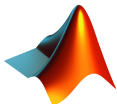


# Introduction of MATLAB and Simulink

Neeraj Kumar Singh and Marc Pantel

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

September 2024



# Outline

- 1 Introduction
- 2 Simulink
- 3 Importing and Exporting Data
- 4 System Modelling
- 5 Continuous and Discrete Modelling
- 6 Subsystems
- 7 Stateflow
- 8 S-Function

# MATLAB and Simulink

# Application

- Use of MATLAB and Simulink in Industry.

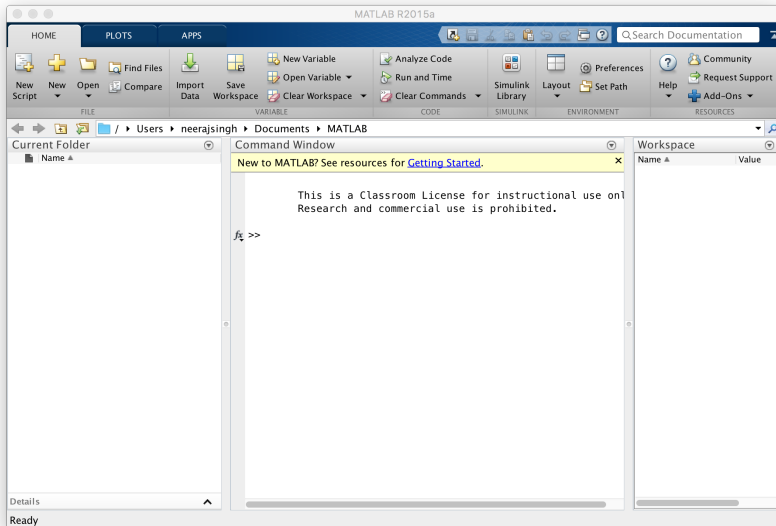


# What is MATLAB?

- MATrix LABoratory
- Computational software
  - The MathWorks: [www.mathworks.com](http://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



# 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

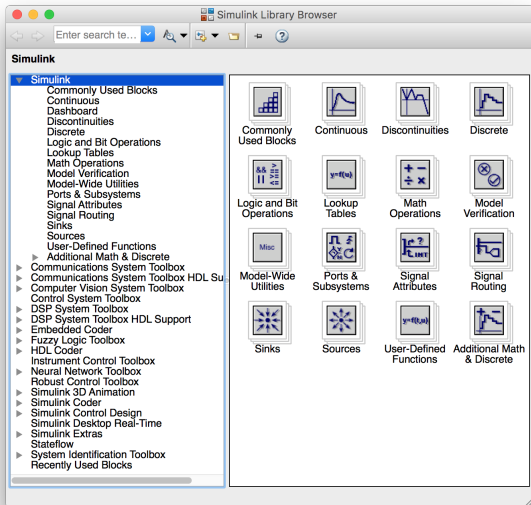


# Launching Simulink

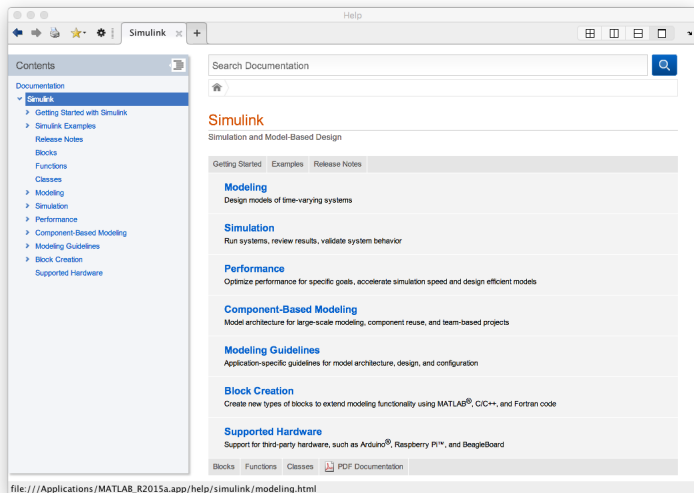
The screenshot displays the MATLAB R2015a environment. The top toolbar features the Simulink Library icon, which is circled in red. A red arrow points from this icon to the Command Window, where the command `simulink` has been entered. A yellow box with the text "Starting Simulink" is positioned near the Command Window. The Simulink Library Browser is open, showing a list of blocks categorized into Commonly Used Blocks, Continuous, Discontinuities, Discrete, Logic and Bit Operations, Lookup Tables, Math Operations, Model Verification, Ports & Subsystems, Signal Attributes, Signal Routing, Sinks, Sources, and User-Defined Functions. The main workspace shows a Simulink model titled "Bouncing Ball Model" with a block diagram. The diagram includes a "Gravitational acceleration" block, an "Initial Velocity" block, a "Second-Order Integrator" block, a "Coefficient of Restitution" block, and a "Memory" block. The model is connected to two "Terminator" blocks. The bottom status bar indicates the model is "Ready" and the simulation is at 88% completion.

# Simulink Library Browser

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

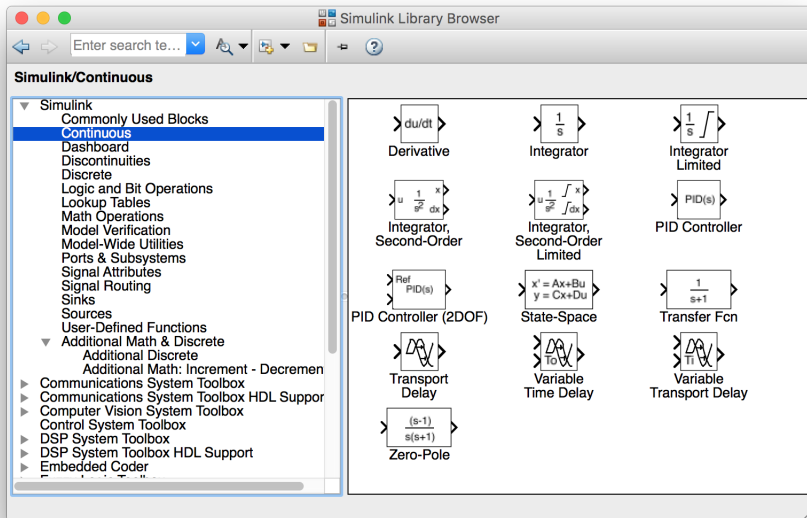


# Simulink Help

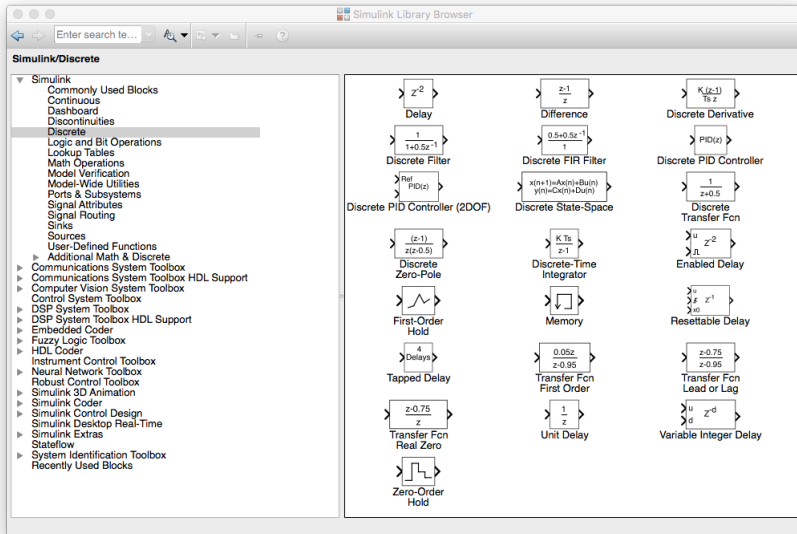


Use the “**Help**” button in the library browser for finding tutorials, demos, information on available blocks, and so on.

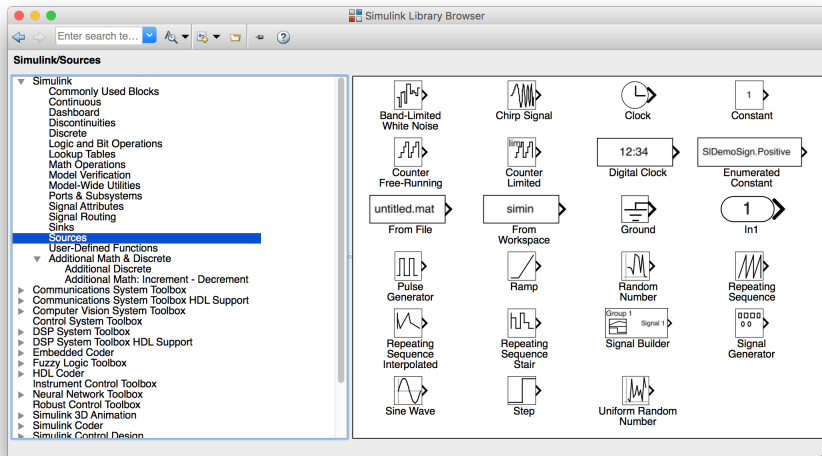
# Continuous Blocks



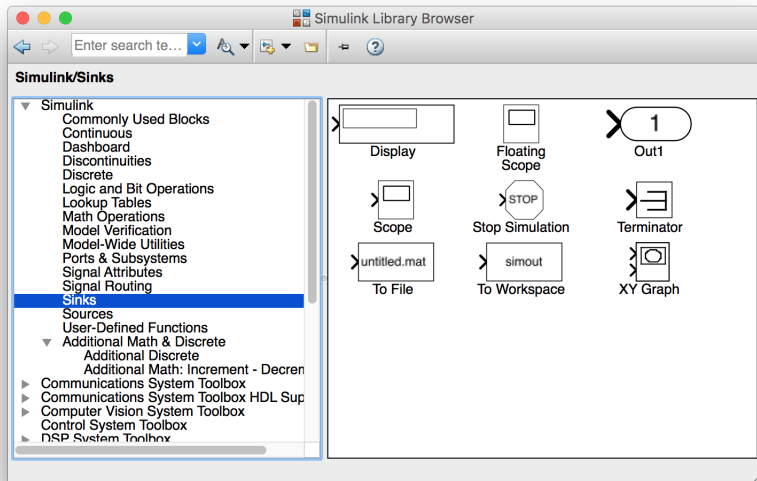
# Discrete Blocks



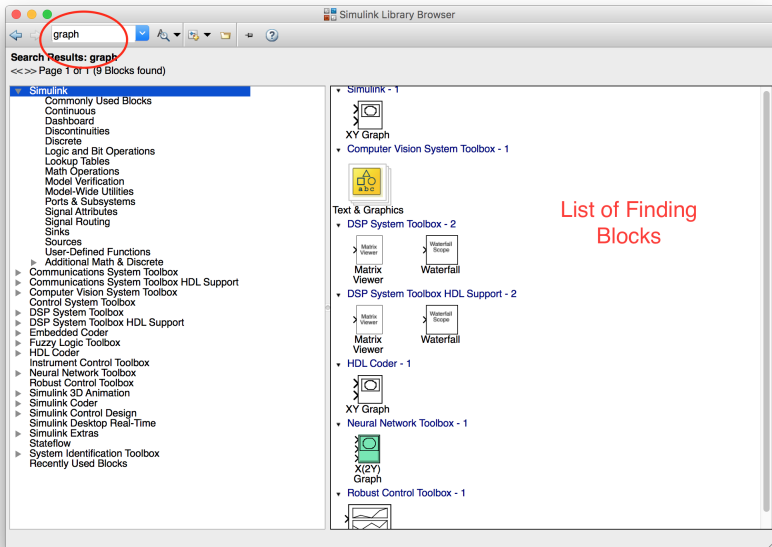
# Source Blocks



# Sink Blocks

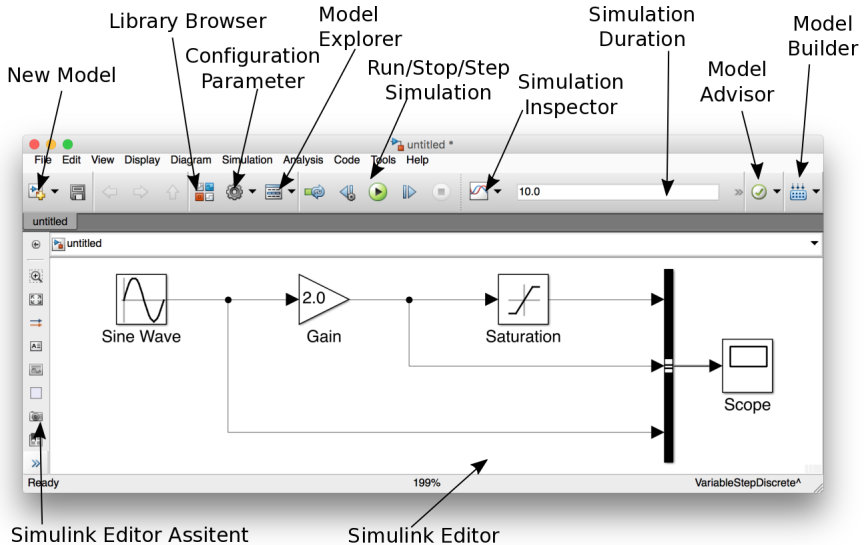


# Finding Blocks





# Simulink Window (Editor)



# Importing and Exporting Data

# Importing and Exporting Data

## Importing Data into Simulink

- In
- Constant
- From Workspace
- From File

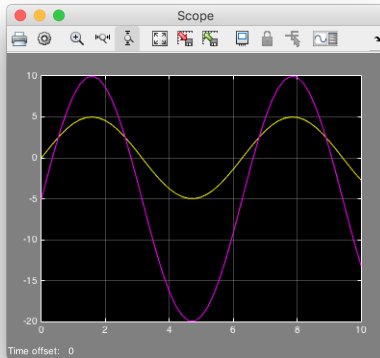
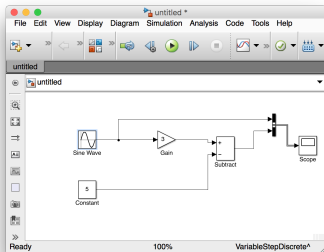
## Exporting Data from Simulink

- Out
- To Workspace
- To File

# System Modelling

# System Modelling: Example

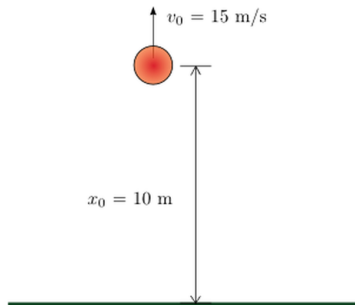
$$y = 3 * \sin(t) - 5$$



# Continuous and Discrete Modelling

# Continuous Systems

# Modelling of a Bouncing Ball



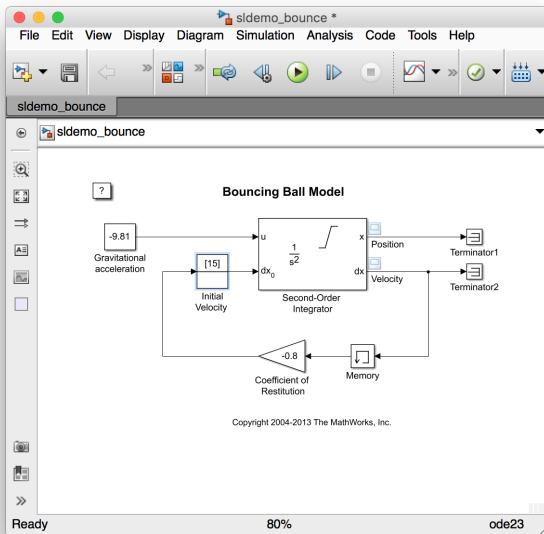
$$\frac{dx}{dt} = v \quad (1)$$

$$\frac{dv}{dt} = -g \quad (2)$$

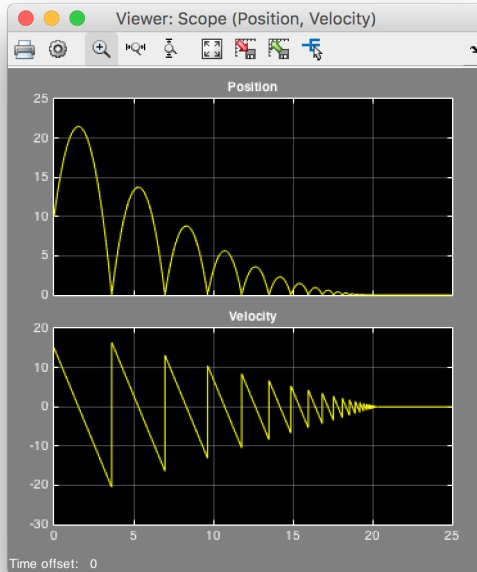
Coefficient of restitution ( $v^+ = K.v^-$ )



# Simulink Model of a Bouncing Ball



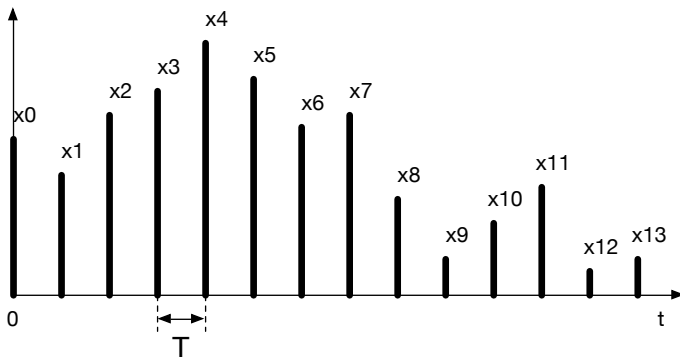
# Simulation Results of a Bouncing Ball



## Discrete Systems

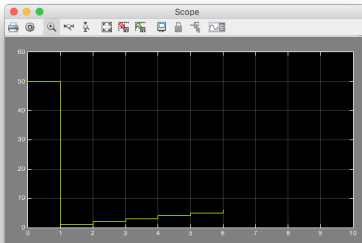
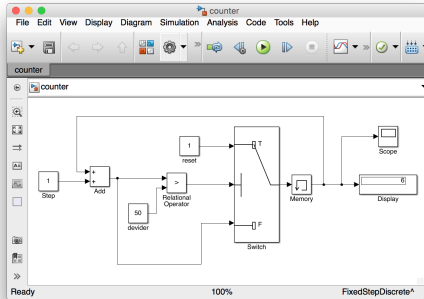
# Discrete Systems

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

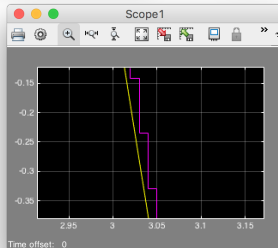
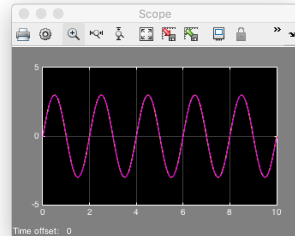
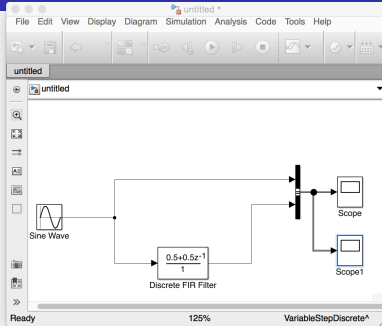


The sample period ( $T$ ) is the time between two successive samples.

# System Modelling: Digital Counter



# System Modelling: Example



# Solver

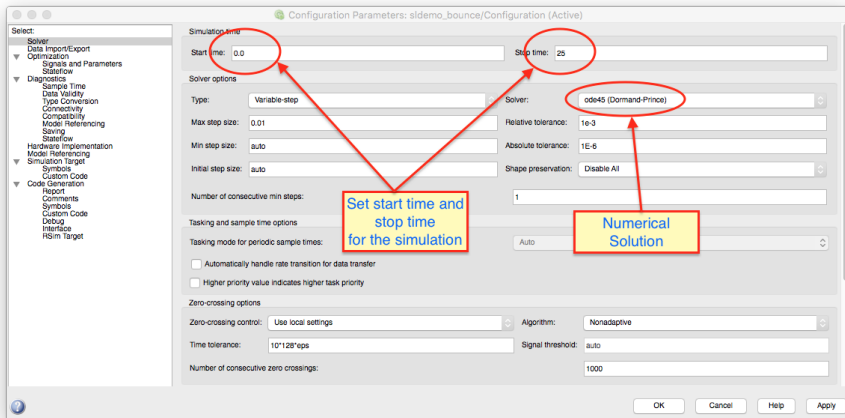
- Determines **solution** at current time step
- Determines the **next simulation** time step

## 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)





# Subsystem

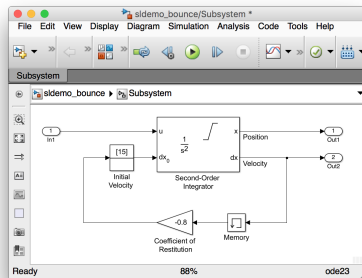
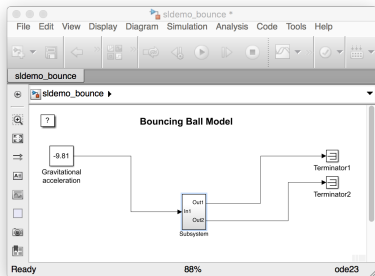
# Subsystem

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.

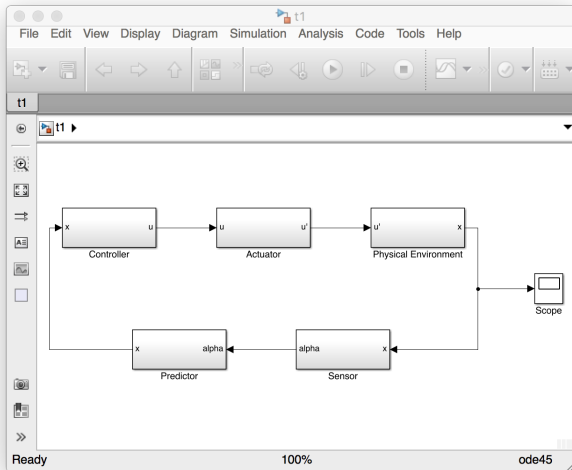
The prime benefits are given as follows:

- 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

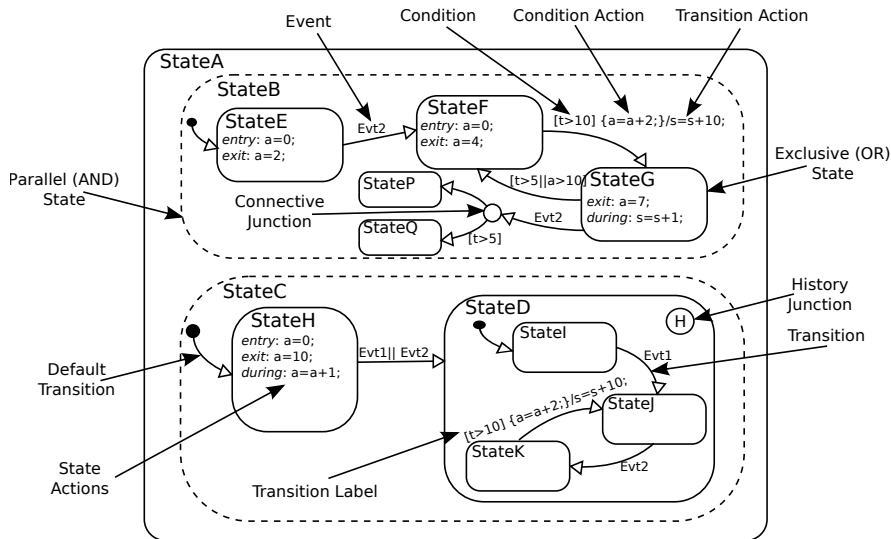
Stateflow is a **graphical modelling language** for specifying the behaviours of reactive systems using **hierarchical** state machines, similar to those of Statecharts (**semantically different**). It includes

- event broadcasting
- interlevel transition
- complex transition through junctions

## Main Elements of Stateflow

- States
- Transitions
- Events
- Junctions

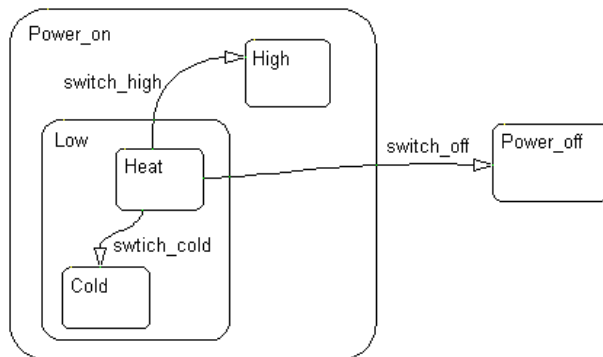
# Stateflow Components



# 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

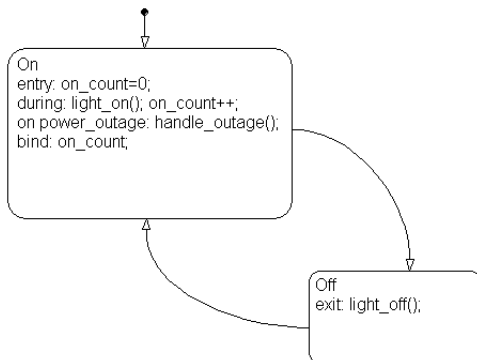




# State Actions

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

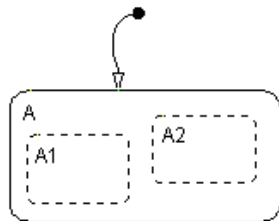
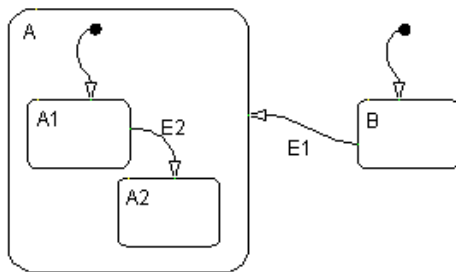
- *entry* actions, performed when entering a state
- *exit* actions, performed when leaving the state
- *during* actions, performed when remaining in the state
- etc.



# State Decomposition

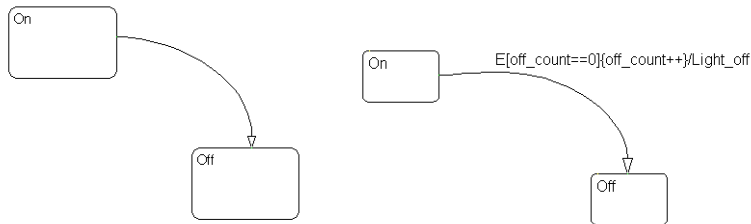
The **decomposition** of a state defines the same compositional structure of substates. There two types of compositional structure.

- 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.



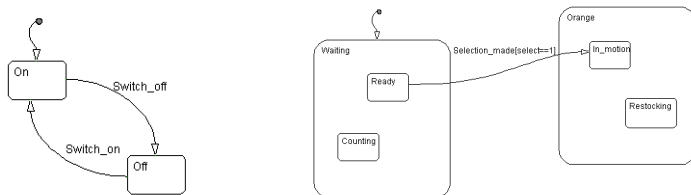
## Transition Label Notation

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.

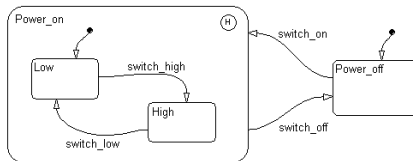
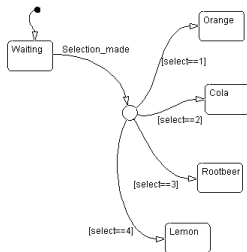
# Junctions

## Connective Junctions

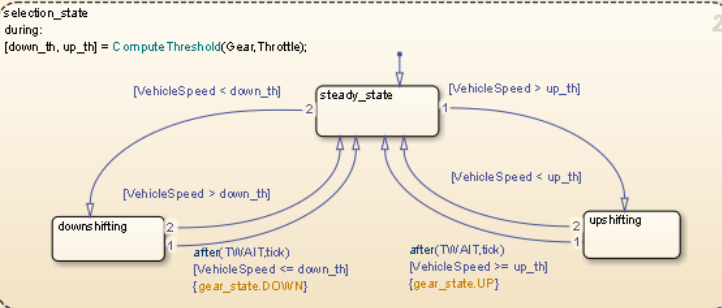
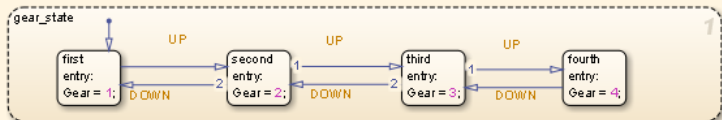
The **connective junction** enables representation of different possible transition paths for a single transition. It is mainly used for implementing *if – then – else, case, for loop, etc.*

## History Junctions

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



# Example



Simulink Function

[down\_th,up\_th] = ComputeThreshold(Gear,Throttle)

# S-Function

# S-Function

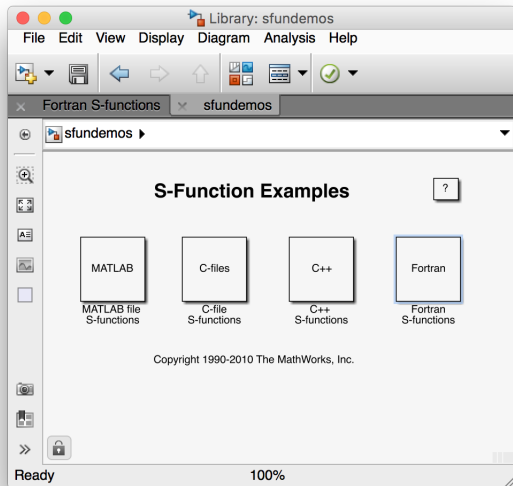
S-Function is a [programming language environment](#) in form of Simulink blocks that allows to add your [own code](#) to Simulink model.

The S-Function block supports the following programming languages:

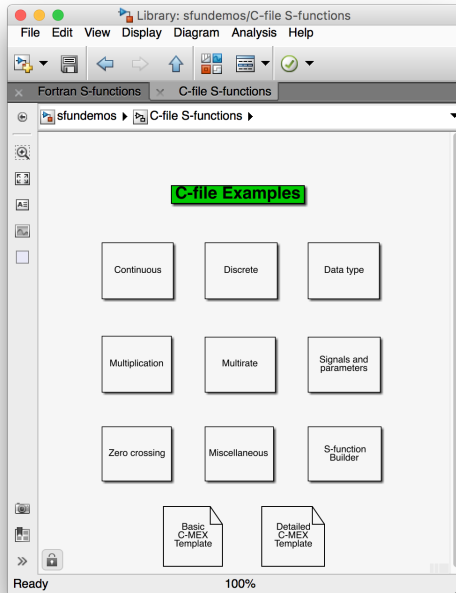
- [MATLAB](#) (M-file S-Function)
- [C](#) (C Mex S-Function)
- [C++](#) (C Mex S-Function)
- [Fortran](#) (C Mex S-Function)



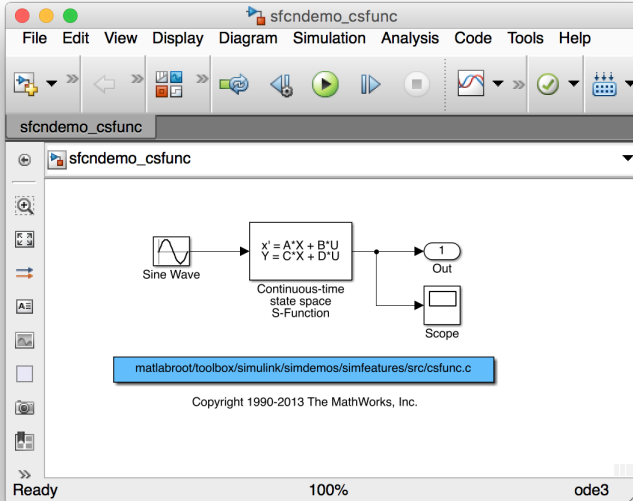
# S-Function Examples



# S-Function Examples in C Language



# S-Function: Continuous Time System



# S-Function Templates

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

