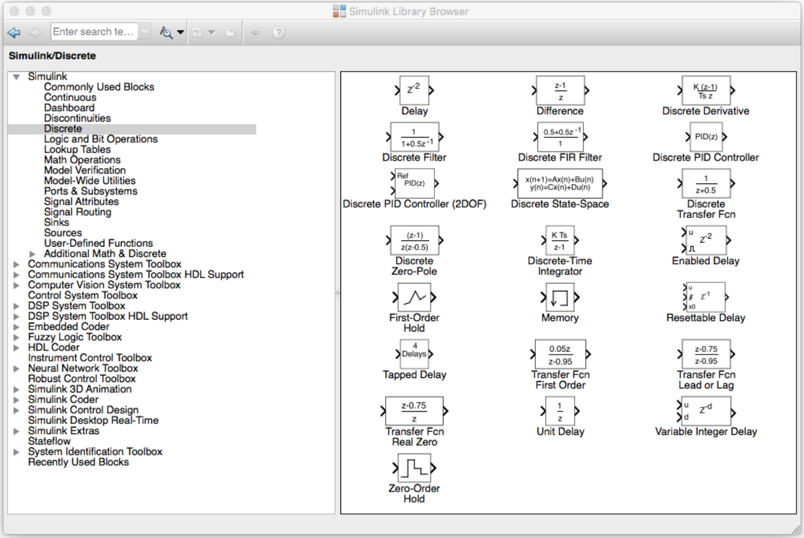
Continuous Blocks



Simulink/Continuous

- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
    - Additional Discrete
    - Additional Math: Increment - Decremen
- Communications System Toolbox
- Communications System Toolbox HDL Suppor
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder

Discrete Blocks



Simulink Library Browser

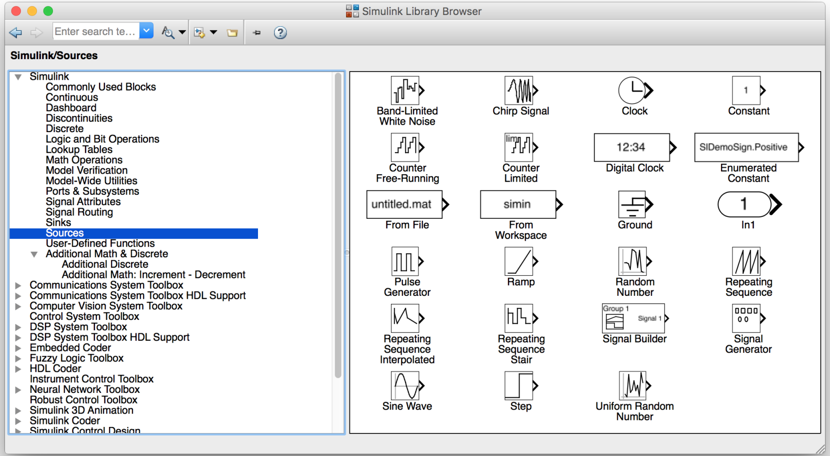
Enter search te…

Simulink/Discrete

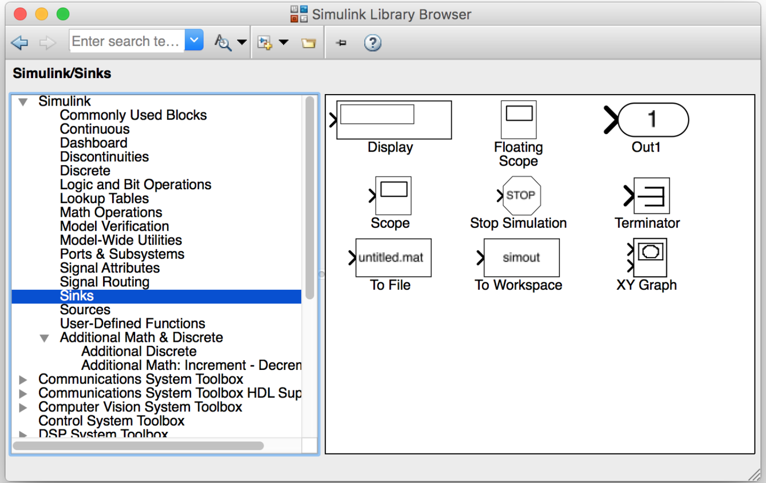
- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
- Communications System Toolbox
- Communications System Toolbox HDL Support
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder
- Fuzzy Logic Toolbox
- HDL Coder
- Instrument Control Toolbox
- Neural Network Toolbox
- Robust Control Toolbox
- Simulink 3D Animation
- Simulink Coder
- Simulink Control Design
- Simulink Desktop Real-Time
- Simulink Extras
- Stateflow
- System Identification Toolbox
- Recently Used Blocks

z^{-2} Delay	$\frac{z-1}{z}$ Difference	$\frac{K(z-1)}{Ts\,z}$ Discrete Derivative
$\frac{1}{1+0.5z^{-1}}$ Discrete Filter	$\frac{0.5+0.5z^{-1}}{1}$ Discrete FIR Filter	PID(z) Discrete PID Controller
Ref PID(z) Discrete PID Controller (2DOF)	x(n+1)=Ax(n)+Bu(n) y(n)=Cx(n)+Du(n) Discrete State-Space	$\frac{1}{z+0.5}$ Discrete Transfer Fcn
$\frac{(z-1)}{z(z-0.5)}$ Discrete Zero-Pole	$\frac{K\,Ts}{z-1}$ Discrete-Time Integrator	u z^{-2} Enabled Delay
First-Order Hold	Memory	u x0 z^{-1} Resettable Delay
4 Delays Tapped Delay	$\frac{0.05z}{z-0.95}$ Transfer Fcn First Order	$\frac{z-0.75}{z-0.95}$ Transfer Fcn Lead or Lag
$\frac{z-0.75}{z}$ Transfer Fcn Real Zero	$\frac{1}{z}$ Unit Delay	u d z^{-d} Variable Integer Delay
Zero-Order Hold		

Source Blocks



Sink Blocks

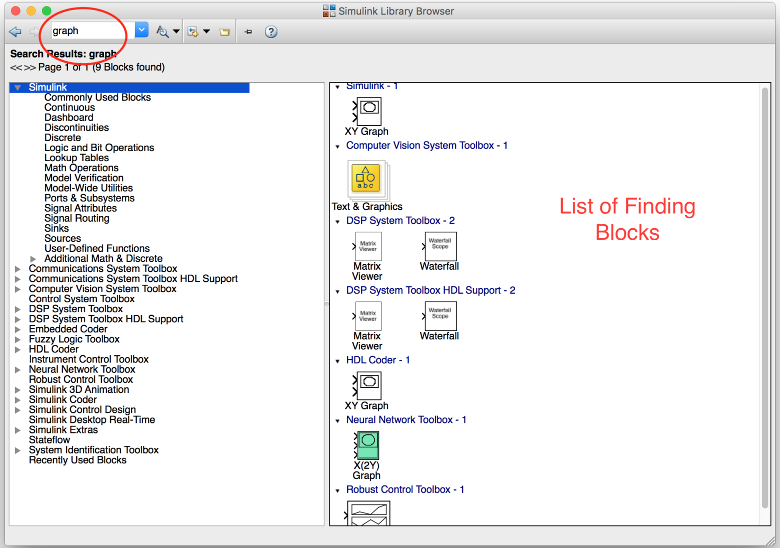


Simulink Library Browser

Simulink/Sinks

- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
    - Additional Discrete
    - Additional Math: Increment - Decren
- Communications System Toolbox
- Communications System Toolbox HDL Sup
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox

Finding Blocks



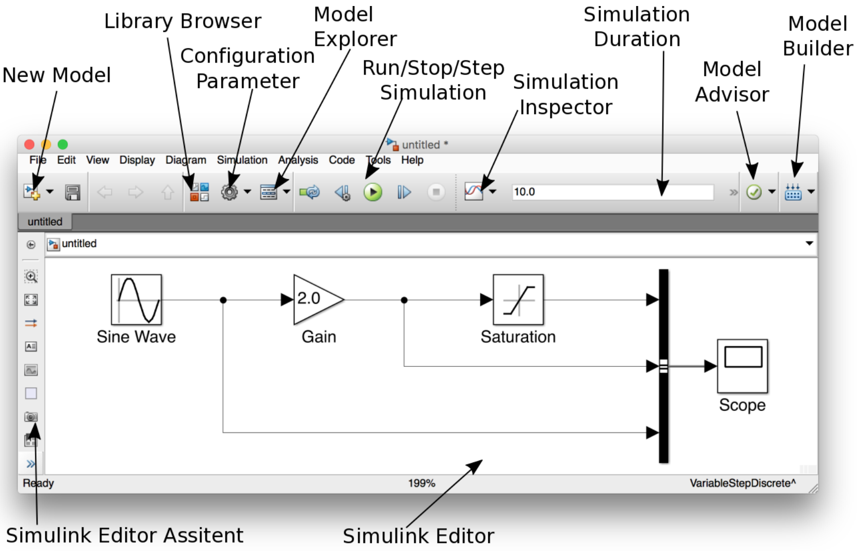
Search Results: graph
<<>> Page 1 of 1 (9 Blocks found)

- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
- Communications System Toolbox
- Communications System Toolbox HDL Support
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder
- Fuzzy Logic Toolbox
- HDL Coder
- Instrument Control Toolbox
- Neural Network Toolbox
- Robust Control Toolbox
- Simulink 3D Animation
- Simulink Coder
- Simulink Control Design
- Simulink Desktop Real-Time
- Simulink Extras
- Stateflow
- System Identification Toolbox
- Recently Used Blocks

Simulink - 1
XY Graph
Computer Vision System Toolbox - 1
Text & Graphics
DSP System Toolbox - 2
Matrix Viewer
Waterfall
DSP System Toolbox HDL Support - 2
Matrix Viewer
Waterfall
HDL Coder - 1
XY Graph
Neural Network Toolbox - 1
X(2Y) Graph
Robust Control Toolbox - 1

List of Finding Blocks

Simulink Window (Editor)



Importing and Exporting Data

Importing and Exporting Data

Importing Data into Simulink

- In
- Constant
- From Workspace
- From File

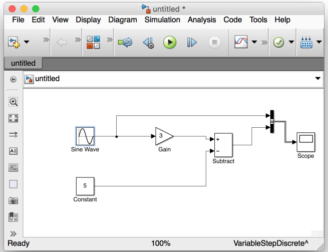
Exporting Data from Simulink

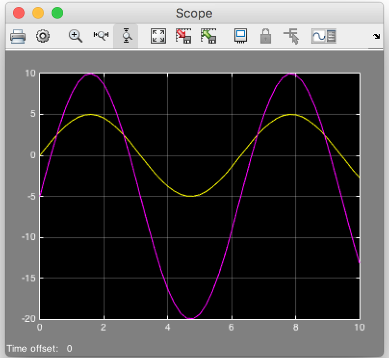
- Out
- To Workspace
- To File

System Modelling

System Modelling: Example

$$y = 3 * Sin(t) - 5$$

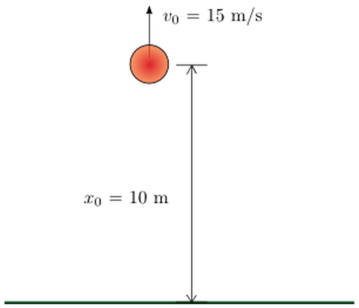




Continuous and Discrete Modelling

Continuous Systems

Modelling of a Bouncing Ball

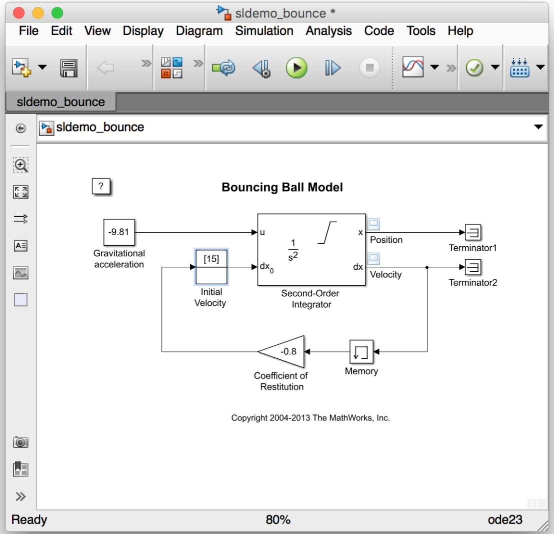


$$\frac{dx}{dt} = v \tag{1}$$

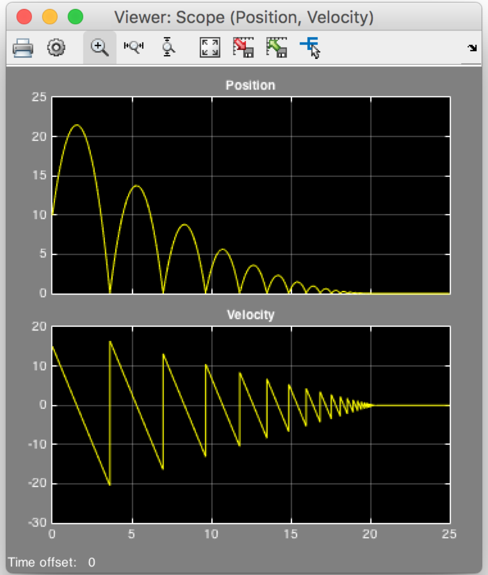
$$\frac{dv}{dt} = -g \tag{2}$$

Coefficient of restitution $(v^+ = K.v^-)$

Simulink Model of a Bouncing Ball



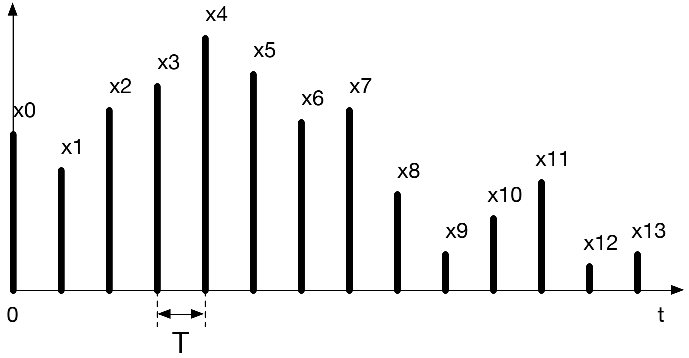
Simulation Results of a Bouncing Ball



Discrete Systems

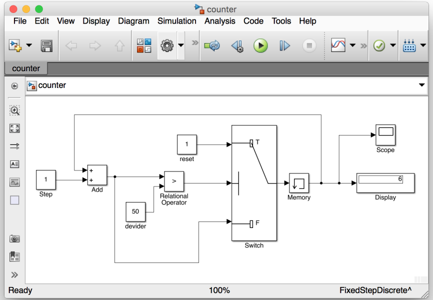
Discrete Systems

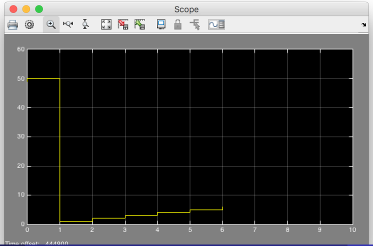
A discrete signal is a signal that has values only at discrete points in time.
A sampled signal is a discrete signal.



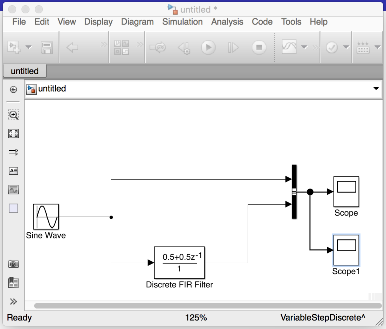
The sample period (T) is the time between two successive samples.

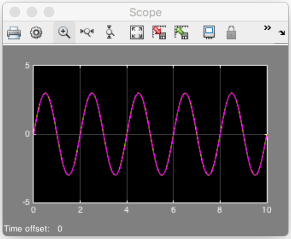
System Modelling: Digital Counter

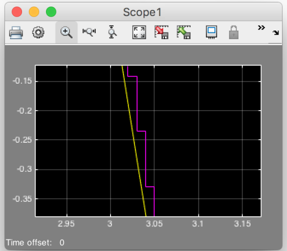




System Modelling: Example







Solver

- Determines solution at current time step
- Determines the next simulation time step

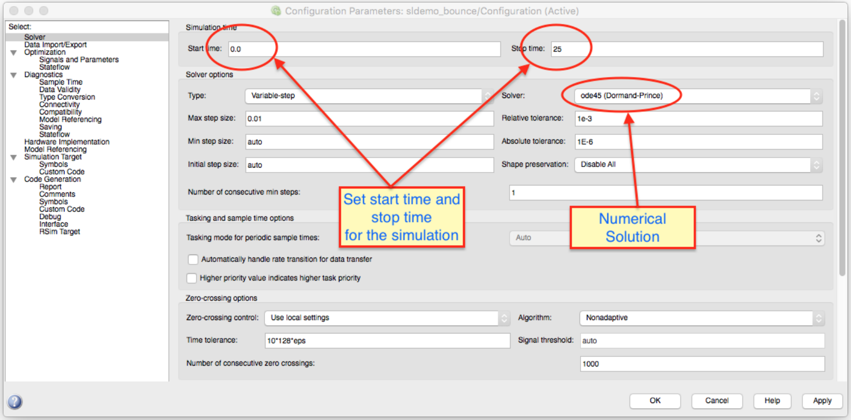
Variable step solver

The time step added to the current time can vary depending on the dynamics of the system. (i.e. ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb)

Fixed step solver

Step size remains constant. They do not control integration errors or detect discontinuities. (ode1, ode2, ode3, ode4, ode5, ode8)

Solver


Configuration Parameters: sldemo_bounce/Configuration (Active)

Select:
- Solver
- Data Import/Export
- Optimization
  - Signals and Parameters
  - Stateflow
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
  - Saving
  - Stateflow
- Hardware Implementation
- Model Referencing
- Simulation Target
  - Symbols
  - Custom Code
- Code Generation
  - Report
  - Comments
  - Symbols
  - Custom Code
  - Debug
  - Interface
  - RSim Target

Simulation time

Start time: 0.0
Stop time: 25

Solver options

Type: Variable-step
Solver: ode45 (Dormand-Prince)
Max step size: 0.01
Relative tolerance: 1e-3
Min step size: auto
Absolute tolerance: 1E-6
Initial step size: auto
Shape preservation: Disable All
Number of consecutive min steps: 1

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

Subsystem

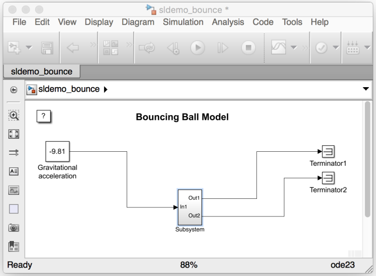
Subsystem

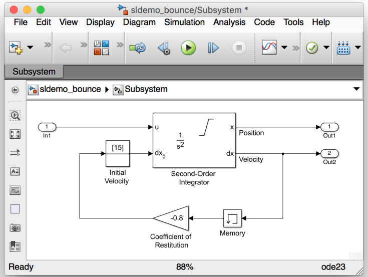
A subsystem block allows to contains a subset of blocks or code within a model to organise and to provide a hierarchical layout or to form a virtual subsystem.

The prime benefits are given as follows:

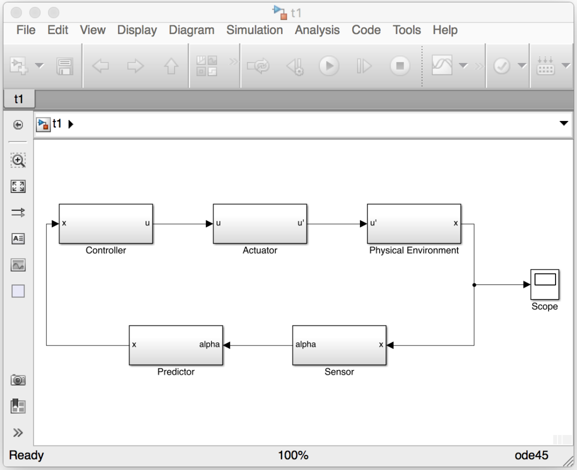
- To reduce a set of displayed blocks in a model window
- To keep functionally related block together to increase comprehensibility
- To provide a better layout in form of hierarchical block diagram

Subsystem: Example





Closed-loop Modelling



Stateflow

Stateflow

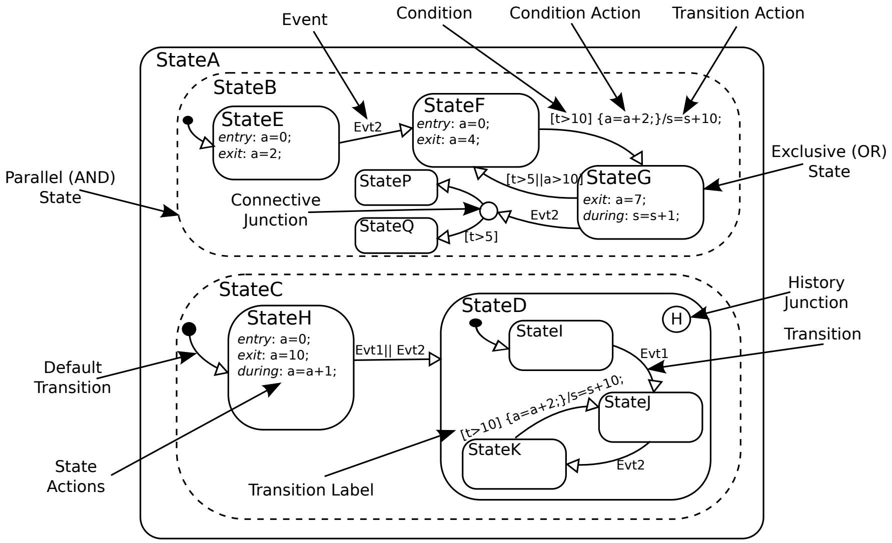
Stateflow is a graphical modelling language for specifying the behaviours of reactive systems using hierarchical state machines, similar to those of Statecharts (semantically different). It includes

- event broadcasting
- interlevel transition
- complex transition through junctions

Main Elements of Stateflow

- States
- Transitions
- Events
- Junctions

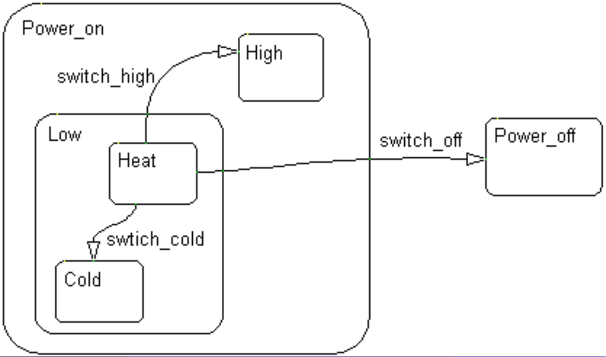
Stateflow Components



States

A state describes a mode of a reactive system. The activity or inactivity of the states dynamically changes based on events and conditions. States can be defined as:

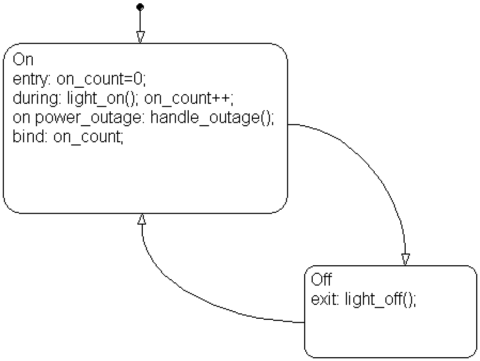
- Superstate
- Substate
- State



State Actions

States can have different types of actions, which can be executed in a sequence. For example,

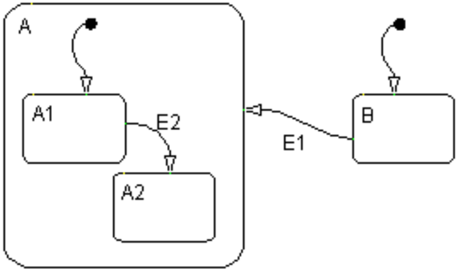
- *entry* actions, performed when entering a state
- *exit* actions, performed when leaving the state
- *during* actions, performed when remaining in the state
- etc.

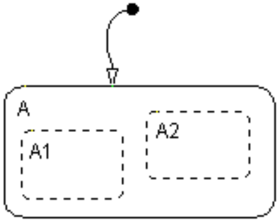

On
entry: on_count=0;
during: light_on(); on_count++;
on power_outage: handle_outage();
bind: on_count;
Off
exit: light_off();

State Decomposition

The decomposition of a state defines the same compositional structure of substates. There two types of compositional structure.

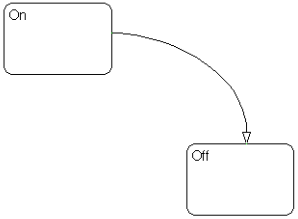
- Exclusive (OR)
- Parallel (AND)

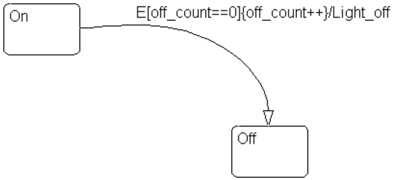




Transitions

A transition is a line or curve with an arrowhead that connects two states. The transition shows changing from one mode to other mode.





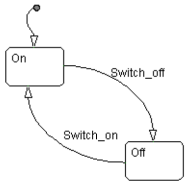
Transition Label Notation

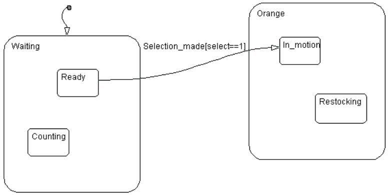
event [condition] {condition_action} / transition_action

Transition Connections

Transition

Default transitions are used to distinguish as a default enter state to avoid the ambiguity among two or more exclusive (OR)-states.





There are several other transitions. For example,

- Transition to and from OR States
- Transition to and from Junctions
- Transition to and from OR Superstates
- Transition to and from OR Substates
- Self-Loop Transitions, etc.

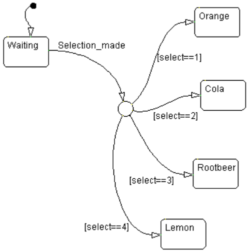
Junctions

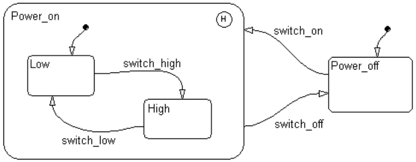
Connective Junctions

The connective junction enables representation of different possible transition paths for a single transition. It is mainly used for implementing *if − then − else*, *case*, *for loop*, etc.

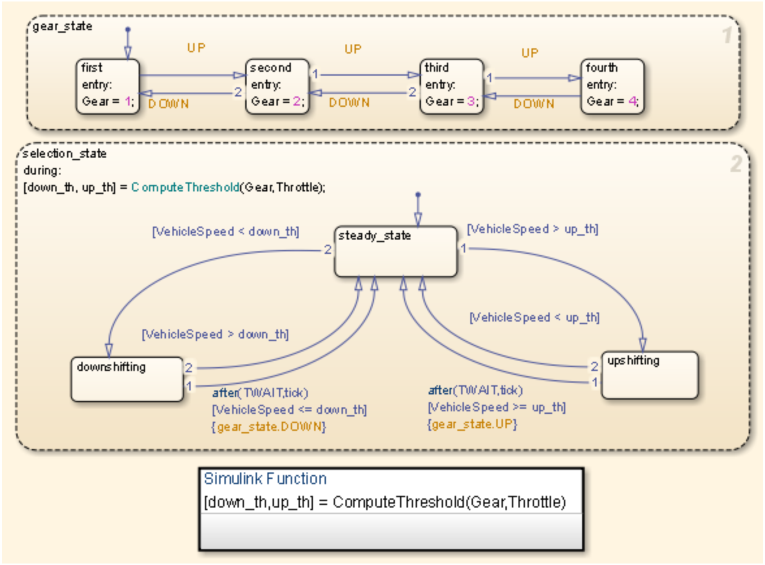
History Junctions

History junctions record the previously active state of the state in which they are resident.





Example


gear_state
1
first
entry:
Gear = 1;
second
entry:
Gear = 2;
third
entry:
Gear = 3;
fourth
entry:
Gear = 4;
UP
UP
UP
DOWN
DOWN
DOWN
selection_state
2
during:
[down_th, up_th] = ComputeThreshold(Gear,Throttle);
steady_state
[VehicleSpeed < down_th]
[VehicleSpeed > up_th]
downshifting
upshifting
[VehicleSpeed > down_th]
[VehicleSpeed < up_th]
after(TWAIT,tick)
[VehicleSpeed <= down_th]
{gear_state.DOWN}
after(TWAIT,tick)
[VehicleSpeed >= up_th]
{gear_state.UP}
Simulink Function
[down_th,up_th] = ComputeThreshold(Gear,Throttle)

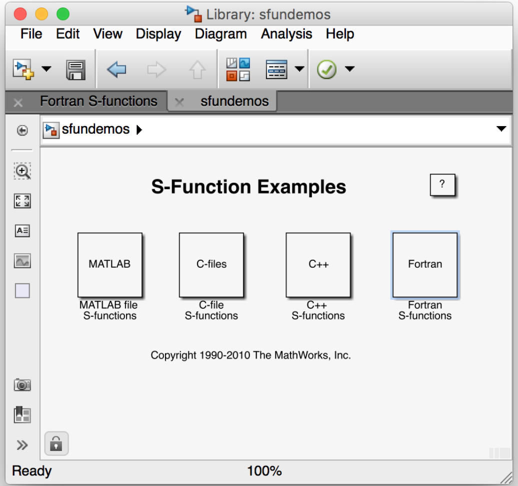
S-Function

S-Function

S-Function is a programming language environment in form of Simulink blocks that allows to add your own code to Simulink model.
The S-Function block supports the following programming languages:

- MATLAB (M-file S-Function)
- C (C Mex S-Function)
- C++ (C Mex S-Function)
- Fortran (C Mex S-Function)

S-Function Examples

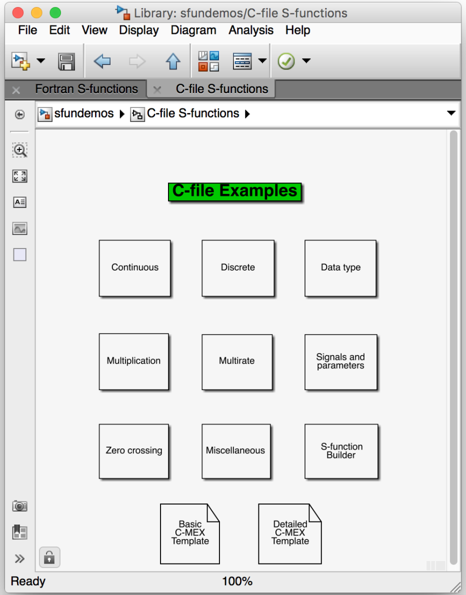


S-Function Examples

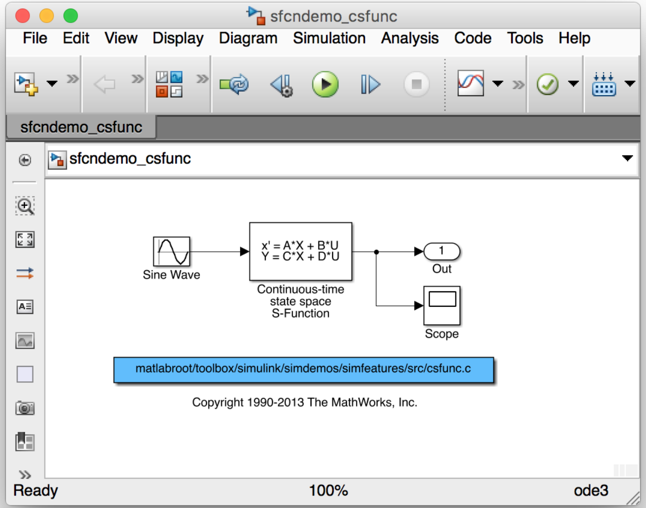
MATLAB	C-files	C++	Fortran
MATLAB file S-functions	C-file S-functions	C++ S-functions	Fortran S-functions

Copyright 1990-2010 The MathWorks, Inc.

S-Function Examples in C Language



S-Function: Continuous Time System


sfcndemo_csfunc
File Edit View Display Diagram Simulation Analysis Code Tools Help
sfcndemo_csfunc
sfcndemo_csfunc
Sine Wave
x' = A*X + B*U
Y = C*X + D*U
Continuous-time state space S-Function
1
Out
Scope
Ready
100%
ode3

matlabroot/toolbox/simulink/simdemos/simfeatures/src/csfunc.c

Copyright 1990-2013 The MathWorks, Inc.

S-Function Templates

1. C Language Template
2. MATLAB Language Template
3. ...

?
R
D