Solving Continuous & Hybrid Systems in MATLAB

Presented by
Leng-Feng

Presentation Outline

- Introduction to ODE
  - The $\dot{\underset{\sim}{y}} = \underset{\sim}{f}\left(t, \underset{\sim}{y}\right)$
  - ode function in MATLAB
- Solving ODE's in MATLAB and Simulink
  - Ball Free-Falling
    - Using MATLAB M-file
    - Using Simulink Blocks
- Hybrid System – Event Function
  - Ball Bouncing
    - Using MATLAB M-file
    - Using Simulink Blocks

ODE

- Every general explicit ordinary differential equation (ODE) can be represented by a set of equation in the form of:

$$\dot{\underset{\sim}{y}} = \underset{\sim}{f}\left(t, \underset{\sim}{y}\right)$$

- Most mechanical dynamic system can be represented in this form.

ODE (Cont.)

$$\dot{\underset{\sim}{y}} = \underset{\sim}{f}\left(t, \underset{\sim}{y}\right)$$

- What is so special?
  - They are a set of **first order** differential equations.
  - $\underset{\sim}{y}$ is a vector of the **states** (twice of the independent variables) of the system.
  - By just providing the **initial value** of the states, the system can be solved **numerically** easily. (This is how “ode” in MATLAB works)

ODE (Cont.)

$$\dot{\underset{\sim}{y}} = \underset{\sim}{f}\left(t, \underset{\sim}{y}\right)$$

- What are the states of the system?
  - The quantities that are **changing with respect to time** and **influential** to the system.
  - **Infinite many ways** of defining states. But, there are **rules of thumb**.
  - Normally, we require **minimal sets of states**. Some times, more states might be helpful.
  - Generally, in mechanical system, **the displacements** and **the velocities** of generalized coordinates are defined as the states of the system.

ode Function in MATLAB

- The routines available (problem dependent):
  - **`ode45, ode23, ode113, ode23t, ode15s, ode23s, ode23tb`**
- **`[T,Y] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS,P1,P2...)`**
  - **`ODEFUN`**: Return a column vector of first order differential equation
  - **`TSPAN`**: Vector of time interval
  - **`Y0`**: Initial condition (the length is the same as the number of states.
  - **`OPTIONS`**: Use **`odeset`** to set
  - **`P1,P2, …`**: Passing additional parameters.

Ball Free-Falling

$$\ddot{x} = -g = -9.8m/s^2$$

Drop a ball from 15m height, acceleration is –9,8m/s^2. Second order system.

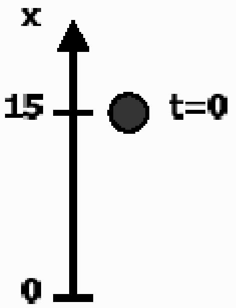
State equation, choosing the displacement and velocity as state variables (y_1, y_2):

$y_1 = x$ Displacement

$y_2 = \dot{x}$ Velocity

The time rates of change (derivatives) of the two state variables are:

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -9{,}8$$



This is our Ordinary Differential Equation (ODE).

Solving by M-file

```
[T,Y] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS,P1,P2...)
```

- Convert into the form of:

$$\dot{\underset{\sim}{y}} = \underset{\sim}{f}\left(t, \underset{\sim}{y}\right)$$

States:

$$\underset{\sim}{y} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -9{,}8$$

$$\dot{\underset{\sim}{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \boxed{\begin{bmatrix} y_2 \\ -9.8 \end{bmatrix}}$$

Vector to be returned for ODEFUN

Solving by M-file (Cont.)

$$\dot{\underset{\sim}{y}} = \boxed{\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -9.8 \end{bmatrix}}$$

```
function f=ball(t,y)

f(1)=y(2);
f(2)=-9.8;

f=[f(1) f(2)]';
```

The ODEFUN must has input arguments of time vector and the states.

Returning a column vector of first order DE.

Solving by M-file (Cont.)

```
[T,Y] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS,P1,P2...)
```

- Simulate for 2 sec.
  - `TSPAN = linspace(0,2,1001)'`
- Initial value

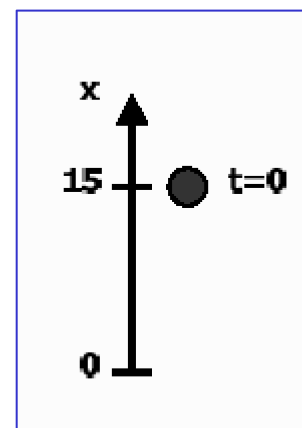
States:

$$\underset{\sim}{y} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Initial displacement = 15

Initial velocity = 0

```
Y0 = [15 0]'
```

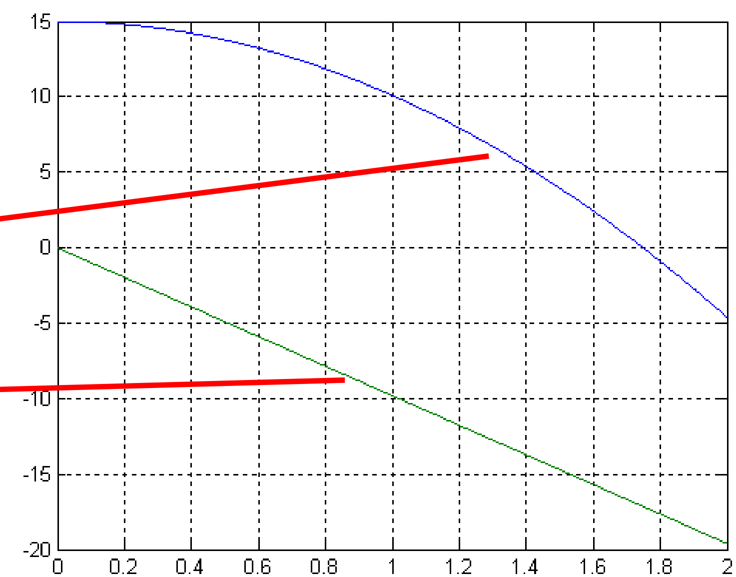


Solving by M-file (Cont.)

```
clear all
tspan=linspace(0,2,1001)';
y0=[15 0]';
[t,y]=ode45(@ball,tspan,y0);
plot(t,y)
grid on
```

Displacement profile (parabolic)

Velocity profile (linear)



Ex01_BallFalling_Main.m

```
% ----------------------------------------------------------------------
% Name: Leng-Feng Lee
% Description: Example of using MATLAB ODE for MAE505 Math Method in
%           Robotics.
% - Setting up the ODE;
% - Simulate the system.
%
% For more information about MATLAB ODE:
% http://www.mathworks.com/support/tech-notes/1500/1510.html
% ----------------------------------------------------------------------

clc
close all
clear all

Tspan = [0 2]; %simulate for 2 sec, and you don't care about specific time;
% Tspan = linspace(0,2,2001); %Simulate for 2 sec, and you want to know at
                          %each 0.001 step.

Yini = [15 0]; %Initial condition for the states.

[tout,yout] = ode45(@Ex01_Ball, Tspan, Yini);
% tout is the output time;
% yout is the output of the states in column vector;

figure(1)
%Plot the position of the ball:
plot(tout, yout(:,1),'b','linewidth',2); hold on
plot(tout, yout(:,2),':r','linewidth',2);
legend('Position', 'Velocity');
grid on
```

The Output

```
[t,y]=ode45(@ball,tspan,y0);
```

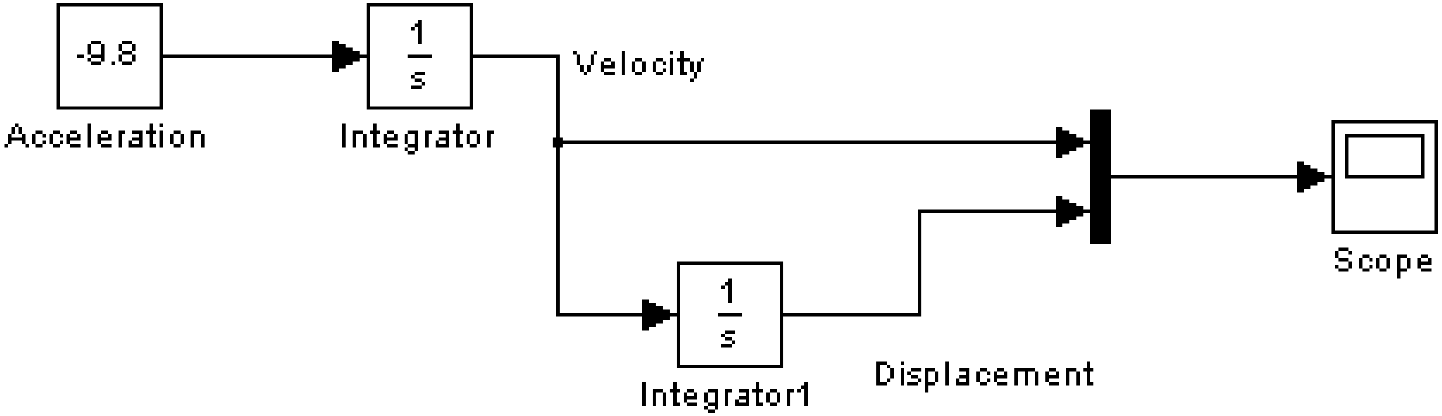
- How is **y** structured?

$$\underset{\sim}{y} = \underbrace{\begin{bmatrix} y_1(t_0) & y_2(t_0) \\ y_1(t_1) & y_2(t_1) \\ y_1(t_2) & y_2(t_2) \\ \vdots & \vdots \\ y_1(t_n) & y_2(t_n) \end{bmatrix}}_{\text{States}} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \\ a \end{bmatrix}}\right\} \text{Time span}$$

t increase

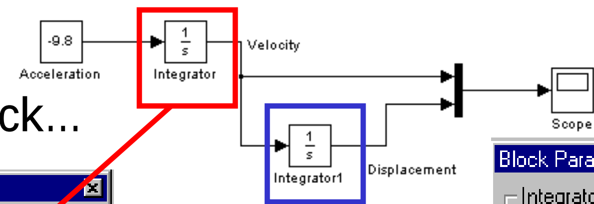
Solving by Simulink

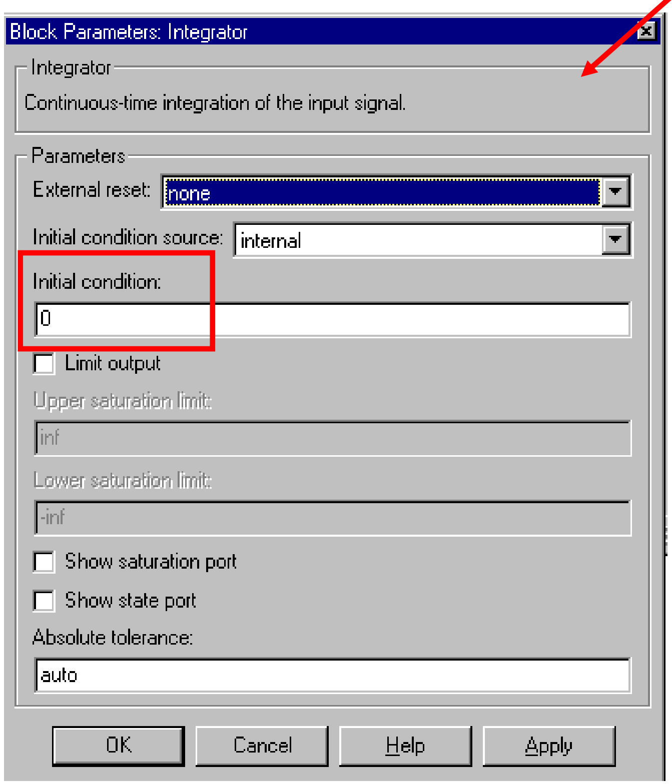
Connect the blocks:

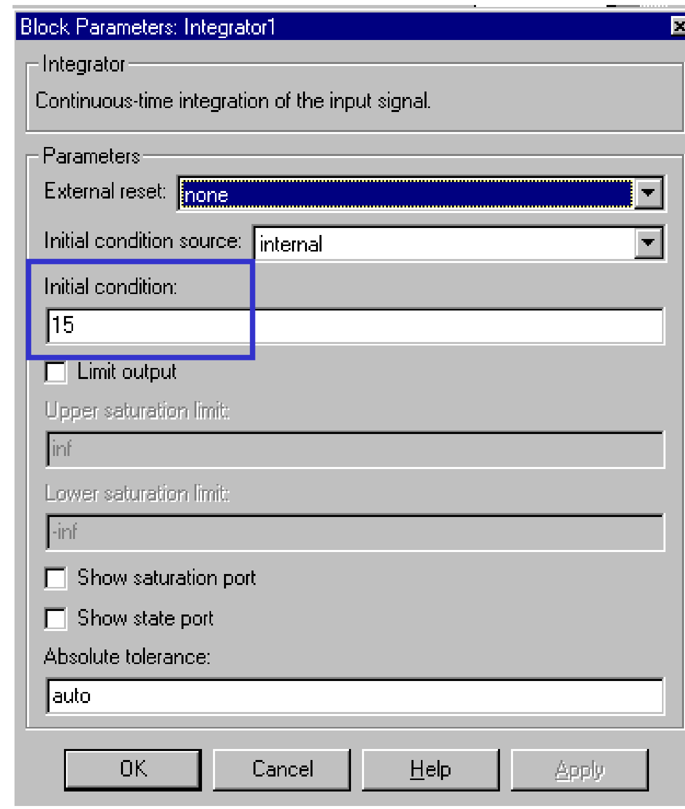


Initial Conditions?

Double click on integrator block...



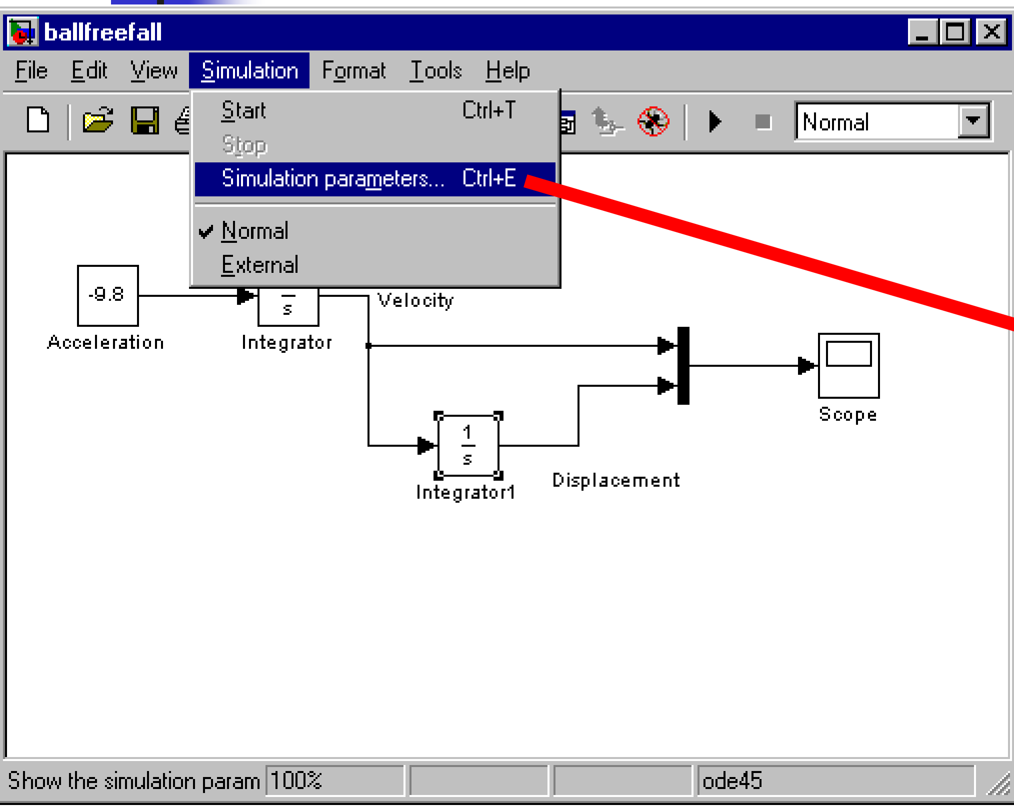


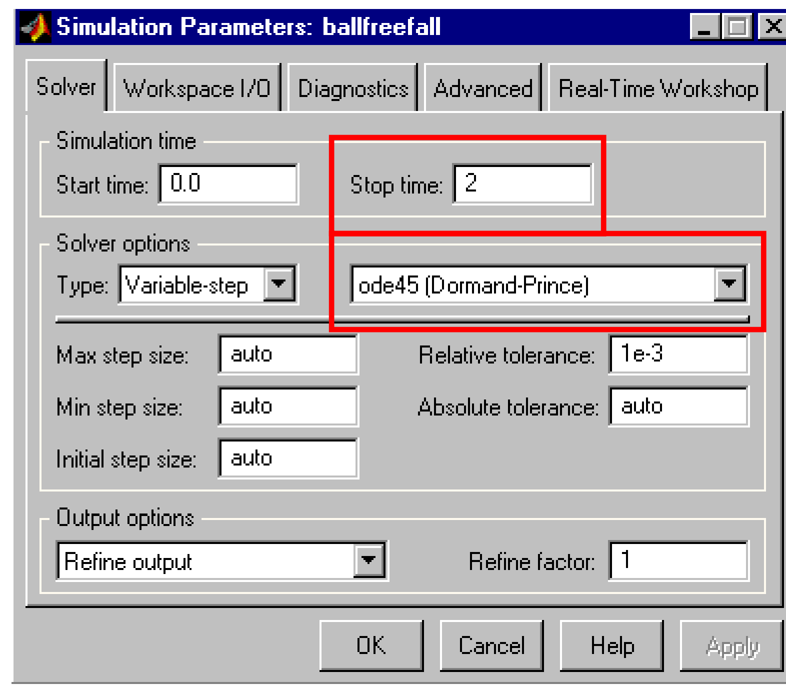


Initial velocity = 0

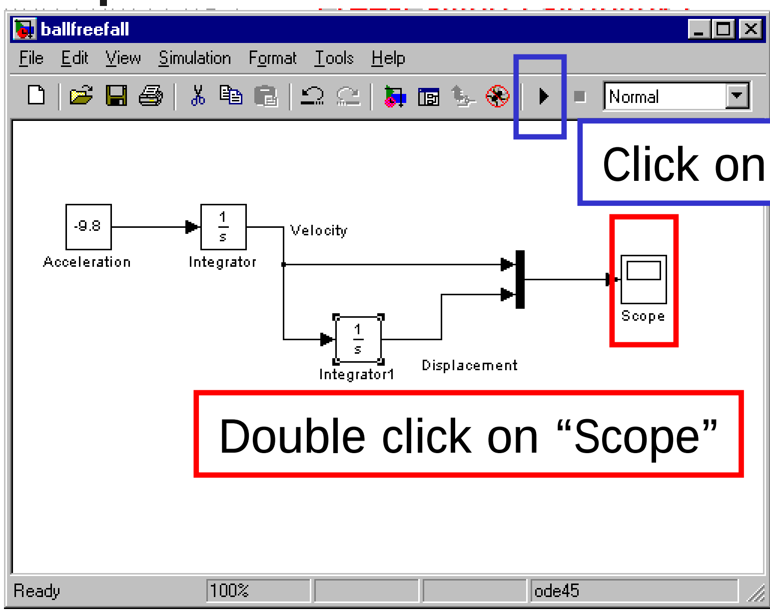
Initial displacement = 15

Time Span? ODE Function?





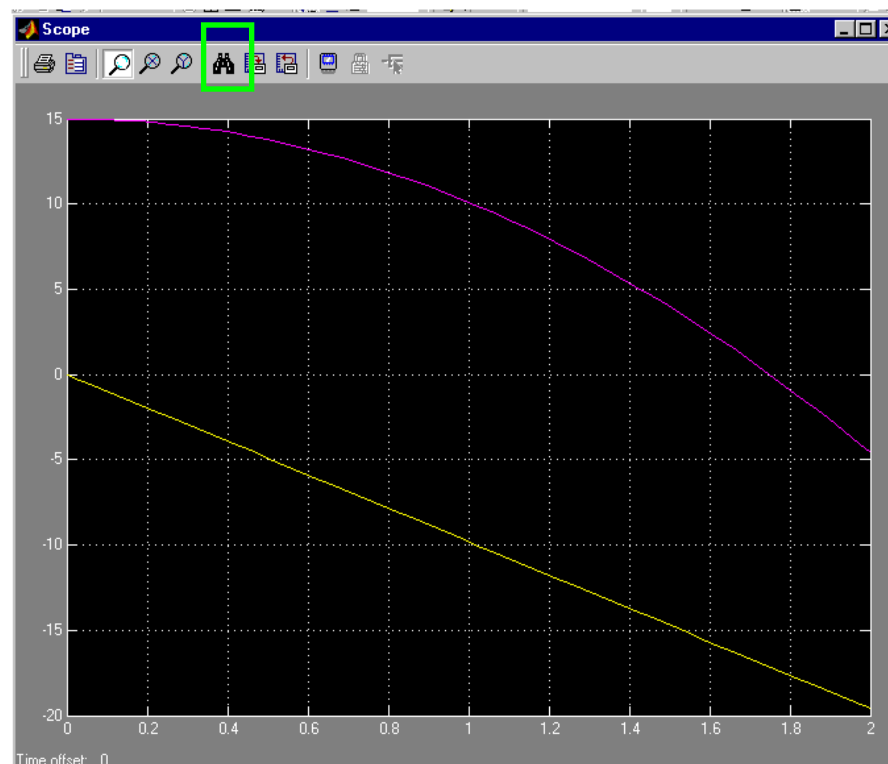
Run...



Click on “run”

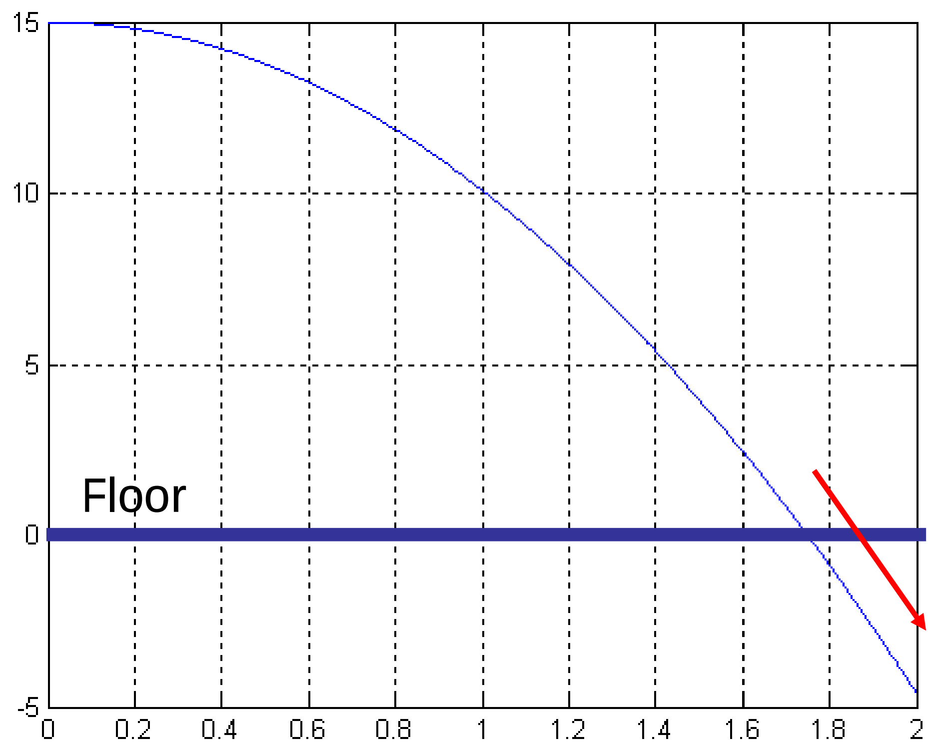
Double click on “Scope”

Click on “Autoscale” to fit in window



More Interesting Problem

- How if the ball hits the floor?



It passes through the floor...

It should bounce!

Hybrid System

- A **hybrid system** is a physical system that model using continuous components as well as discrete components.
  - A bouncing ball is a hybrid system since the evolution of its state variables vary continuously when falling,
  - But have a discrete change (velocity is reversed) when entering in collision with the ground

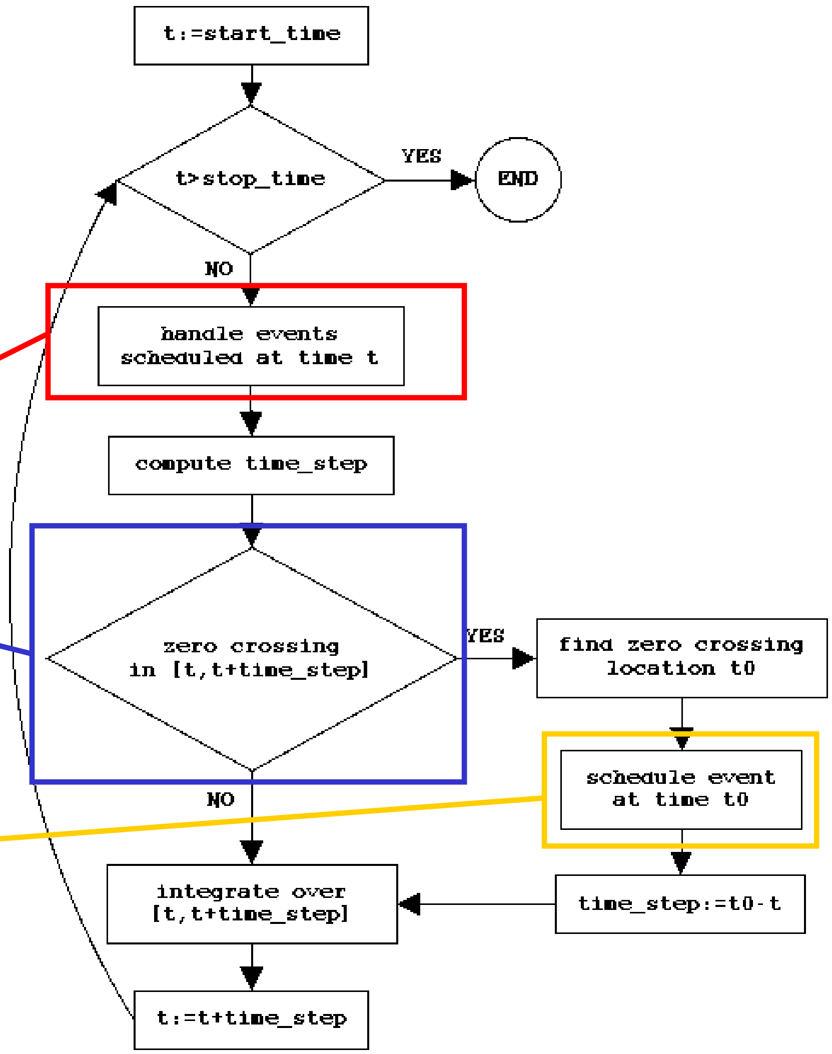
Solving Hybrid System

One way of solving...

Events Function

Zero Crossing

What's next?



Event Function in MATLAB

- Turn on event function in ode’s routine:
  - Use **odeset**
  - **OPTIONS = ODESET(‘Events’,@events)**

Turn on Event function (parameter)

Function returning the events (value)

  - **[T,Y] = ODE45(MYFUN,TSPAN,Y0,OPTIONS)**

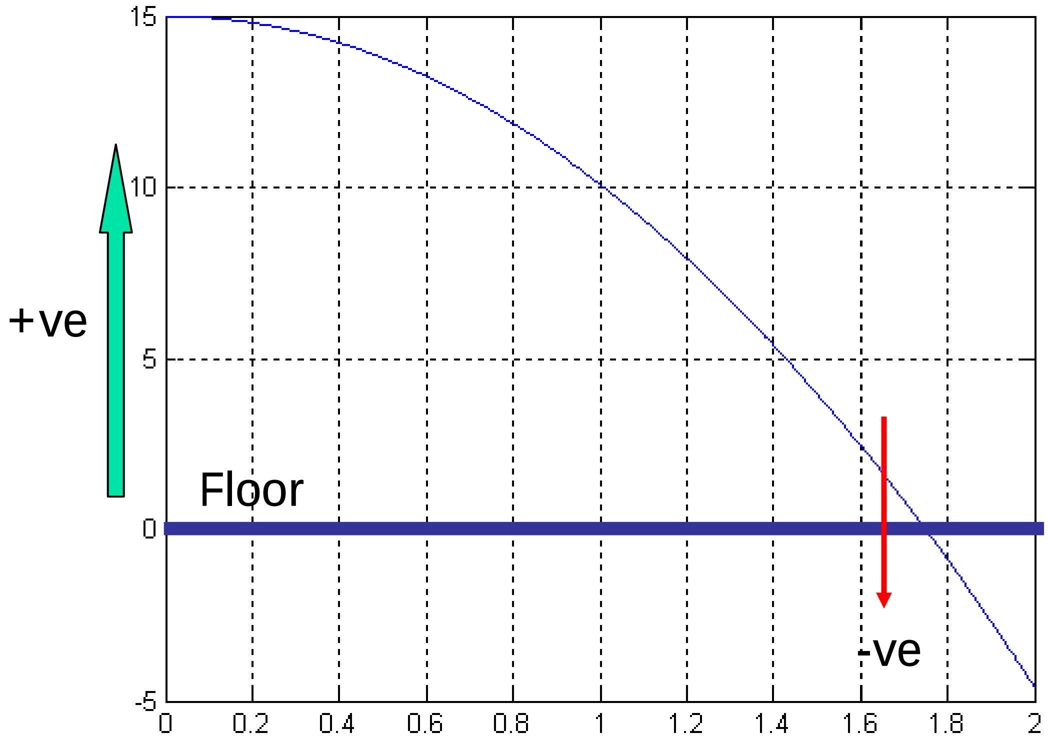
Event Function (Cont.)

```
[value,isterminal,direction] = events(t,y)
```

- Input arguments: time span and states
- Output arguments:
  - A value of the event function
  - The information whether or not the integration should stop when value = 0
    - 1: Stop the integration
    - 0: Continue the integration
  - The desired directionality of the zero crossings:
    - -1: Detect zero crossing in the negative direction only
    - 0: Detect all zero crossing
    - 1: Detect zero crossing in the positive direction only

Event Function for Bouncing

```
[value,isterminal,direction] = events(t,y)
```



$$\underset{\sim}{y} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

`value`: We are tracking the displacement of the ball (first state). The event will occur when y(1) = 0. So, return **`value = y(1)`**

`isterminal`: The integration Should stop when the ball hits the floor.
So, **`isterminal = 1`**

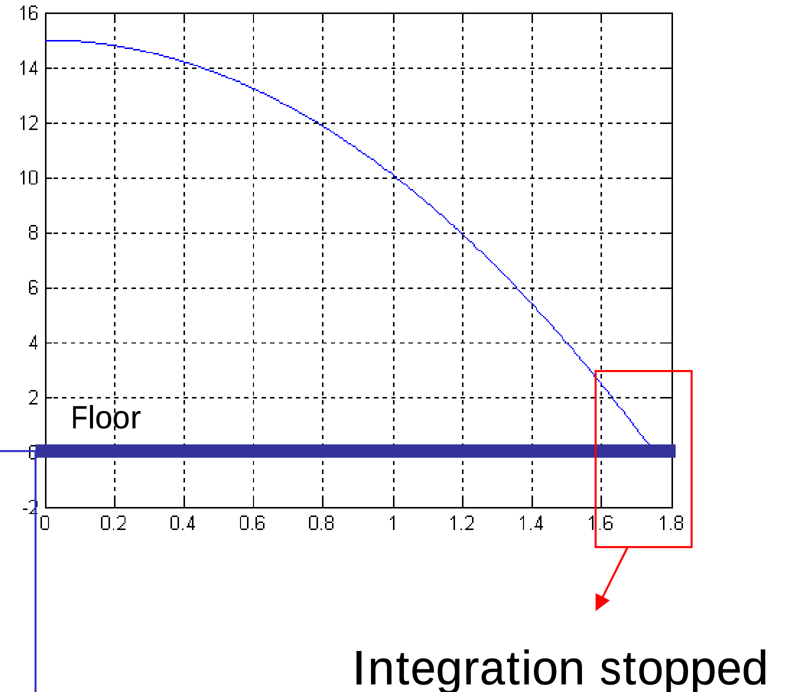
`direction`: The event should occur at the –ve direction.
So, **`direction = -1`**

Results

```
clear all
tspan=linspace(0,2,1001)';
y0=[15 0];
opt = odeset('Events',@events);
[t,y]=ode45(@ball,tspan,y0,opt);
plot(t,y(:,1))
grid on
```

```
function
   [value,isterminal,direction]=events(t,y)

value=y(1);
isterminal=1;
direction=-1;
```



- Ex02_BallFalling_Main.m
- Stop without bouncing

Multiple Bouncing

- How should we make it bounce?
  - [T,Y,TE,YE,IE]=ODE45(MYFUN,TSPAN,Y0,OPTION)
  - More information can be extracted from ode45 routine:
    - TE is a column vector of times at which events occur
    - Rows of YE are the solution values corresponding to times in TE
    - Indices in vector IE specify which event occurred at the time in TE
  - “For” loop can be implemented to have multiple bouncing.

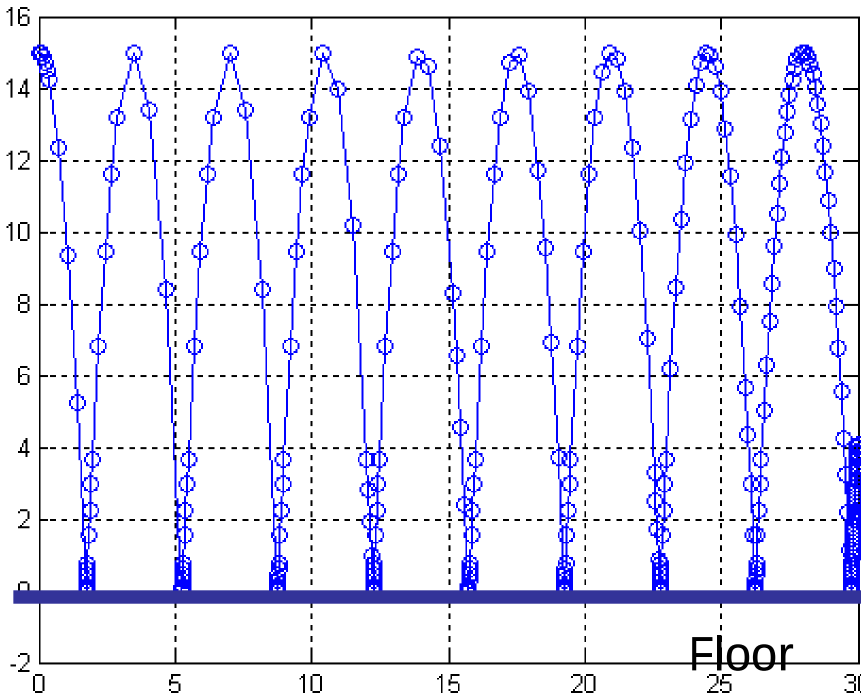
Multiple Bouncing (Cont.)

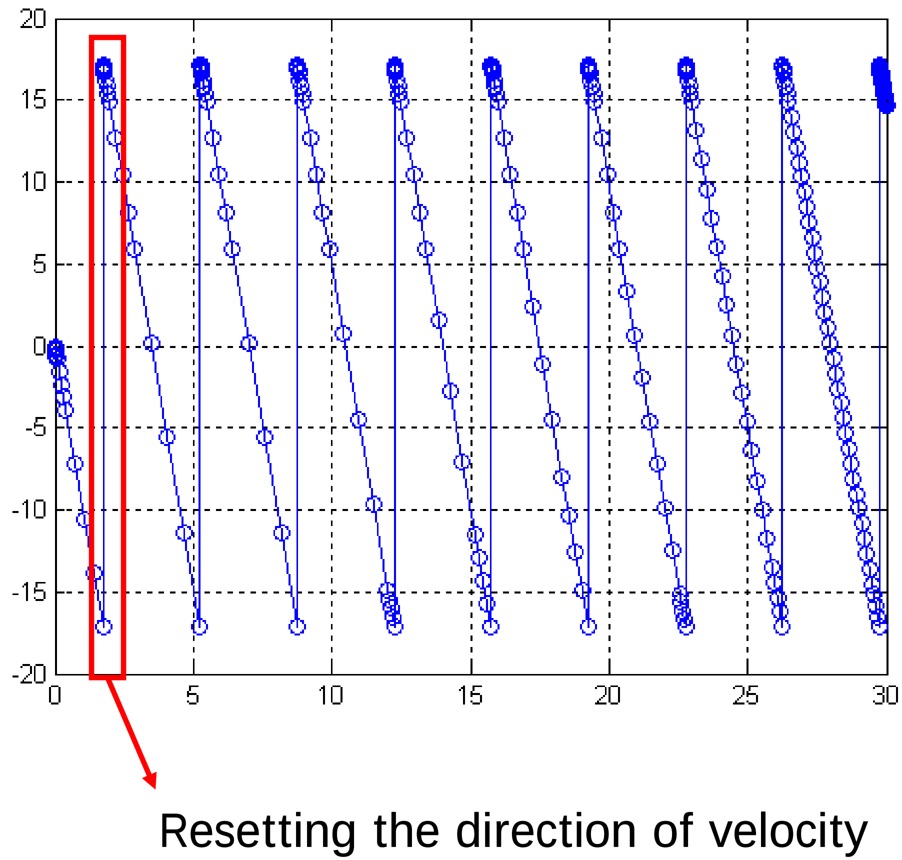
```
clear all
tstart = 0;
t_cum = tstart;
tfinal = 30;
y0 = [15 0];
y_cum = y0;
opt =
odeset('Events',@events);
```

```
for i = 1:10,
    [t,y,te,ye,ie]=ode45(@ball,[tstart
tfinal],y0,opt);
    m = length(t);
    t_cum = [t_cum;t];
    y_cum = [y_cum;y];
    tstart = te;
    y0 = [0 -y(m,2)];
end
plot(t_cum,y_cum(:,1),'-o')
grid on
```

The direction of the Velocity changed

Multiple Bouncing (Cont.)

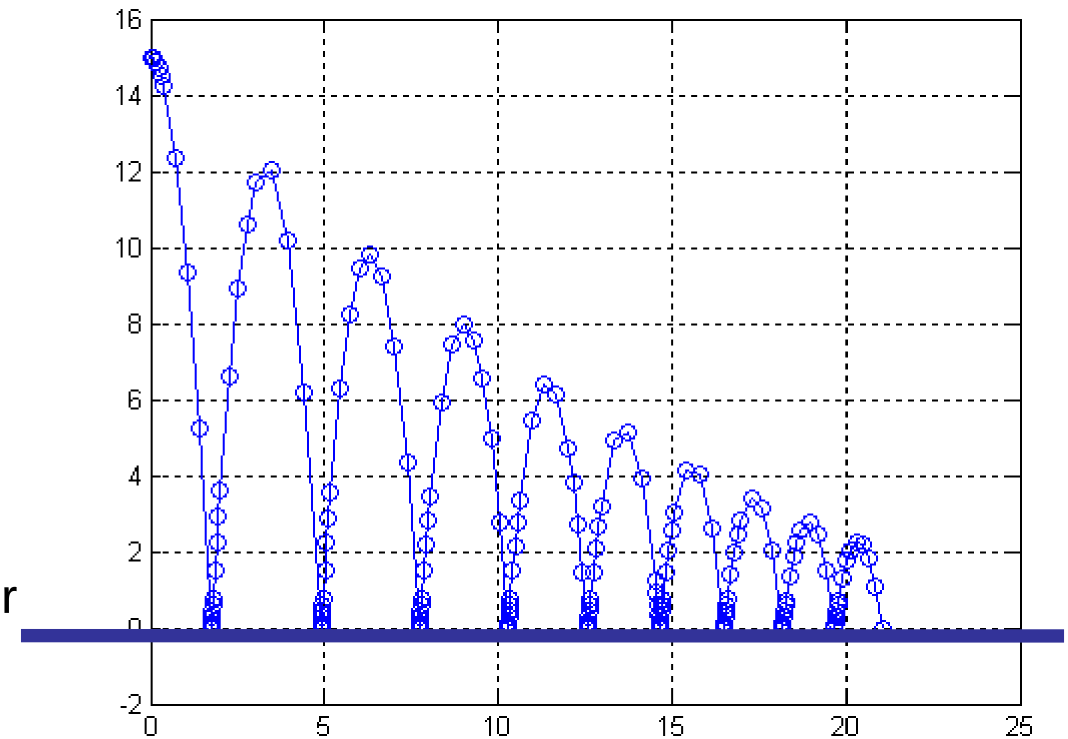




More Realistic Modeling

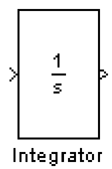
- If the ball bouncing is not elastic.
  - Reset the initial velocity at every initial conditions.

```
y0 = [0 -0.9*y(m,2)];
```

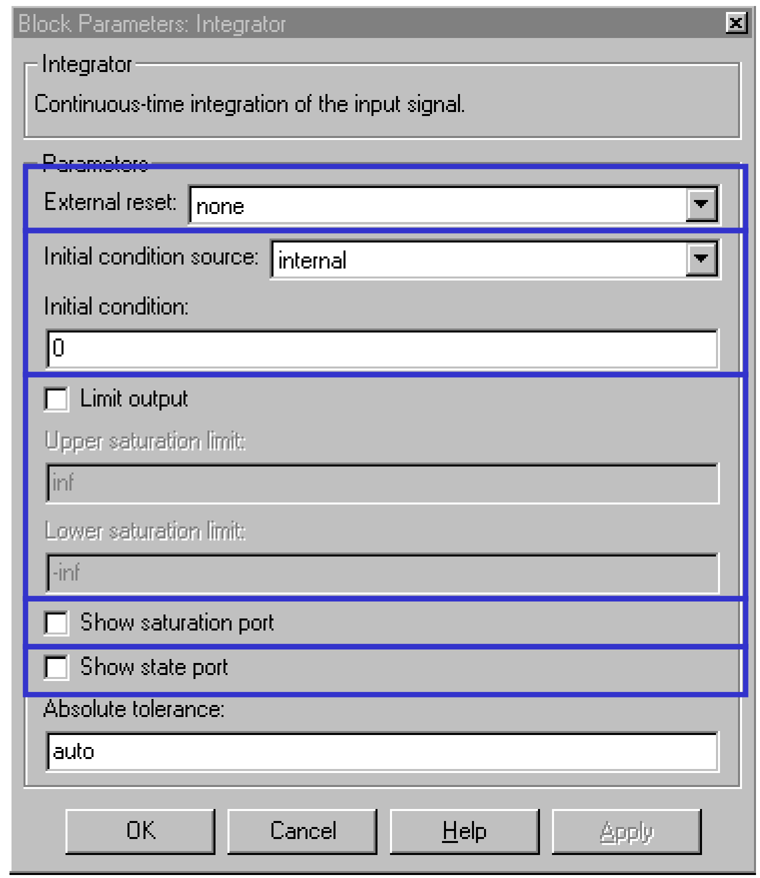


The Integrator in Simulink

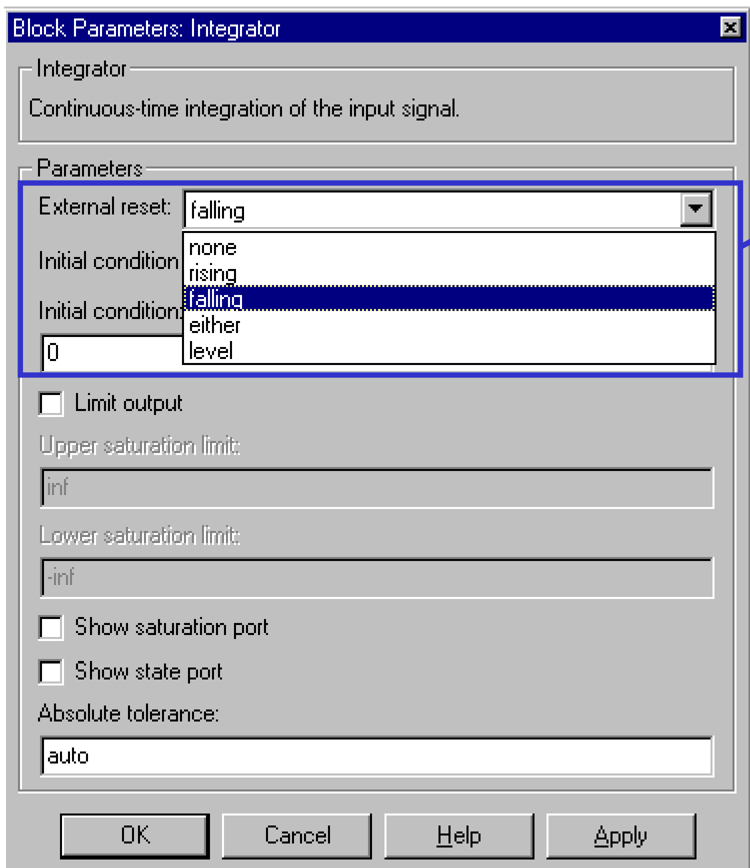
- Let's look at the integrator block closer.

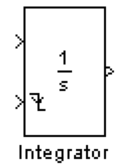


Double-click



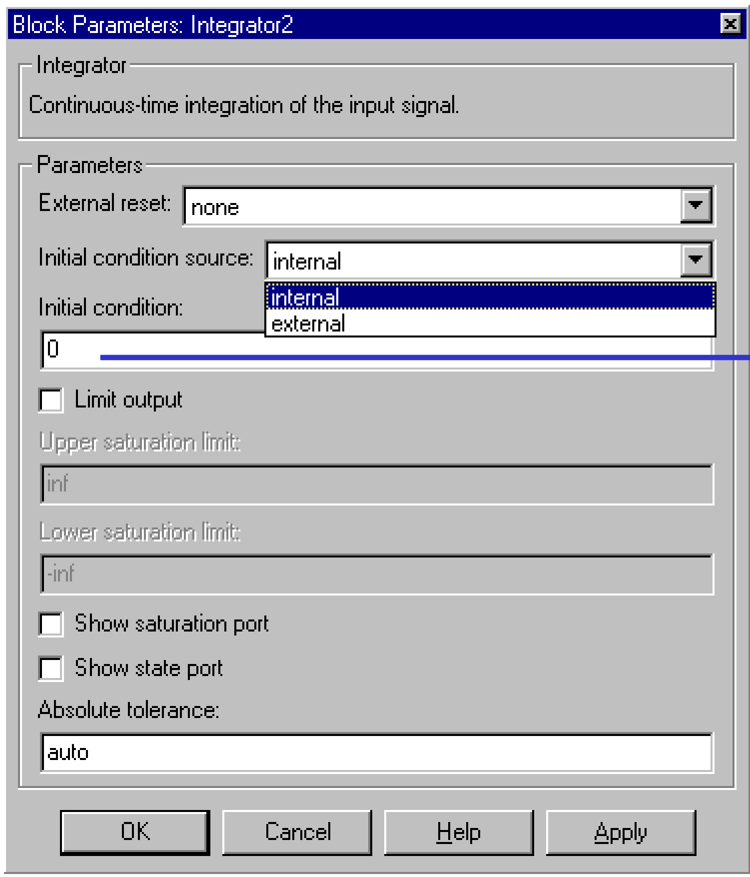
Zero Crossing





Rising: reset to IC when rising edge is detected
Falling: reset to IC when falling edge is detected
Either: reset to IC either rising of falling edge is detected
Level: reset and hold the output to IC while the reset signal is not zero

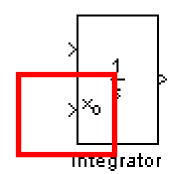
Initial Conditions



The IC can be provided internally or externally.

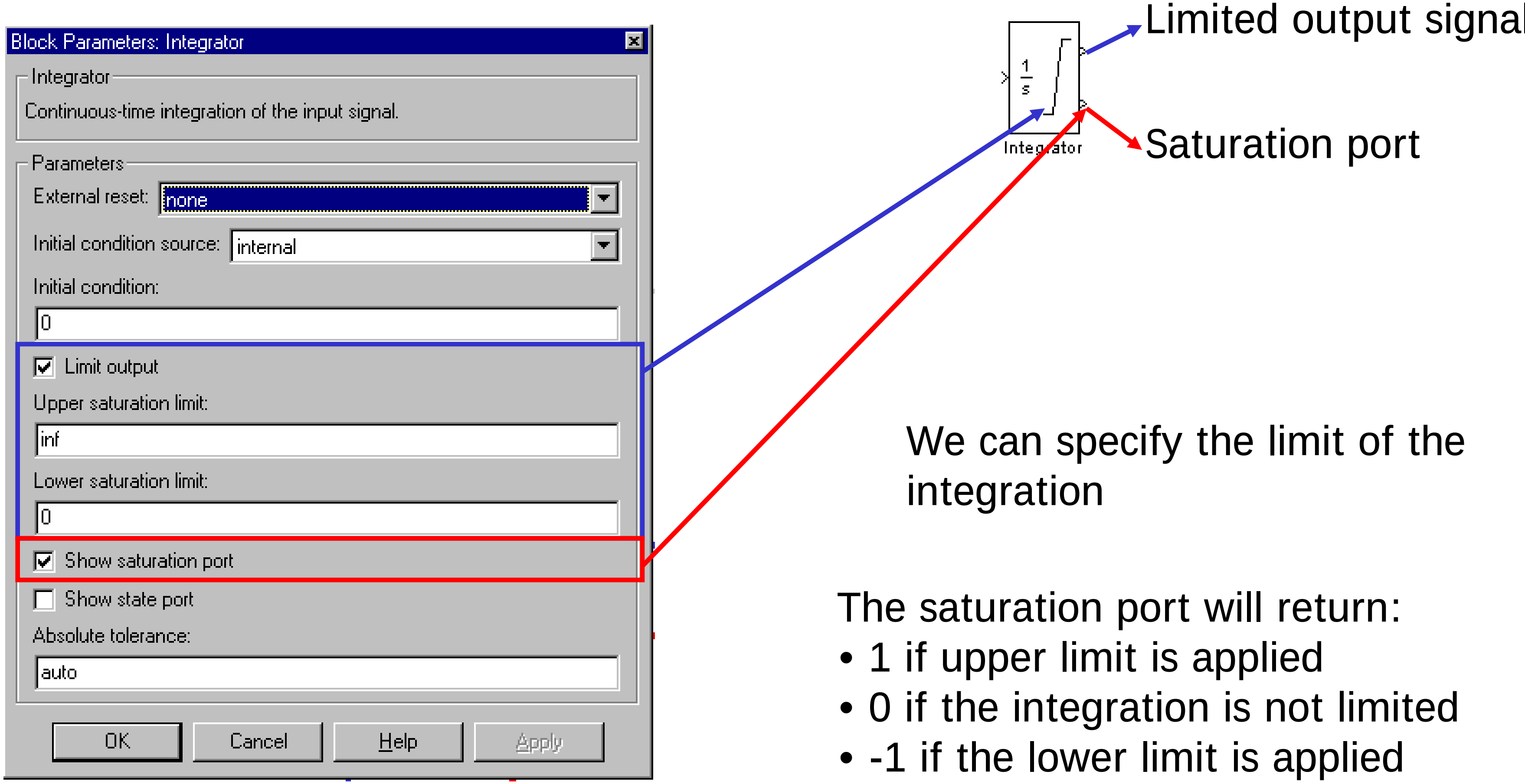
If “internal” is selected, then provide a fixed IC at the space provided.

If “external” is selected, then an IC port will be shown:



So that you can provide IC externally.

Output Limit

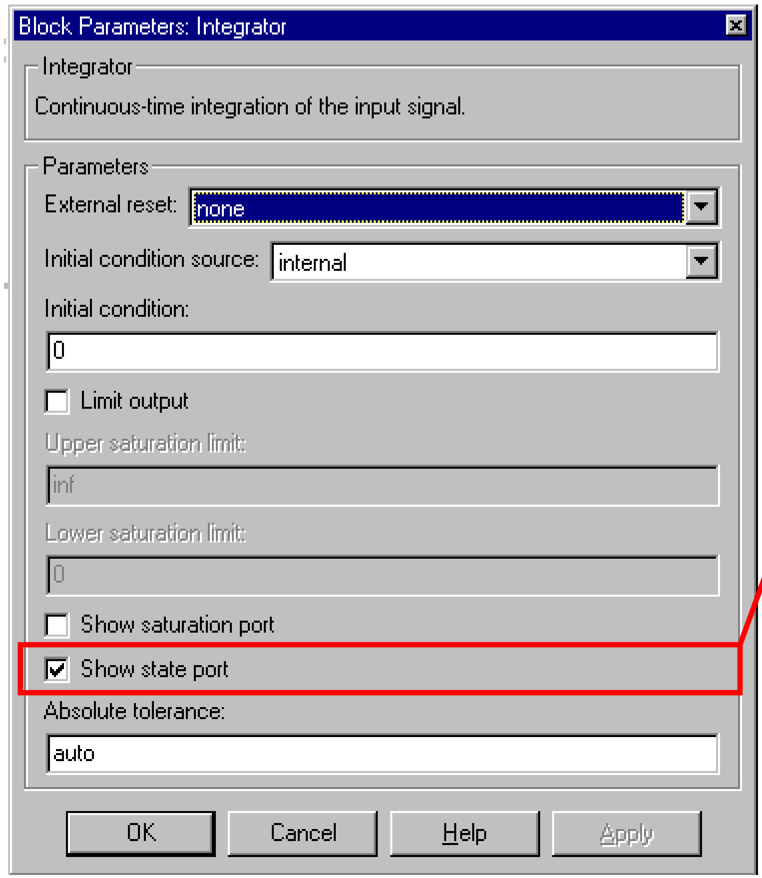


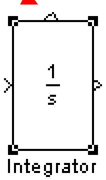
We can specify the limit of the integration

The saturation port will return:

- 1 if upper limit is applied
- 0 if the integration is not limited
- -1 if the lower limit is applied

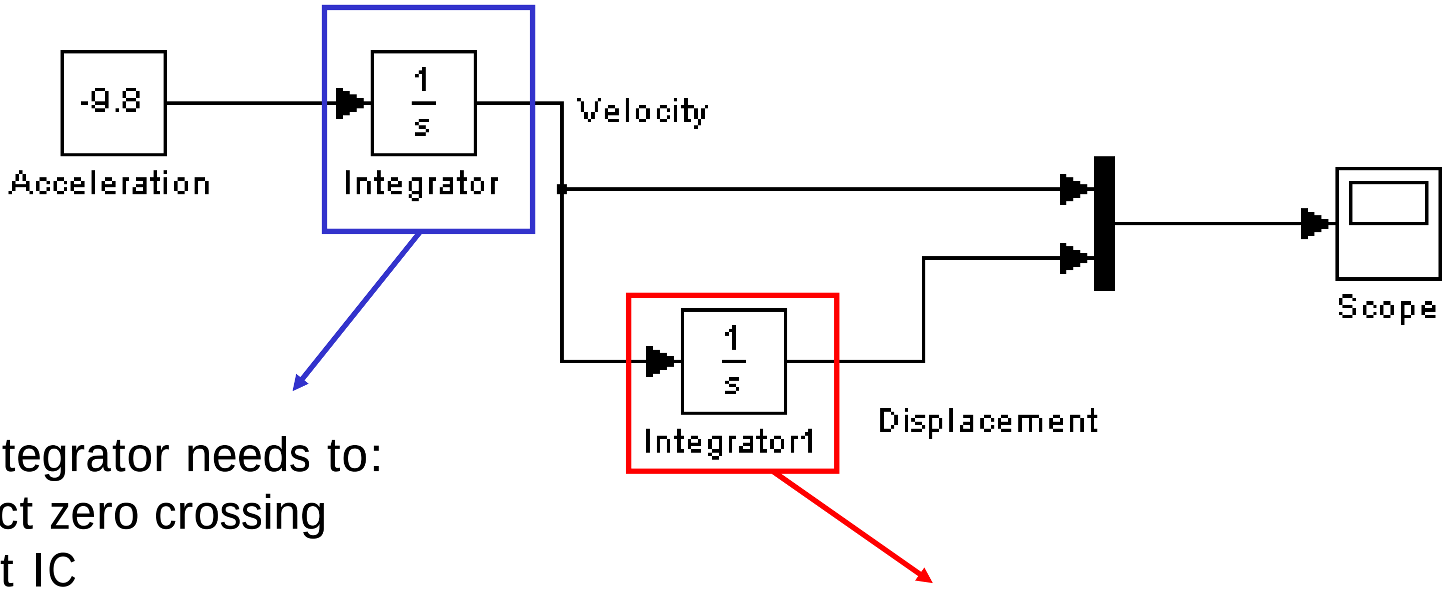
State Port





State port allow us to feedback the state signal.
This is very useful when resetting IC.

Recall No Bouncing Situation

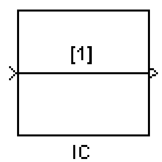


This integrator needs to:
- Detect zero crossing
- Reset IC

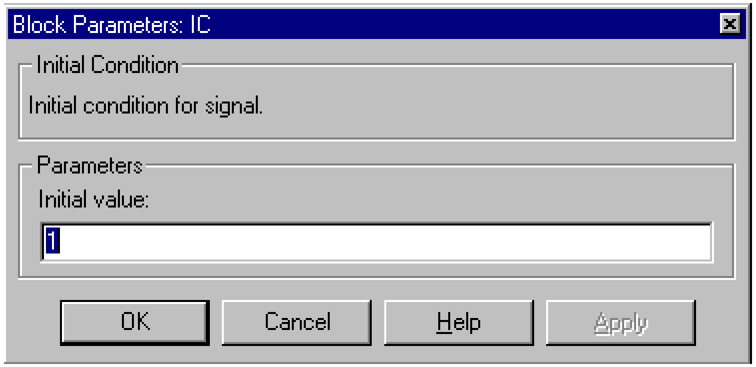
This integrator needs to:
- Limit the integration

IC Block

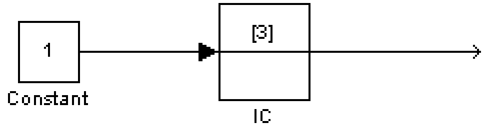
One more useful block before we go on...



Double-click

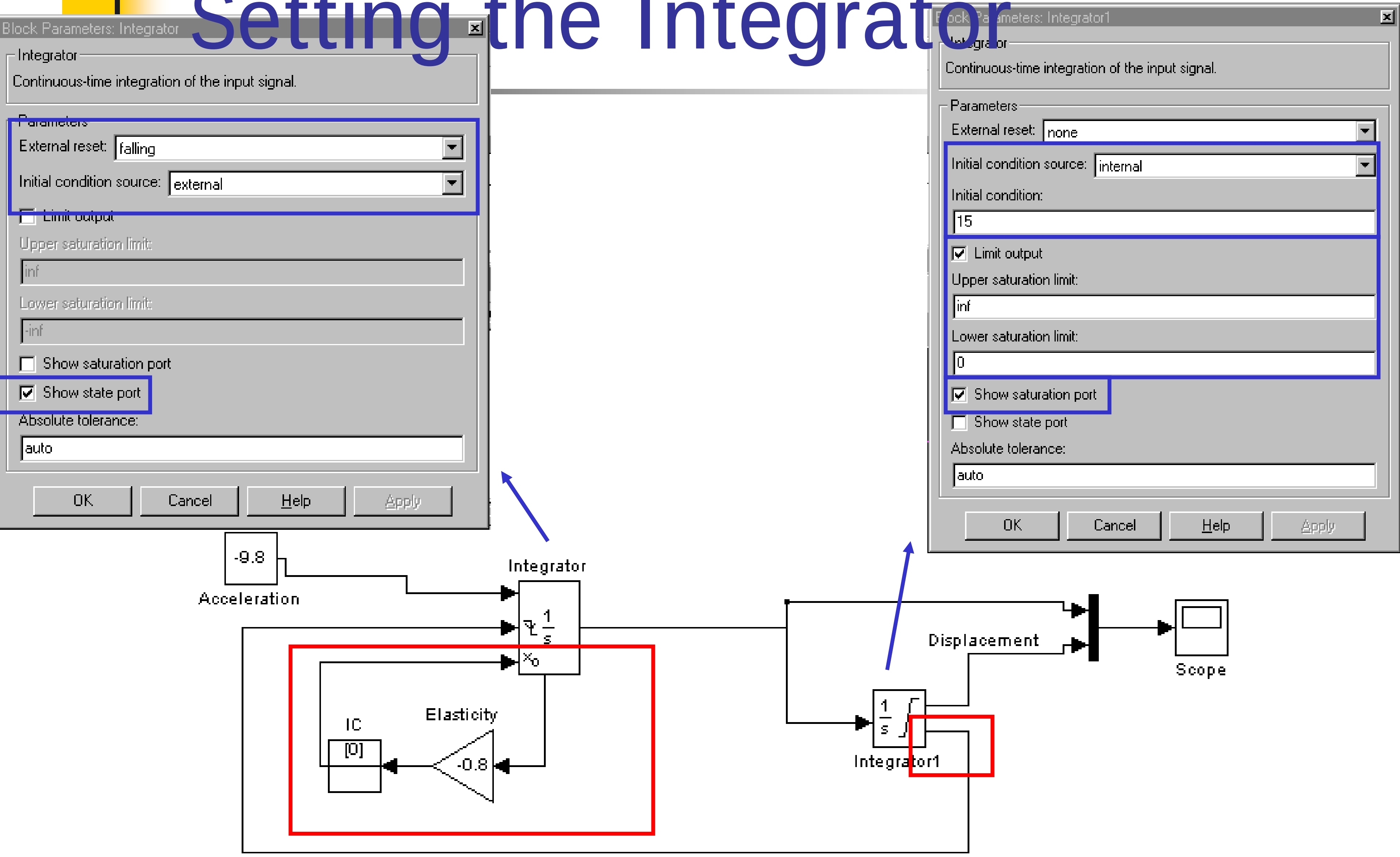


Example:

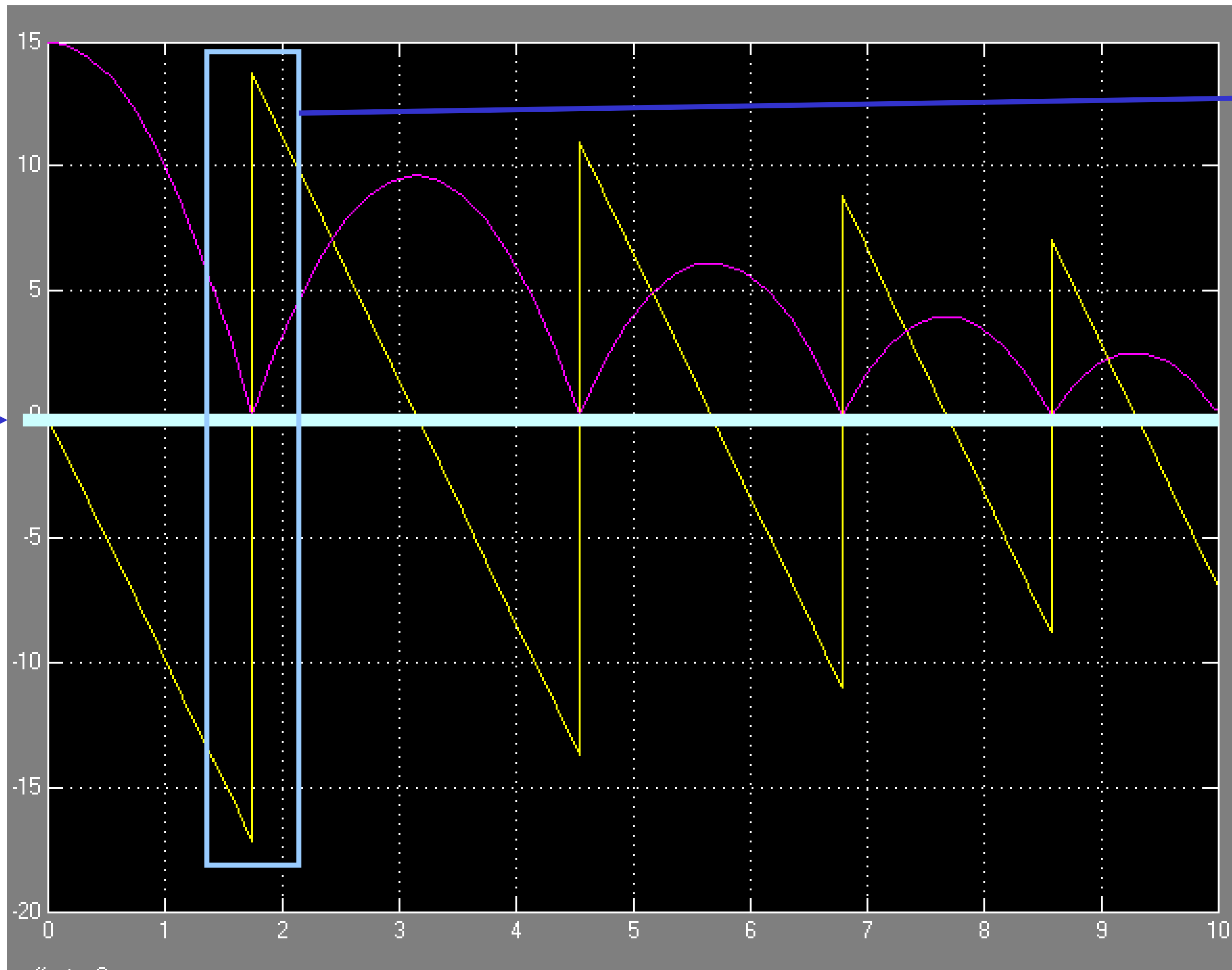


At t=0, the initial value is 3. Afterwards, the signal value is 1.

Setting the Integrator



Results



Velocity direction change

Floor

Conclusion

- Issues that are not discussed here:
  - Step sizes of the numerical integration.
  - Complexity of mixture of continuous and discrete system.
  - The modeling of the events. (Function? If... Then...?)
- Simulink vs. MATLAB M-files.
  - Which is better?
  - Visualization: Simulink might be better.
  - Some specific routines like Optimization, we might still need MATLAB.
  - But, most of the routines can be done in both ways.

Thank You!

Questions & Suggestions?