MATLAB 科学计算语言与应用

Lecture 5: Various functions and toolboxes

Outline

- **Symbolic Toolbox**
- Simulink
- Image Processing
- Miscellaneous Useful Functions
- Graphical User Interfaces

Miscellaneous Matlab (1)

- The command deal can make variable initialization simpler

```
»[x, y, z] = deal(zeros(20, 30));
»[a, b, c, d] = deal(5);
»[m, n] = deal(1, 100);
```

- The command eval can execute a string!

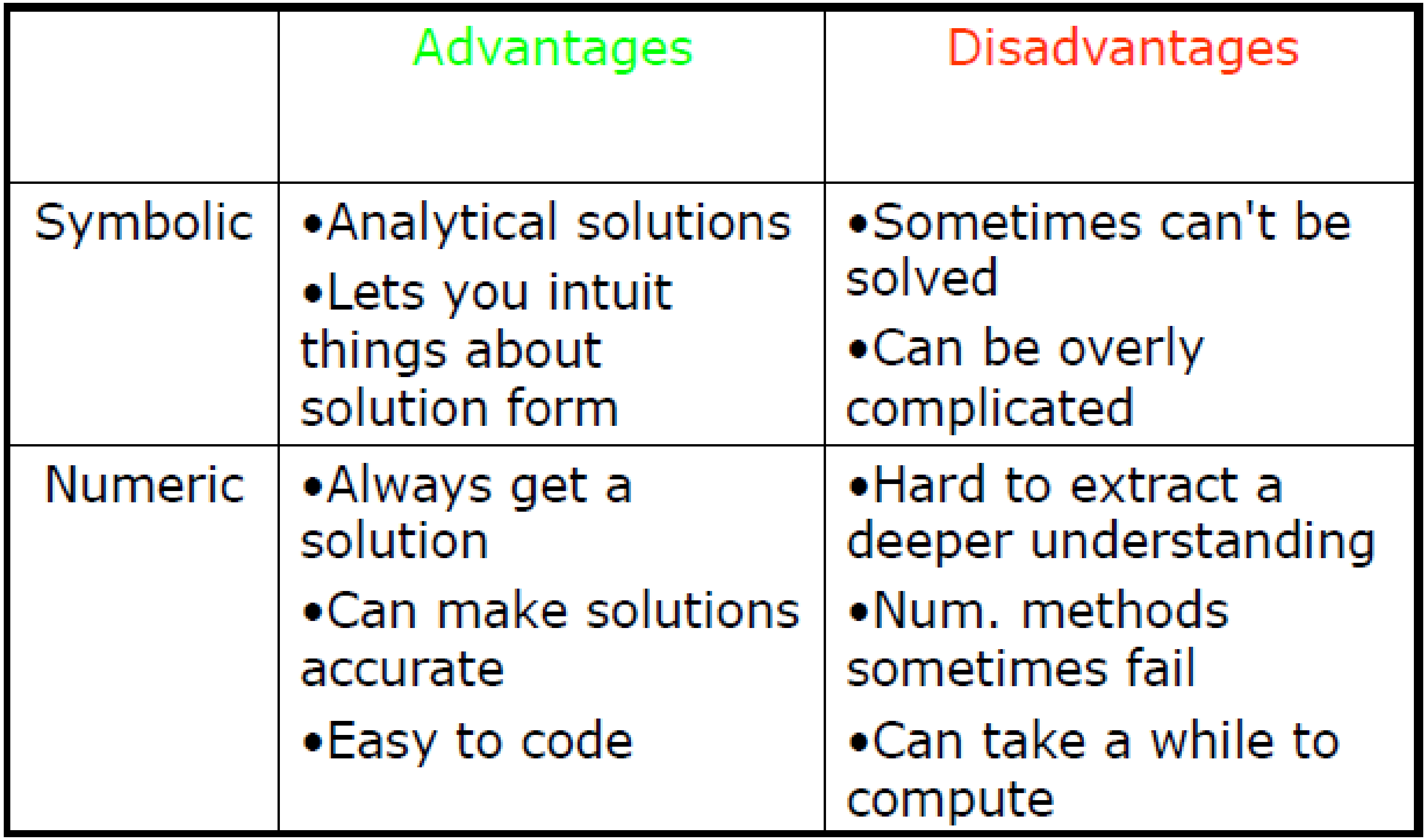
```
»a1 = 1; n = 1;
»eval([ ‘a’  num2str(n)  ‘ = 5;’  ]);
»disp([ ‘a1 is now  ’ num2str(a1)]);
```

- The command repmat can create replicas easily

```
»A = repmat([1 2;3 4], 2, 2);
```

Symbolic Toolbox

- Don’t do nasty calculations by hand!
- Symbolics vs. Numerics



	Advantages	Disadvantages
Symbolic	•Analytical solutions •Lets you intuit things about solution form	•Sometimes can't be solved •Can be overly complicated
Numeric	•Always get a solution •Can make solutions accurate •Easy to code	•Hard to extract a deeper understanding •Num. methods sometimes fail •Can take a while to compute

Symbolic Variables

- Symbolic variables are a type, like **double** or **char**
- To make symbolic variables, use **sym**

```
»a=sym('1/3'); %fractions remain as fractions
»b=sym('4/5');
»mat=sym([1 2;3 4]);
```

- can add tags to narrow down scope

```
»c=sym('c','positive');
```

  see **help sym** for a list of tags

- Or use **syms**

```
»syms x y real
```

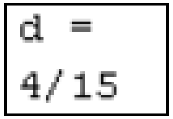
  - shorthand for x=sym('x','real'); y=sym('y','real');

Symbolic Expressions

- Multiply, add, divide expressions

```
» d=a*b
```

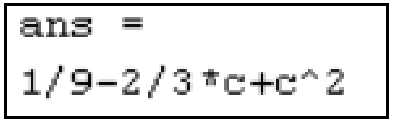
  - does 1/3*4/5=4/15;



```
d =
4/15
```

```
» expand((a-c)^2);
```

  - multiplies out



```
ans =
1/9-2/3*c+c^2
```

```
» factor(ans)
```

  - factors the expression

```
ans =
[ 1/9, 3*c - 1, 3*c - 1]
```

```
» pretty(ans)
```

  - makes it look nicer

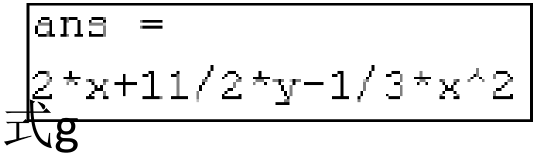
```
 / 1                    \
 |  -, 3 c - 1, 3 c - 1 |
 \ 9                    /
```

```
 2   2 c   1
c  - --- + -
      3    9
```

Cleaning up Symbolic Statements

```
» collect(3*x+4*y-1/3*x^2-x+3/2*y)
```

- collects terms



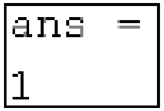
```
ans =

2*x+11/2*y-1/3*x^2
```

collect(g, v)：按指定的表达式或变量v整理表达式g

```
» simplify(cos(x)^2+sin(x)^2)
```

- simplifies expressions

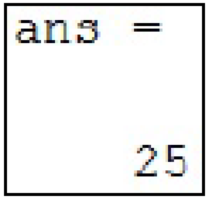


```
ans =

1
```

```
» subs(c^2 , c , 5)
```

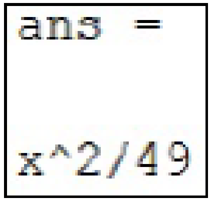
- replaces variables with numbers or expressions. To do multiple substitutions pass a cell of variable names followed by a cell of values



```
ans =

    25
```

```
» subs(c^2 , c , x/7)
```

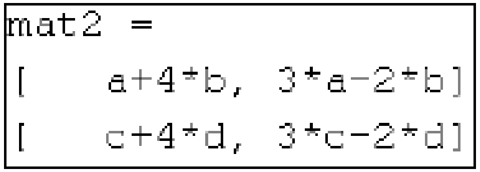


```
ans =

x^2/49
```

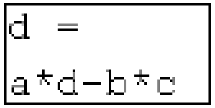
More Symbolic Operations

- We can do symbolics with matrices too
  - » mat = sym([a b; c d])
  - » `mat=sym('A%d%d', [2 2]);`
    - ➢ symbolic matrix of specified size
  - » `mat2=mat*[1 3;4 -2];`



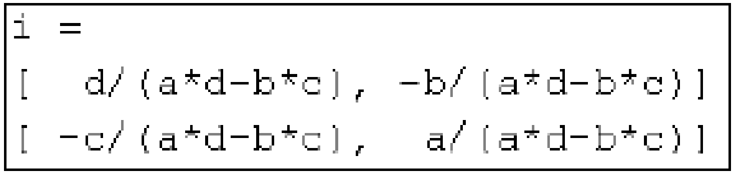
```
mat2 =
[   a+4*b, 3*a-2*b]
[   c+4*d, 3*c-2*d]
```

    - ➢ compute the product
  - » `d=det(mat)`



```
d =
a*d-b*c
```

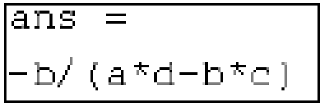
    - ➢ compute the determinant
  - » `i=inv(mat)`



```
i =
[  d/(a*d-b*c), -b/(a*d-b*c)]
[ -c/(a*d-b*c),  a/(a*d-b*c)]
```

    - ➢ find the inverse

- You can access symbolic matrix elements as before
  - » `i(1,2)`



```
ans =
-b/(a*d-b*c)
```

- **求符号根**

```
>> syms x a b c;
>> m=a*x^2+b*x+c;
>> n=[a b c];
>> roots(n)
ans =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

- **求反函数**

g=finverse(f, v) %对指定变量v（默认为x）求反函数

```
>> syms t x a b;
>> f1=b*exp(-t+a*x);
>> g1=finverse(f1,t)
g1 =
a*x - log(t/b)
 >> g2=finverse(f1)
 g2 =
(t + log(x/b))/a
```

• 求复合函数

compose(f, g)   %用g代替f中x的位置

compose(f, g, t)   %用g代替f中t的位置

compose(f, g, x, z)   %用g(z)代替f(x)中x的位置

compose(f, g, t, u, z)   %用g(z)代替f(t,u)中的t，用z代替u

• 求函数极限

limit(f, x, a)   %相当于$\lim\limits_{x\to a} f((x)$

limit(f, x, a, ‘right’)   % x右趋近于a

limit(f, x, a, ‘left’)   % x左趋近于a

• 级数求和

symsum(s,x,a,b)   %求s当x从a到b的级数和

例：

```
>> syms n k;
>> symsum(n,0,k-1)
 ans =
 (k*(k - 1))/2
```

- **符号微分（符号函数求导）**

```
diff(f, x, n)   %计算f对x的n阶导数
diff(f, x, y, z）%f先对x求偏导，再对y求偏导，再对z求偏导
```

- **符号积分**

```
int(S, v, a, b)  %求函数S对变量v在[a, b]区间上的定积分
int(S, v) %求函数S对变量v的不定积分
```

例：求定积分$\int_0^{\infty}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\mathrm{dx}$

```
>> syms x;
>> f=1/sqrt(2*pi)*exp(-x^2/2);
>> int(f,0,inf)
ans =
 (7186705221432913*2^(1/2)*pi^(1/2))/36028797018963968
```

符号方程求解: solve

- The equation of a circle of radius r centered at (a,b) is given by: $(x-a)^2+(y-b)^2=r^2$
- Use `solve` to solve this equation for x and then for y

```
» syms a b r x y
» solve( (x-a)^2+(y-b)^2==r^2, x)
» solve( (x-a)^2+(y-b)^2==r^2, y)
```

- It's always annoying to integrate by parts. Use `int` to do the following integral symbolically and then compute the value by `subs`tituting 0 for a and 2 for b: $$\int_a^b xe^x dx$$

```
» Q=int(x*exp(x), a, b)
» subs(Q,{a,b},{0,2})
```

符号常微分方程求解：dsolve

S=dsolve(equation, condition, ‘x’)

微分方程在指定条件下对指定自变量x进行求解

equation和condition，都需要是符号表达式，并且用“==”表示等号

例：求解$y'' = -a^2 y$，初始条件$y(0) = 1, y'(\frac{\pi}{a})$=0

```
>> syms y(t) a;      %定义y是t的函数，a是符号变量
>> Dy=diff(y);        %定义Dy是y的一阶导数
>> D2y=diff(y,2)     %定义D2y是y的二阶导数
>> yt=dsolve(D2y==-a^2*y, y(0)==1, Dy(pi/a)==0)
```

符号常微分方程组求解：dsolve

[Sv1, Sv2, …]=dsolve(eqn1,eqn2,…, cond1, cond2,…, ‘x1’, ‘x2’,….)

微分方程组在指定条件下对指定自变量进行求解

equation和condition，都需要是符号表达式，并且用“==”表示等号

例：求解下列微分方程组，已知初始条件：$f(0)=1, g(0)=2$

$$\begin{cases} f'(t)=f(t)+g(t) \\ g'(t)=g(t)-f(t) \end{cases}$$

```
>> syms f(t) g(t)
>> Df=diff(f)
>> Dg=diff(g)
>> [sf,sg]=dsolve(Df==f+g,Dg==g-f,f(0)==1,g(0)==2)
```

符号函数可视化：绘图函数前面+ez

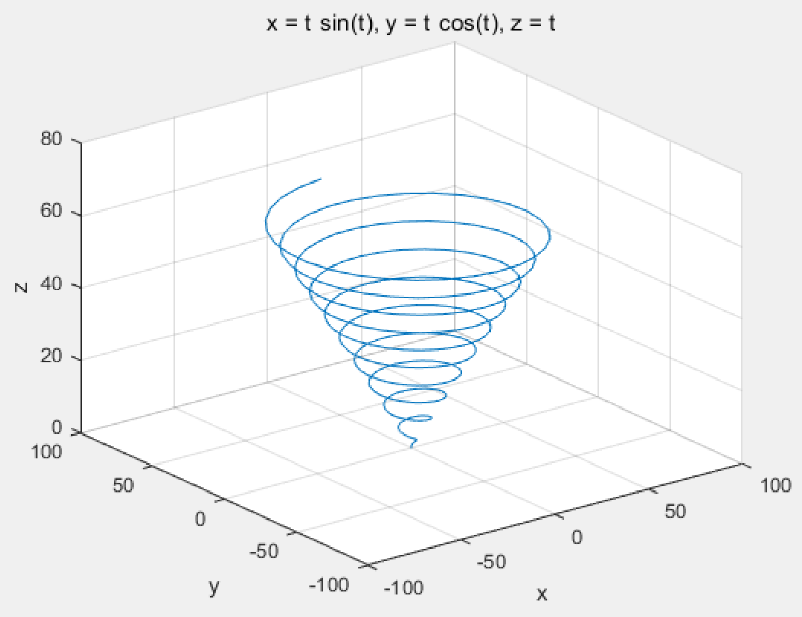
- ezplot, ezplot3, ezpolar, ezsurf, ezmesh, ezcontour,……..

例：绘制以下方程表示的三维图形，t的范围为[0, 20π]

$$\begin{cases} x = tsin(t) \\ y = tcos(t) \\ z = t \end{cases}$$

```
>> syms t;
>> ezplot3(t*sin(t),t*cos(t),t,[0 20*pi])


>> t=0:20*pi;
>> plot3(t.*sin(t),t.*cos(t),t)
```



例：绘制函数

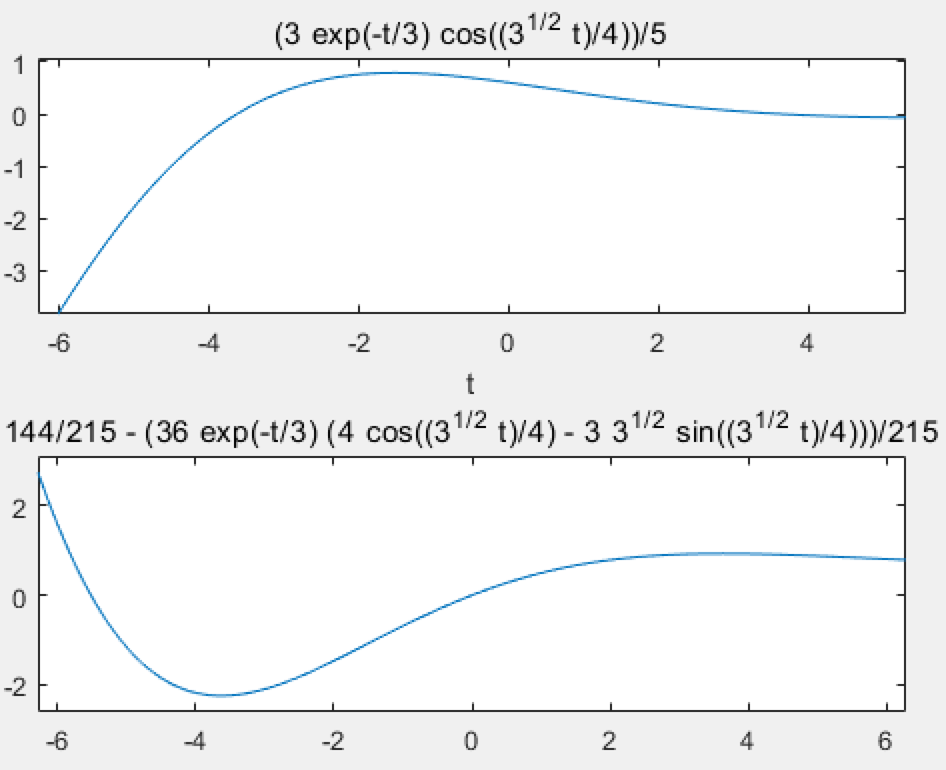
$$y(t) = 0.6e^{-\frac{t}{3}}cos\frac{\sqrt{3}}{4}t,$$

及其积分上限函数

$$s(t) = \int_0^t y(t)dt$$

的图形。

```
>> syms t;
>> y=0.6*exp(-t/3)*cos(sqrt(3)/4*t)
>> subplot(2,1,1);ezplot(y);
>> syms tao;
>> s=subs(int(y,t,0,tao),tao,t)
>> subplot(2,1,2);ezplot(s);
```



Outline

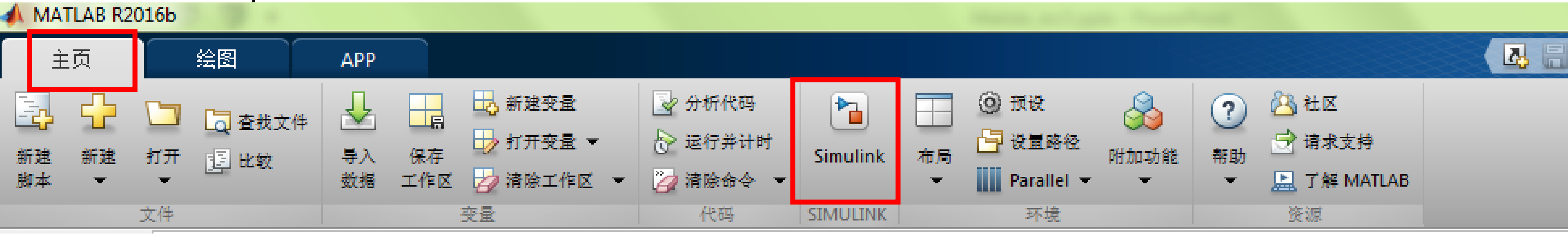
- Symbolic Toolbox
- **Simulink**
- Image Processing
- Miscellaneous Useful Functions
- Graphical User Interfaces

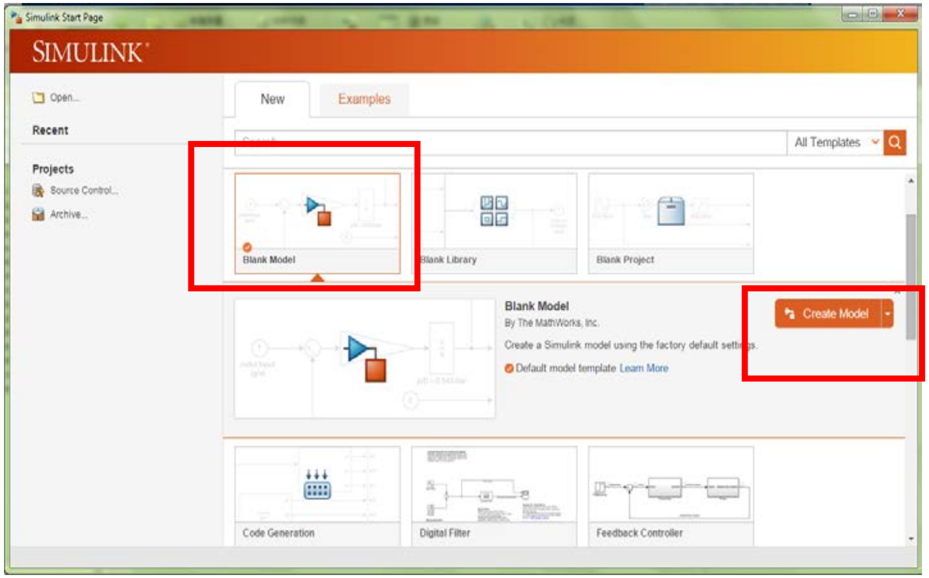
SIMULINK

- Interactive graphical environment
- Block diagram based MATLAB add-on environment
- Design, simulate, implement, and test control, signal processing, communications, and other time-varying systems

Getting Started

- In MATLAB, Start Simulink



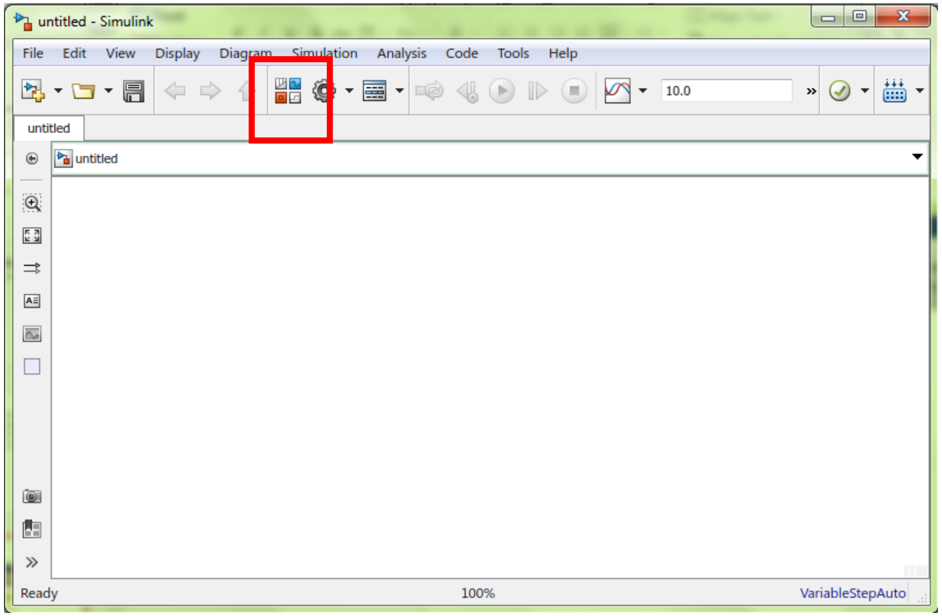


Blank Model

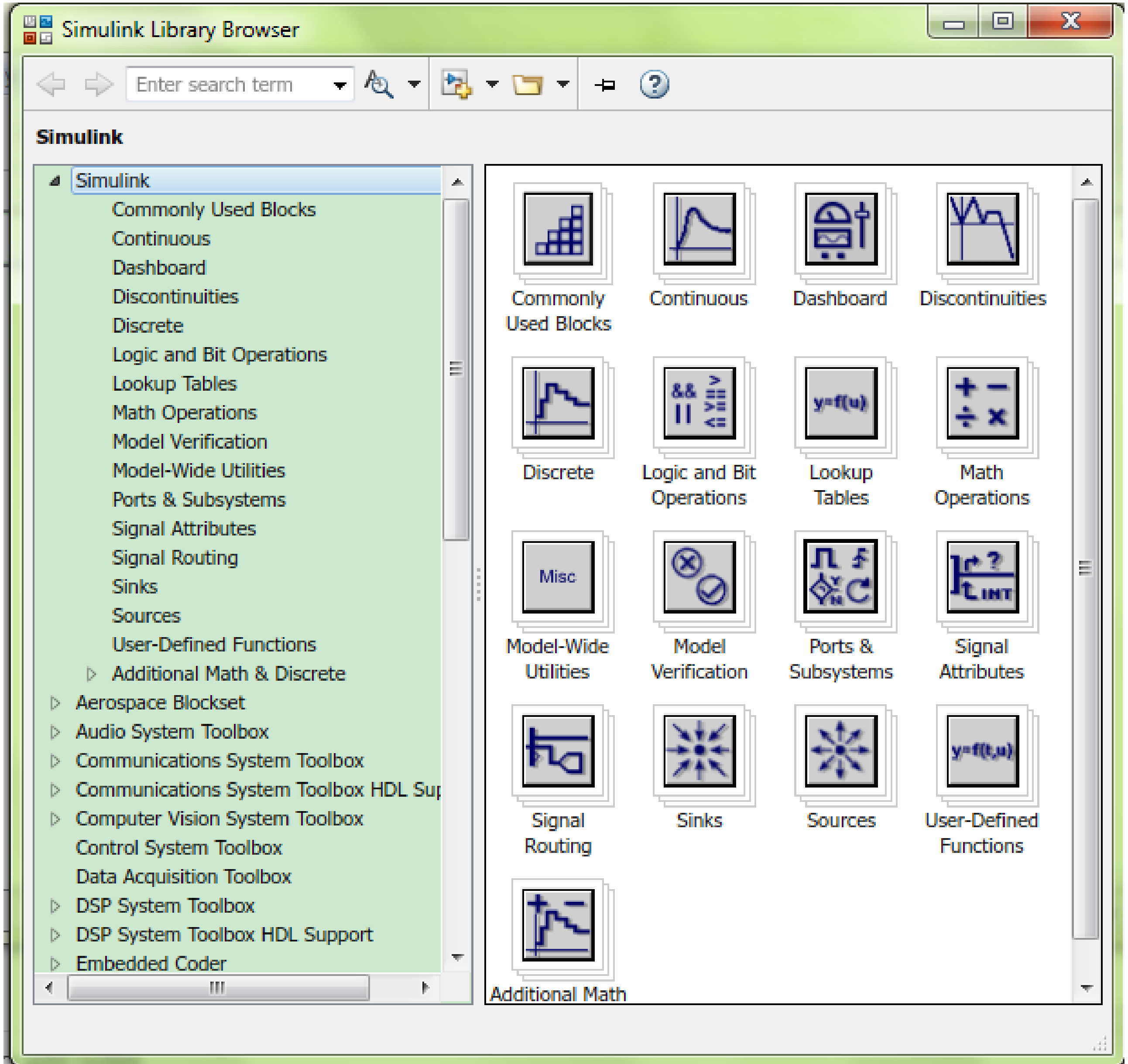
By The MathWorks, Inc.

Create a Simulink model using the factory default settings.

Default model template Learn More



Simulink LibraryBrowser



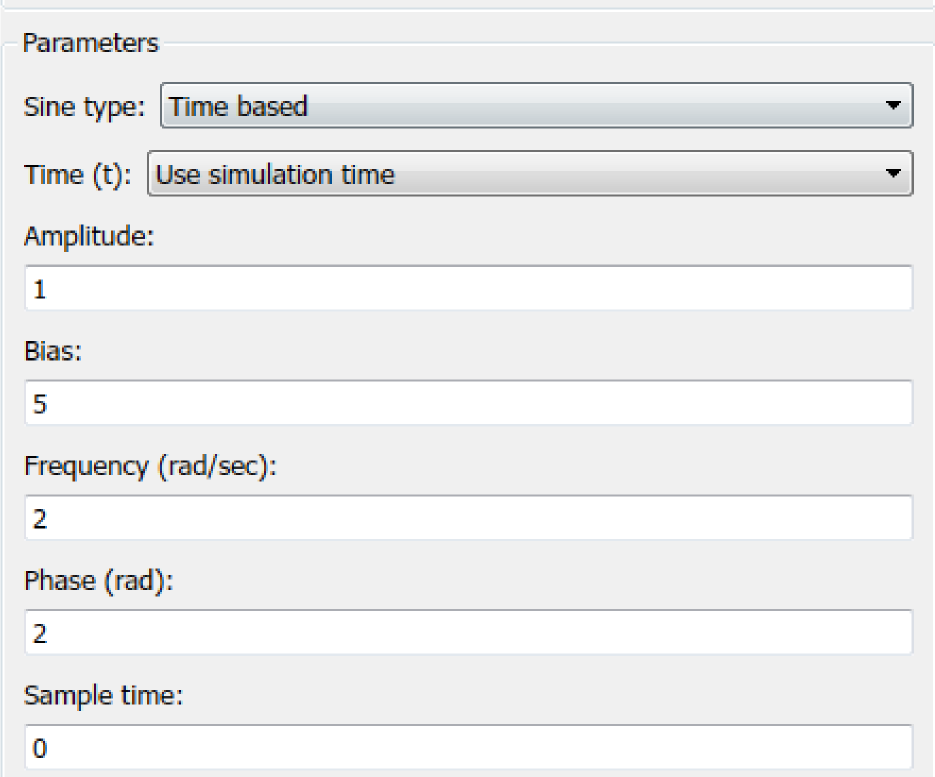
- Simulink
  - Commonly Used Blocks
  - Continuous
  - Dashboard
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
 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

例：将一个正弦信号输出到示波器

1. 打开模块库 ，查找正弦波模块sine
2. 选择0输入单输出的sine wave，拖入
3. 查找示波器模块scope，拖入
4. 连接sine wave的输出和示波器的输入
5. 单击 ▶ 进行系统仿真
6. 双击示波器看结果

改参数试试：双击sine wave模块

改完参数重新仿真，看结果



Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 1

Bias: 5

Frequency (rad/sec): 2

Phase (rad): 2

Sample time: 0

- **步骤总结**：

1. 从模块库中选择合适的模块（信源、信宿、处理系统）；
2. 按照实际系统的控制逻辑连接起来；
3. 设置模块参数（双击模块）和仿真参数（ “Simulation” ->” Configuration Parameters” ）；
4. 仿真调试

Simulink 模块库：

- 基本模块(basic building blocks)
- 各种应用工具箱(toolboxes)

Basic Building Blocks

- **Sources(信号源）**
  - Provides input to your system
    - »Step input, white noise, custom input, sine wave, ramp input（斜阶跃信号）, constant

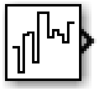
- **Sinks（信宿，输出设备模块库）**
  - »Scope: Outputs to plot，示波器
  - »simout: Outputs to a MATLAB vector (struct) on workspace
  - »Matlab mat file

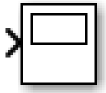
Toolboxes

- Math
  - Takes the signal and performs a math operation
    - »Add, subtract, round, multiply, gain, product
- Continuous
  - Adds differential equations to the system
    - »Integrals, Derivatives, Transfer Functions, State Space
- Discontinuities
  - Adds nonlinearities to your system
    - »quantizer（量化器）, dead zone（死区）
- Discrete
  - Simulates discrete difference equations
    - »Delay（延时），Difference（差分）
  - Useful for digital systems

Simulink Library Browser

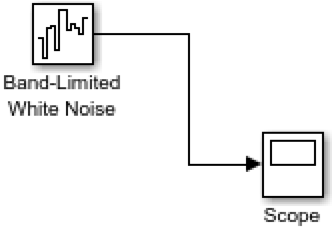
- The **Library Browser** contains various blocks that you can put into your model
- Examine some blocks:
  - ➢ Click on a library: “Sources”
    - – Drag a block into Simulink: “Band limited white noise”
  - ➢ Visualize the block by going into “Sinks”
    - – Drag a “Scope” into Simulink



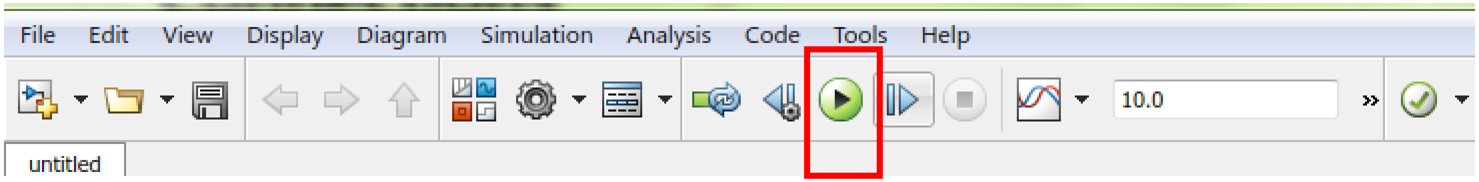


Connections

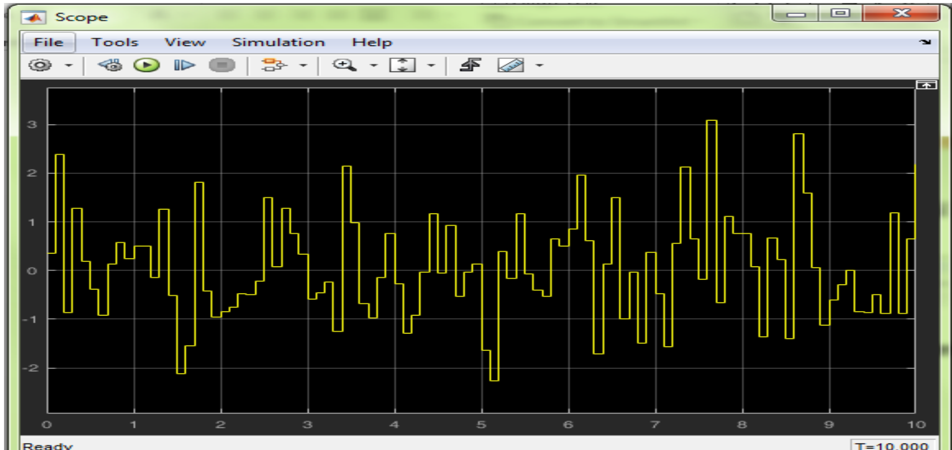
- Click on the arrow on the right of the band limited white noise box
- Drag the line to the scope
  - ➢ Connections between lines represent signal



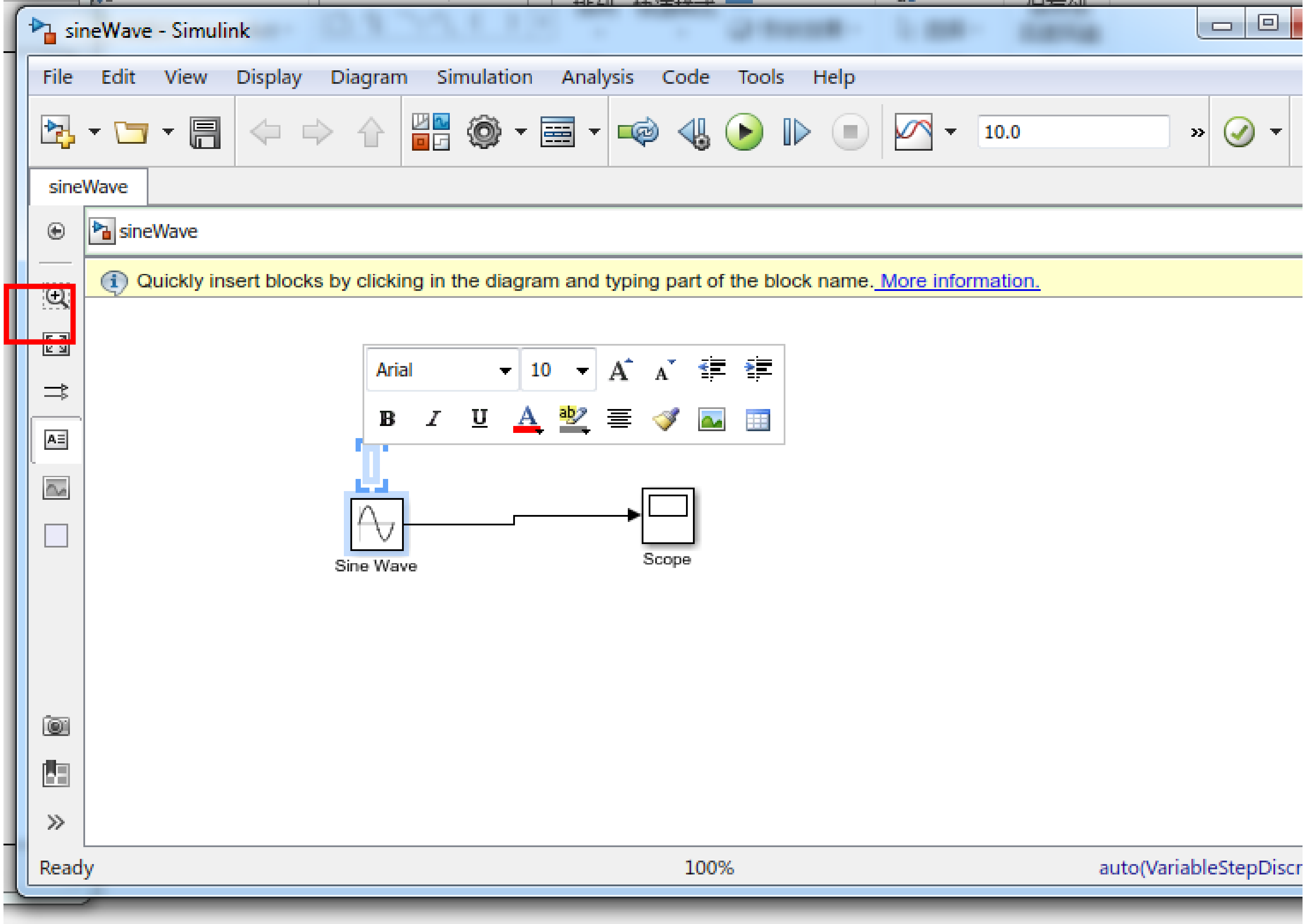
- Click the play button



- Double click on the scope.
  - ➢ This will open up a chart of the variable over the simulation time



- **模型注释**



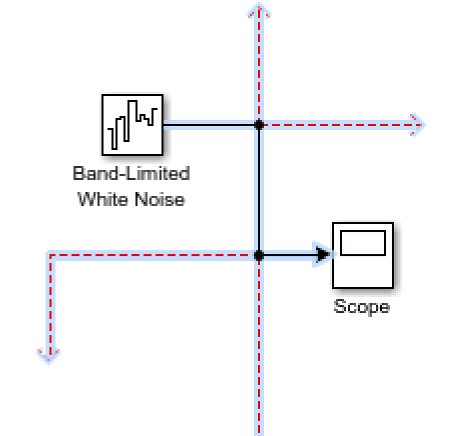
Quickly insert blocks by clicking in the diagram and typing part of the block name. More information.

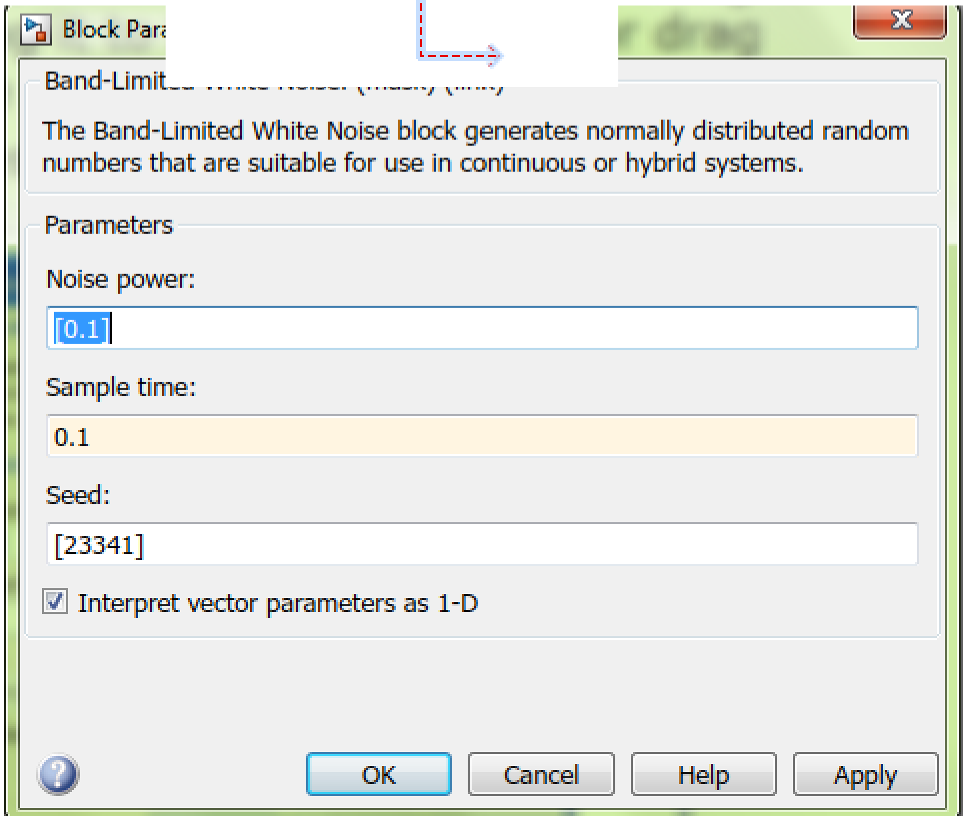
- **信号标签设置**

双击信号（即模块之间的连线），直接写

Connections, Block Specification

- To split connections, hold down ‘Ctrl’ when clicking on a connection, and drag it to the target block; or drag backwards from the target block
- To modify properties of a block, double-click it and fill in the property values.




Block Para

Band-Limit

The Band-Limited White Noise block generates normally distributed random numbers that are suitable for use in continuous or hybrid systems.

Parameters

Noise power:

[0.1]

Sample time:

0.1

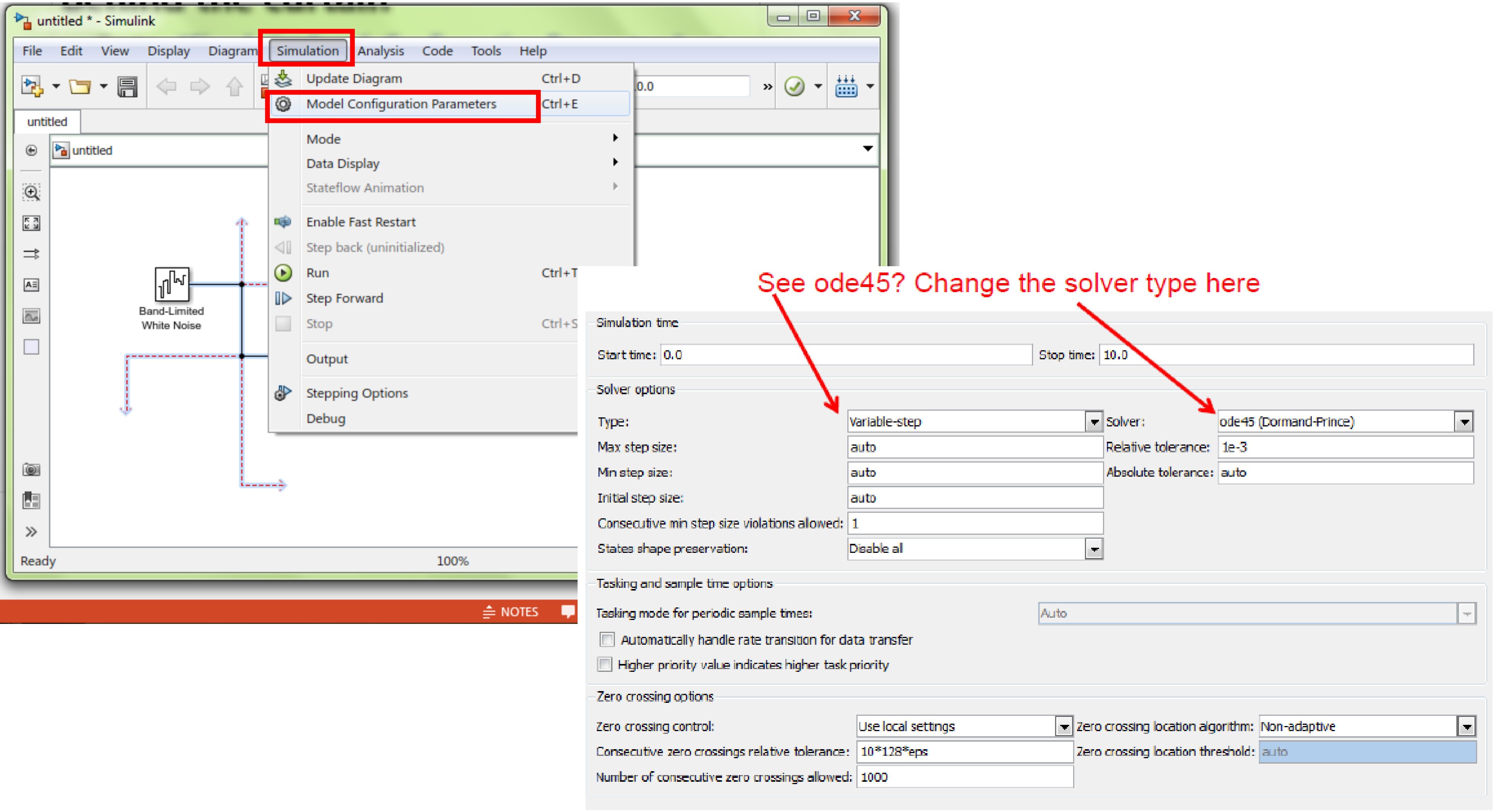
Seed:

[23341]

☑ Interpret vector parameters as 1-D

仿真参数设置

- Go to “Simulation” ->” Configuration Parameters” at the top menu



See ode45? Change the solver type here

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Max step size: auto Relative tolerance: 1e-3

Min step size: auto Absolute tolerance: auto

Initial step size: auto

Consecutive min step size violations allowed: 1

States shape preservation: Disable all

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

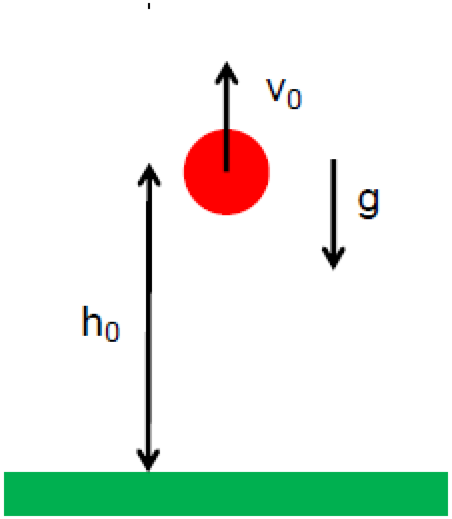
Zero crossing control: Use local settings Zero crossing location algorithm: Non-adaptive

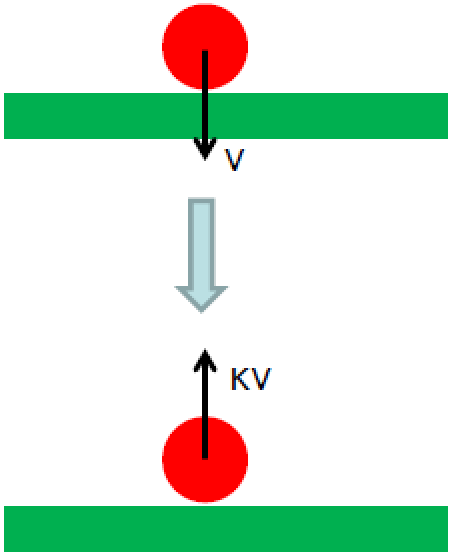
Consecutive zero crossings relative tolerance: 10*128*eps Zero crossing location threshold: auto

Number of consecutive zero crossings allowed: 1000

Exercise: Bouncing Ball Model

- Let’ s consider the following 1 dimensional problem
- A rubber ball is thrown from height h0 with initial velocity v0 in the z-axis (up/down).
- When the ball hits the ground (z=0), its velocity instantaneously flips direction and is attenuated by the impact





Exercise: Bouncing Ball Model

- Let’s consider the following 1 dimensional problem
- A rubber ball is thrown from height h0 with initial velocity v0 in the z-axis (up/down).
- When the ball hits the ground (z=0), its velocity instantaneously flips direction and is attenuated by the impact

$$m\frac{d^2z}{dt^2}=mg \quad v(t)=\frac{dz}{dt} \quad v\left(t^+\big|_{z=0}\right)=-\kappa v\left(t^-\big|_{z=0}\right)$$

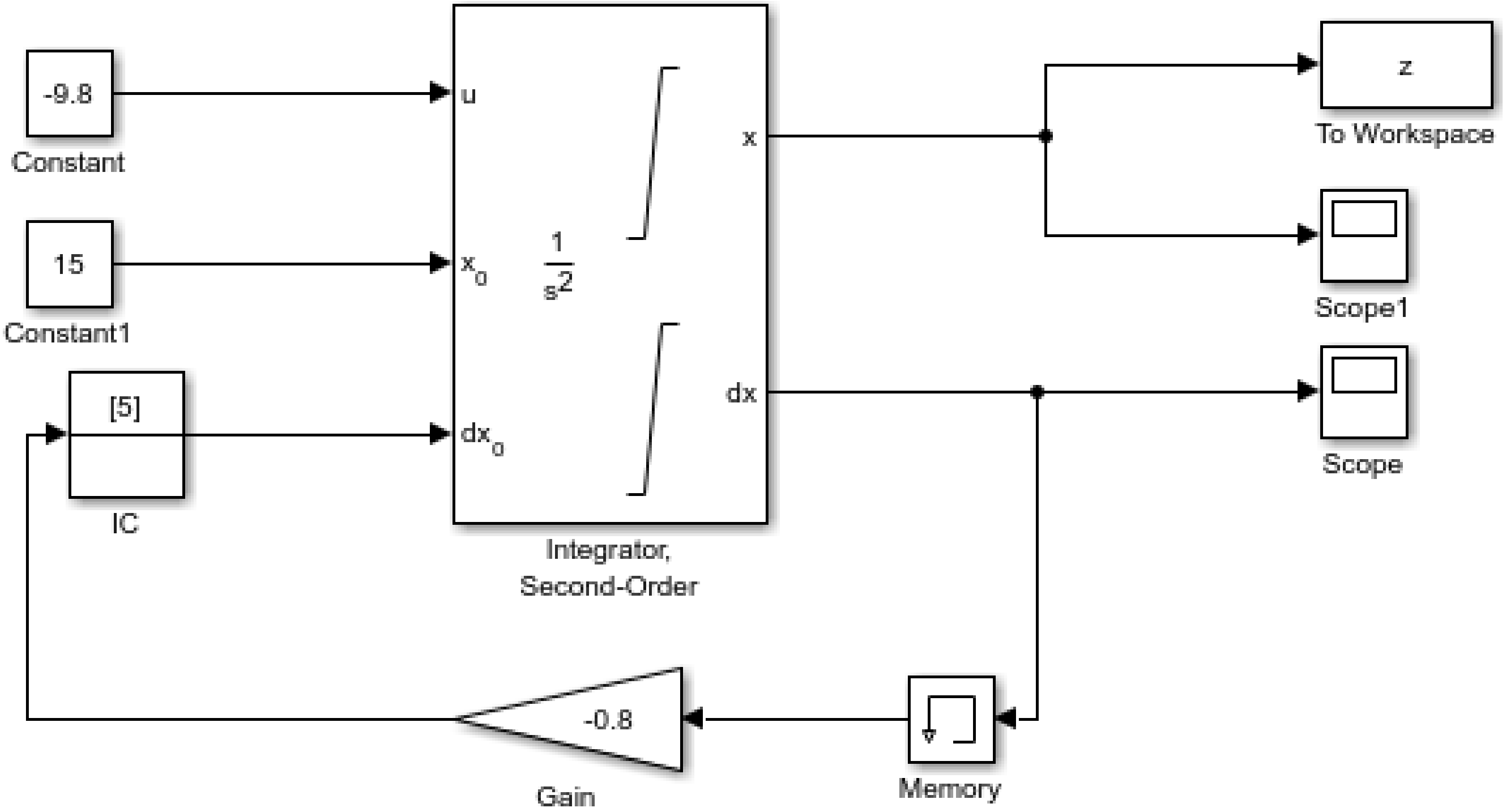
$$z(t=0)=h_0 \quad v(t=0)=v_0$$

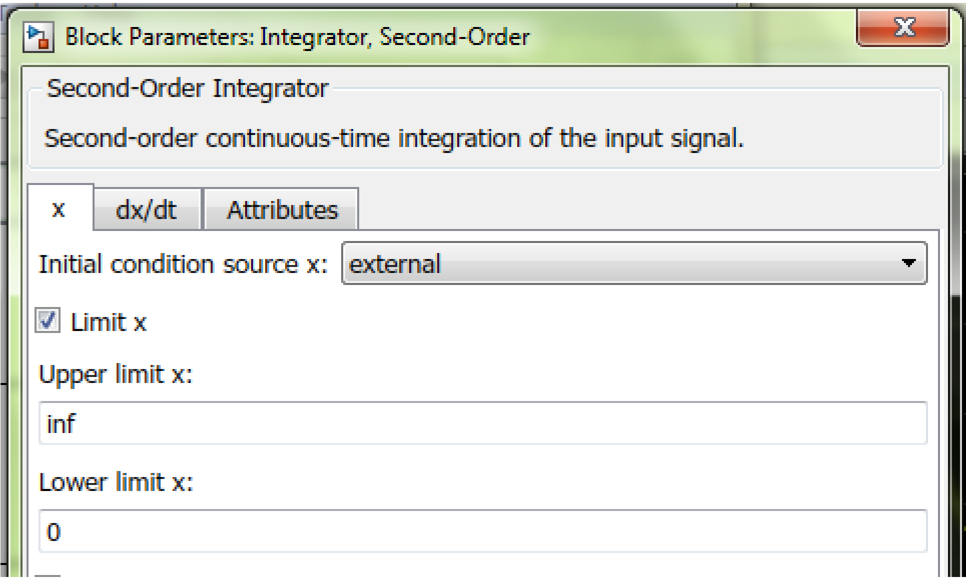
- Integrating, we can obtain the balls height and velocity as a function of time

$$v(t)=\int_0^t g\,d\tau \quad z(t)=\int_0^t v(\tau)\,d\tau$$

Exercise: Simulink Model

- Using the second order integrator with limits and reset, your model will look like this





Block Parameters: Integrator, Second-Order

Second-Order Integrator

Second-order continuous-time integration of the input signal.

Initial condition source x: external

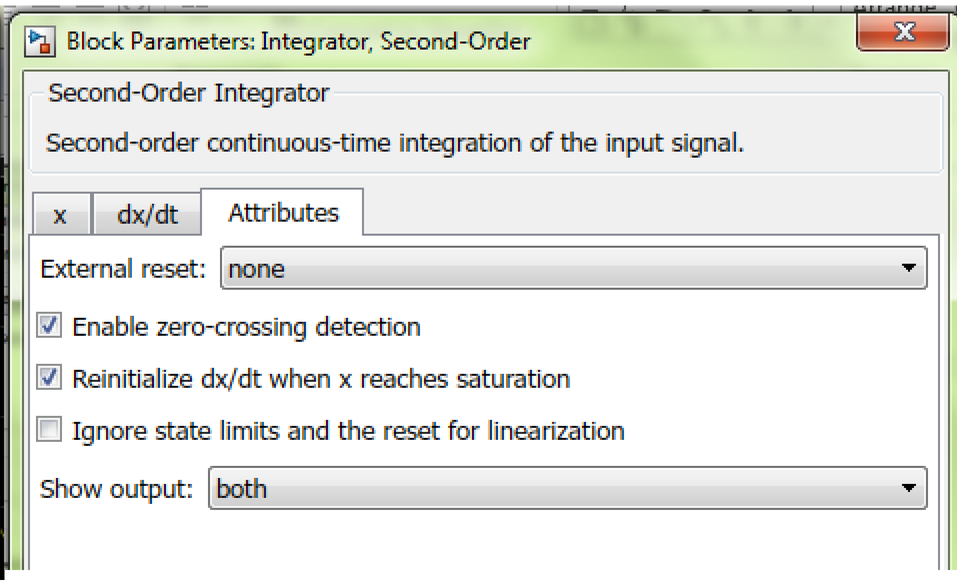
☑ Limit x

Upper limit x:

inf

Lower limit x:

0



Block Parameters: Integrator, Second-Order

Second-Order Integrator

Second-order continuous-time integration of the input signal.

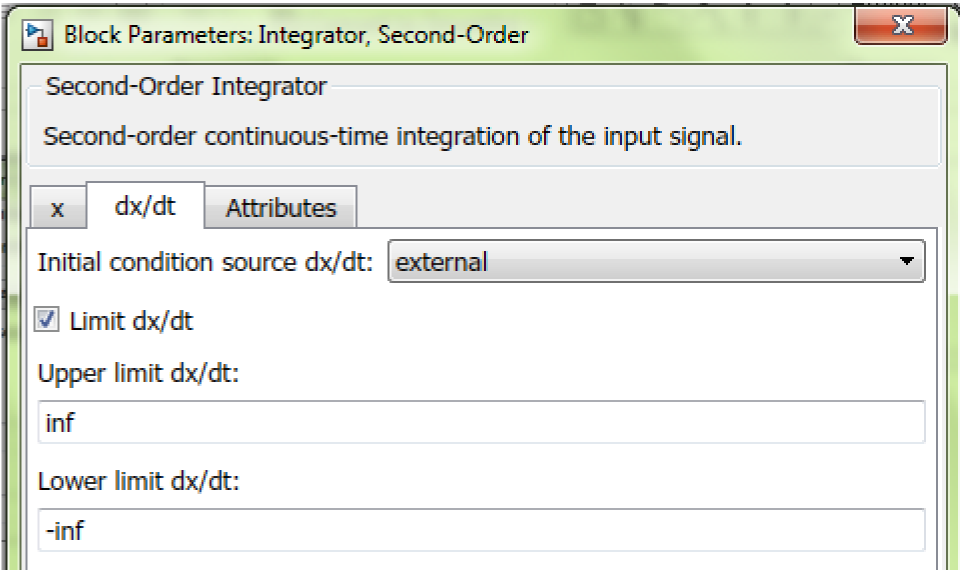
External reset: none

☑ Enable zero-crossing detection

☑ Reinitialize dx/dt when x reaches saturation

□ Ignore state limits and the reset for linearization

Show output: both



Block Parameters: Integrator, Second-Order

Second-Order Integrator

Second-order continuous-time integration of the input signal.

Initial condition source dx/dt: external

☑ Limit dx/dt

Upper limit dx/dt:

inf

Lower limit dx/dt:

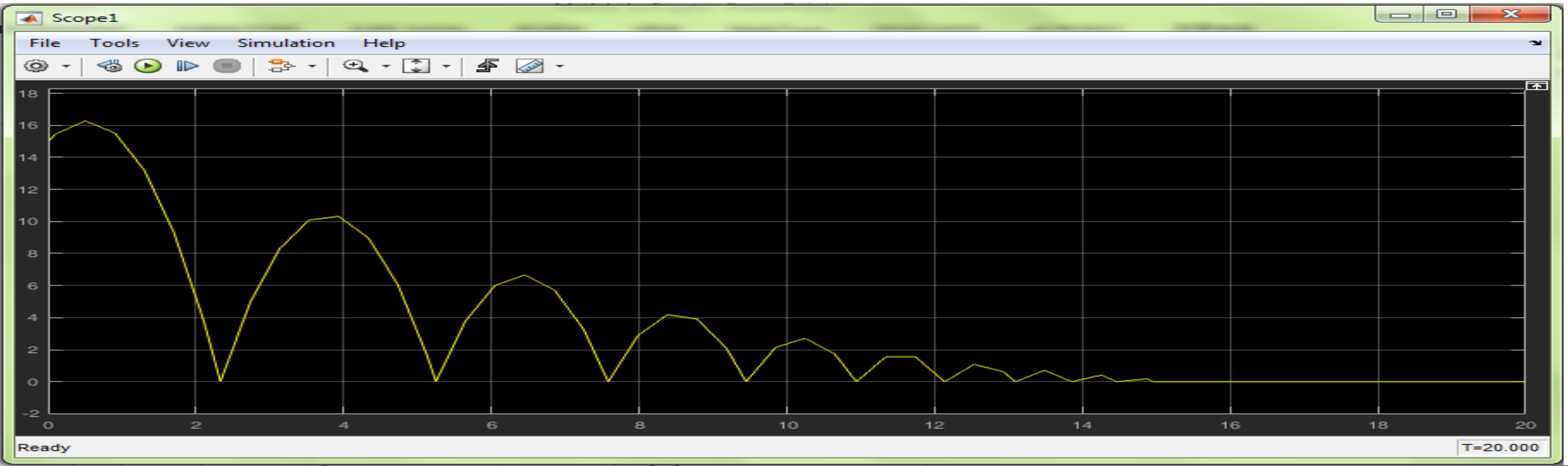
-inf

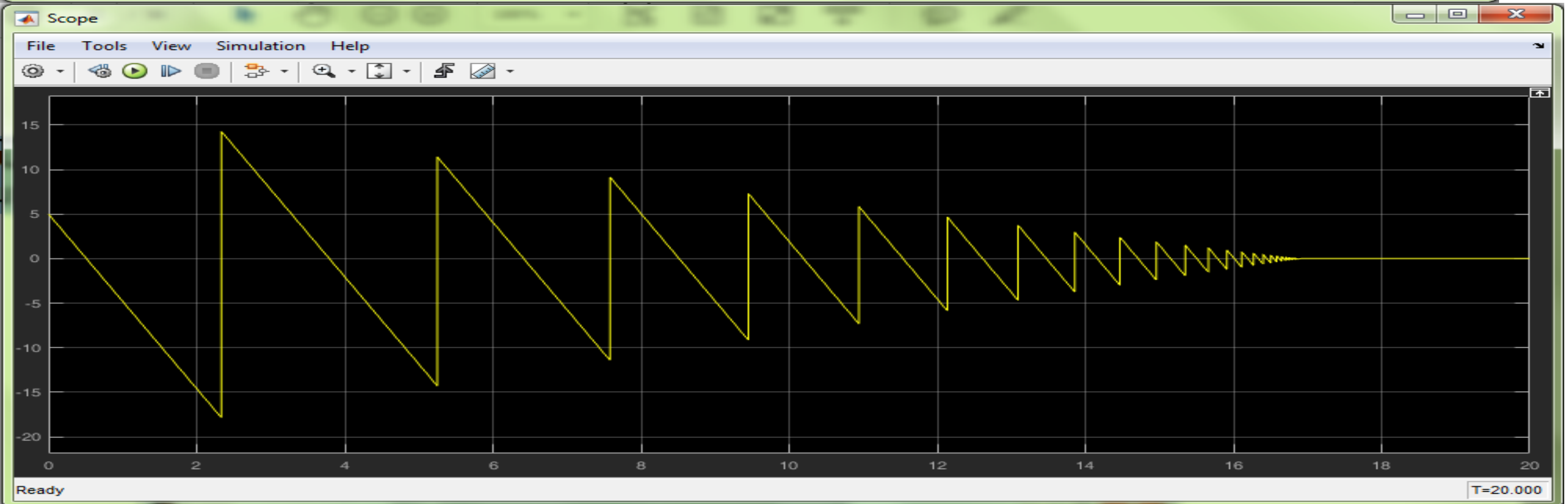
更改输入输出方向：

模块-右键-rotate and flip-flip block

Simulink Results

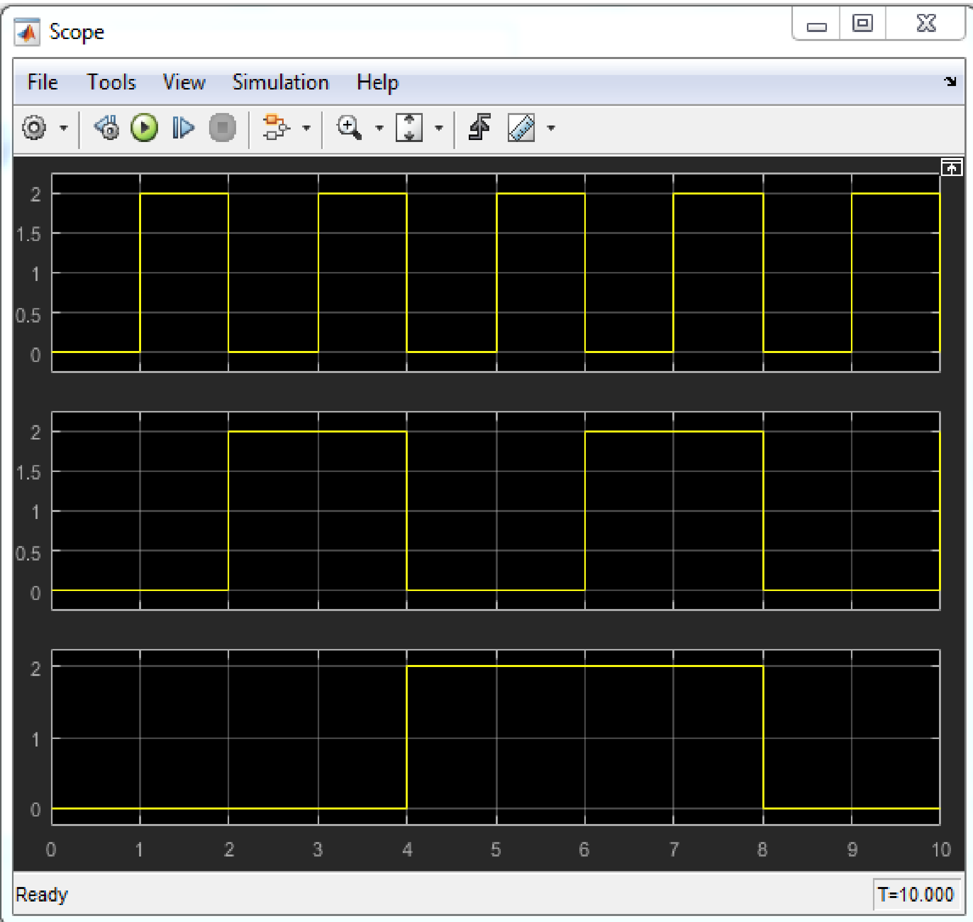
- Running the model yields the balls height and velocity as a function of time



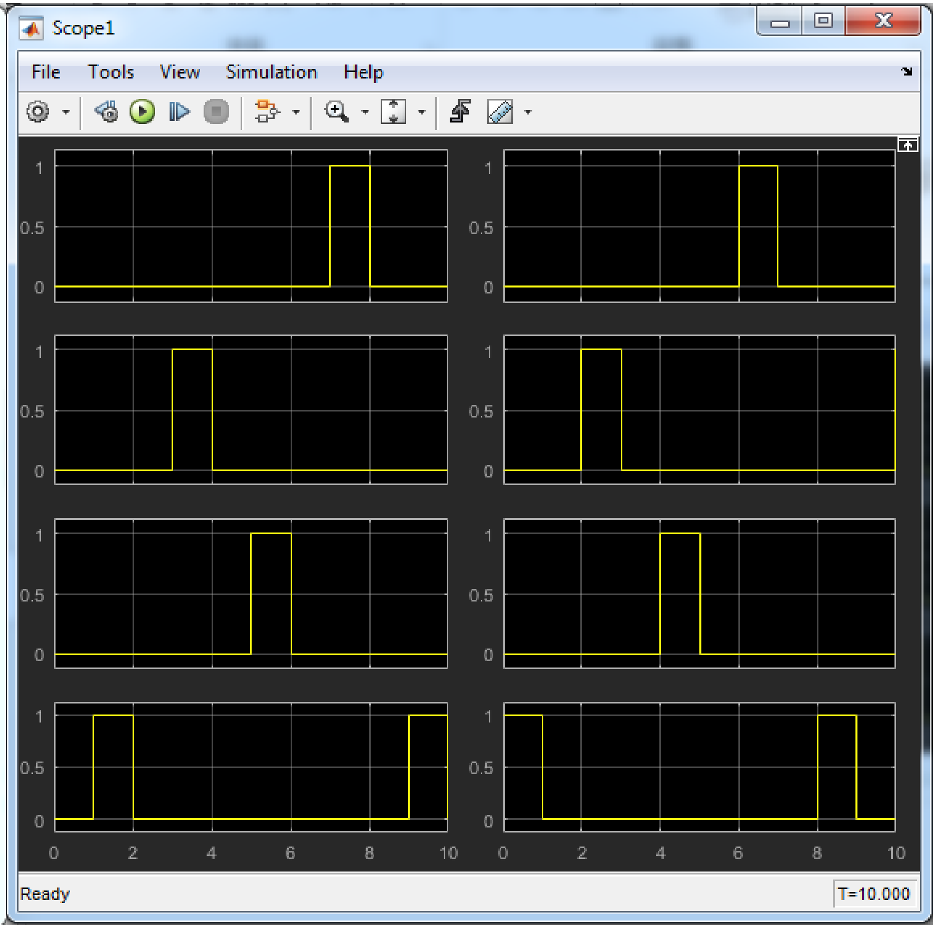


Simulink仿真三八译码电路

输入信号（3路）



输出信号（8路）



- 用到的模块：pulse generator, logical operator（选合适的逻辑运算）, scope（更改输入端口数）
- 按运算逻辑连线
- 仿真参数设置

  Simulation->model configuration parameters->solver->discrete

- 模块参数设置（pulse generator模块）

  Pulse type: sample based;

  Amplitude: 2

  Period: 2

  Pulse width: 1

  Phase delay:1

  Sample time: 1, 2, 4 (respectively)

显示窗口：
View->layout

Outline

- Symbolic Toolbox
- Simulink
- **Image Processing**
- Miscellaneous Useful Functions
- Graphical User Interfaces

Image Processing

- Image enhancement
  - Adjust image contrast, intensities, etc.
- Filtering and deblurring
  - Convolution and deconvolution
- Finding edges
  - Image gradient, edge
- Finding circles
  - Hough transform

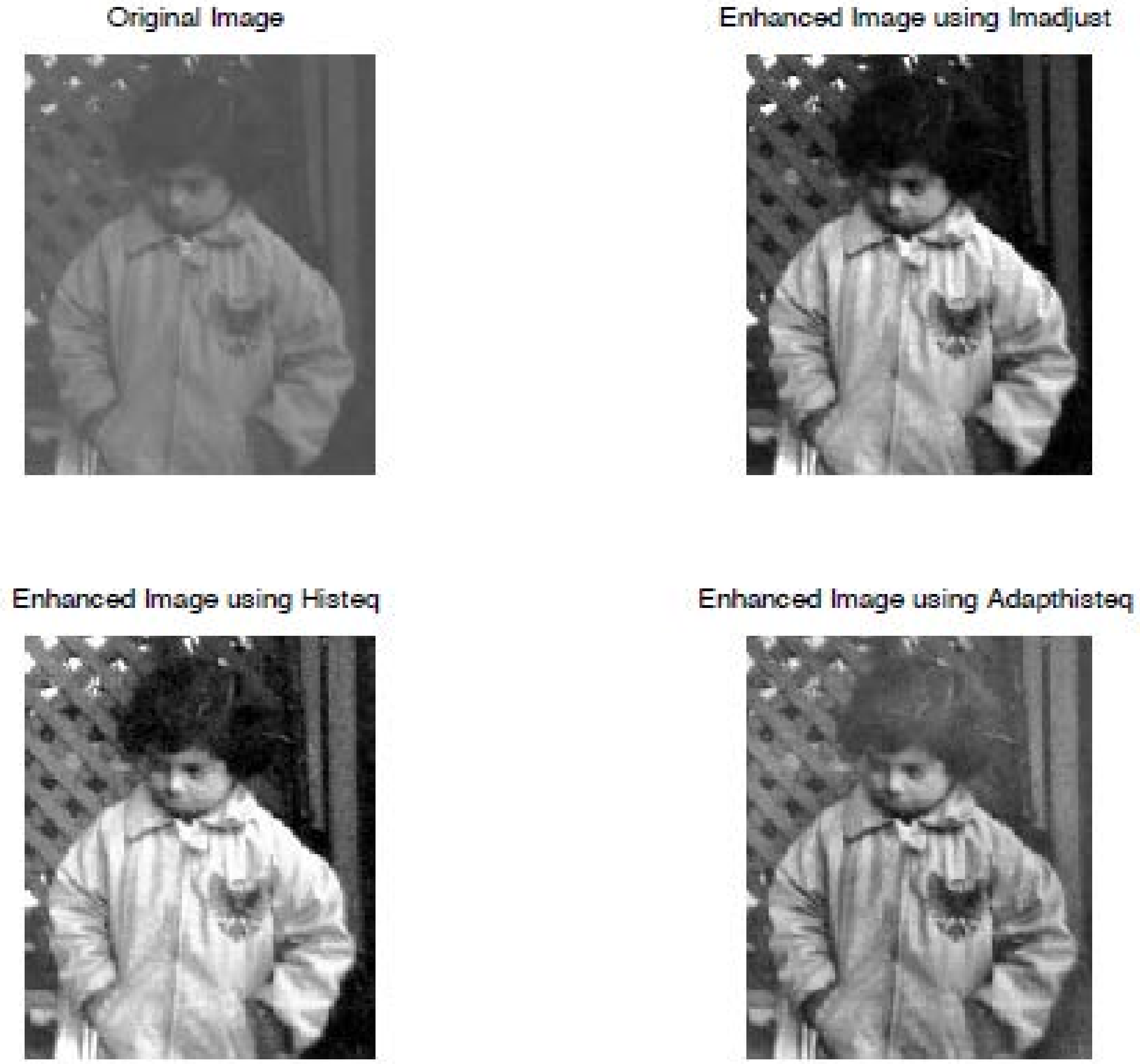
Image Processing

- Image Restoration
  - Denoising
- Image Enhancement & Analysis
  - Contrast Improvement
    - imadjust, histeq, adapthisteq
  - Edge Detection
    - edge
  - Sharpening
  - Segmentation

Exercise: Contrast Improvement

- In this exercise, first we want to load the image “pout.tif”. You can use **imread**.
- Then for a better comparison we want our image to have a width of 200 pixels. Use **imresize**
- Finally, we want to compare the results of three functions **imadjust, histeq, adapthisteq** for contrast enhancement. Display the original image and the three enhanced images in a single figure.

Exercise: Contrast Improvement



Original Image

Enhanced Image using Imadjust

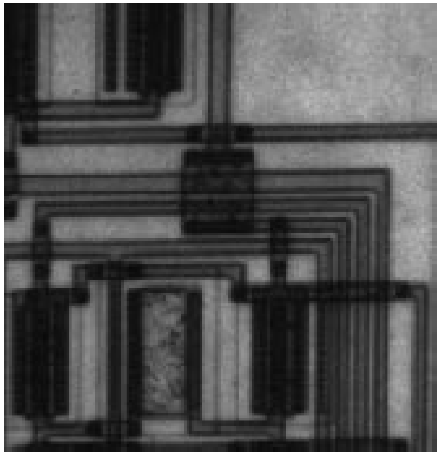
Enhanced Image using Histeq

Enhanced Image using Adapthisteq

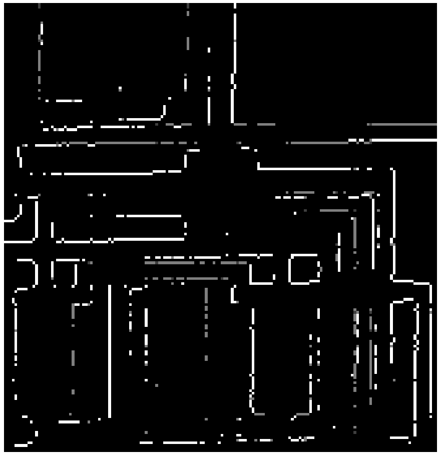
Exercise: Edge Detection

- We know that edge detection is mainly highpass filtering the image.
- First load the image "circuit.tif" and then plot the edges in that figure using the function **edge** and the filters **"sobel"** **,** **"prewitt"** **.** Also use "**canny**" as another method for edge detection using **edge**.

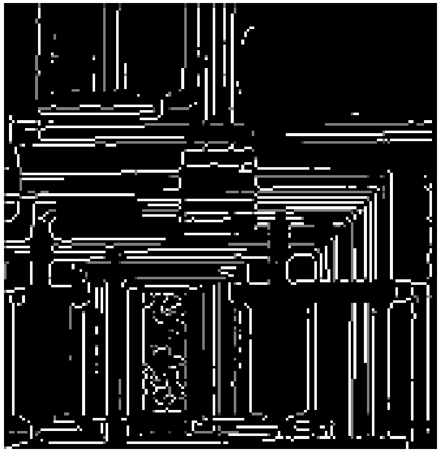
Exercise: Edge Detection



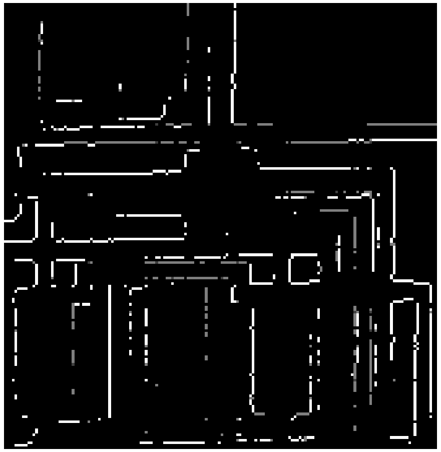
Original Image



Edges found using sobel filter



Edges found using the "canny" method



Edges found using prewitt filter

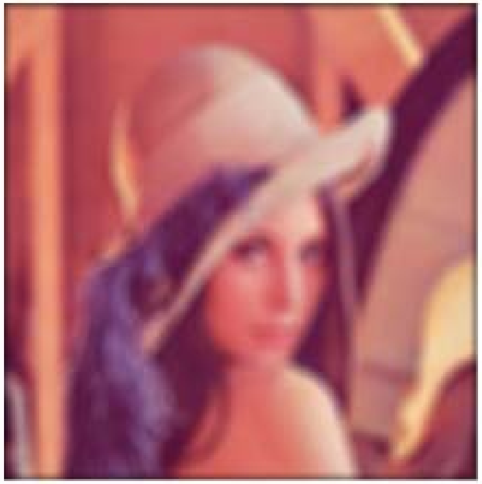
Image Enhancement

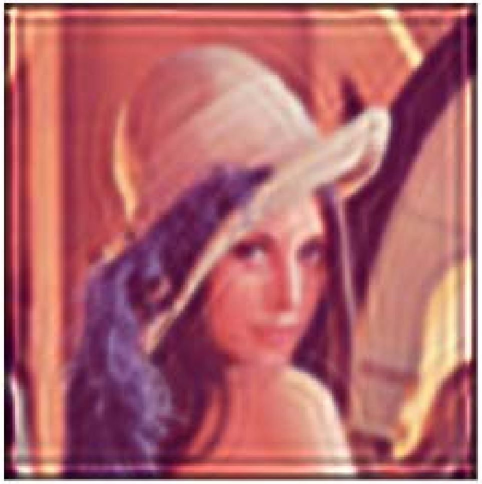
- Commonly-used: **imread**, **imwrite**, **imshow, imresize**
  - » **im = imread('pout.tif');**
  - » **imtool(im);**
    - Convenient for editing in figure window
- Adjust intensity values / colormap
  - » **imadjust(im);**
    - Increase contrast (1% of data saturated at low/high intensities)
  - » **imadjust(im,[.4 .6],[0 1]);**
    - Clips off intensities below .4 and above .6, Stretches resulting intensities to 0 and 1
    - What happens if used [1 0] instead of [0 1]?
    - Also works for RGB; see **doc**

Filtering and Deblurring

Pillbox filter:

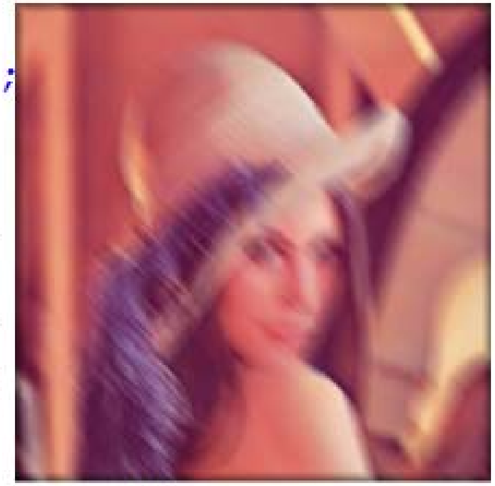
```
f = fspecial('disk',10);
imblur = imfilter(im,f);
deconvblind(imblur,f);
```

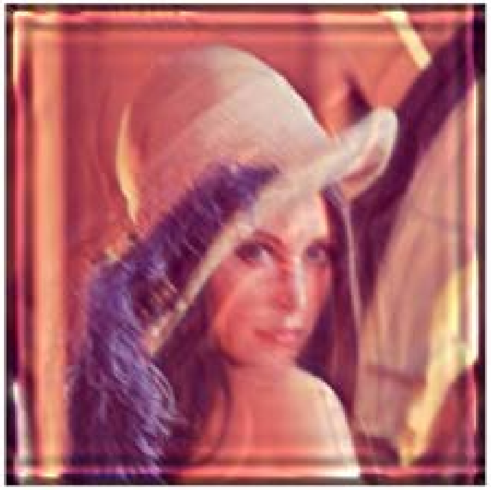


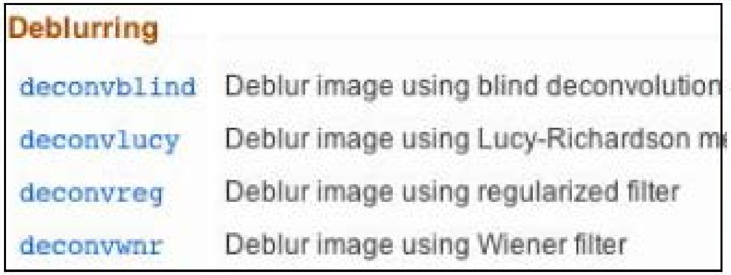


Linear motion blur:

```
f=fspecial('motion',30,135);
imblur = imfilter(im,f);
deconvblind(imblur,f);
```



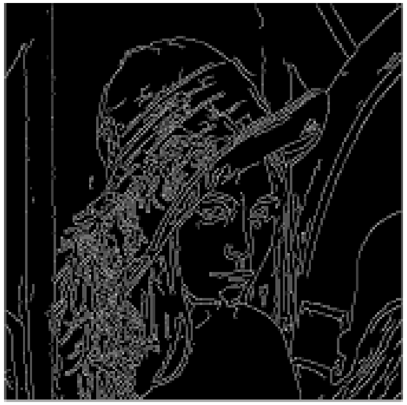




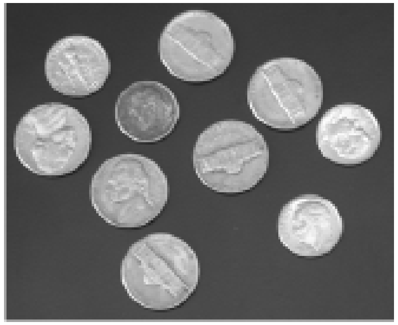
Deblurring	
deconvblind	Deblur image using blind deconvolution
deconvlucy	Deblur image using Lucy-Richardson m
deconvreg	Deblur image using regularized filter
deconvwnr	Deblur image using Wiener filter

Finding Edges

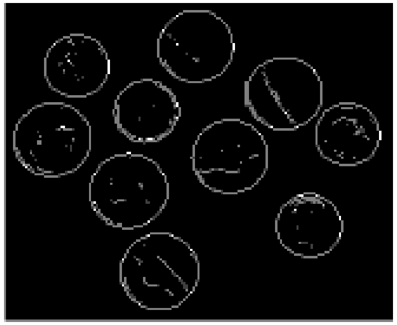
- Image gradients: `imgradient, imgradientxy`
- Application: `edge`
  - » `edge(im); % Sobel`
  - » `edge(im, 'canny');`
- Images must be in grayscale
  - » `rgb2gray`



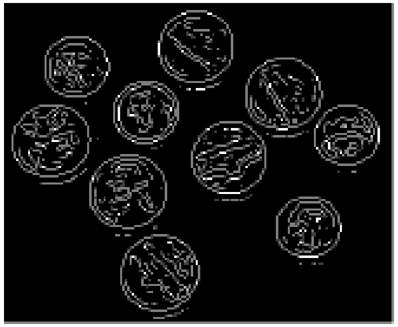
Original (coins.png)



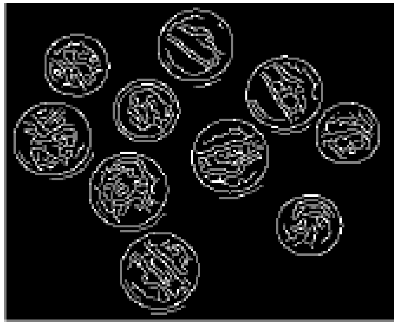
Sobel



Laplacian



Canny



Other Cool Stuff

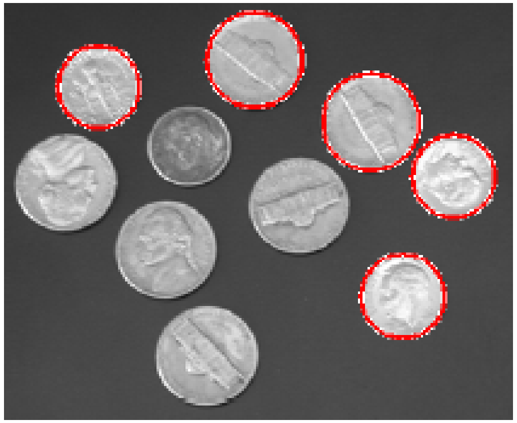
- Finding circles

```
» im = imread('coins.png');
» [centers,radii,metric] = imfindcircles(im, [15 30]);
```

  - ➢ Finds circles with radii within range, ordered by strength

```
» imshow(im)
» viscircles(centers(1:5,:), radii(1:5));
```

- Extract other shapes with Hough transform



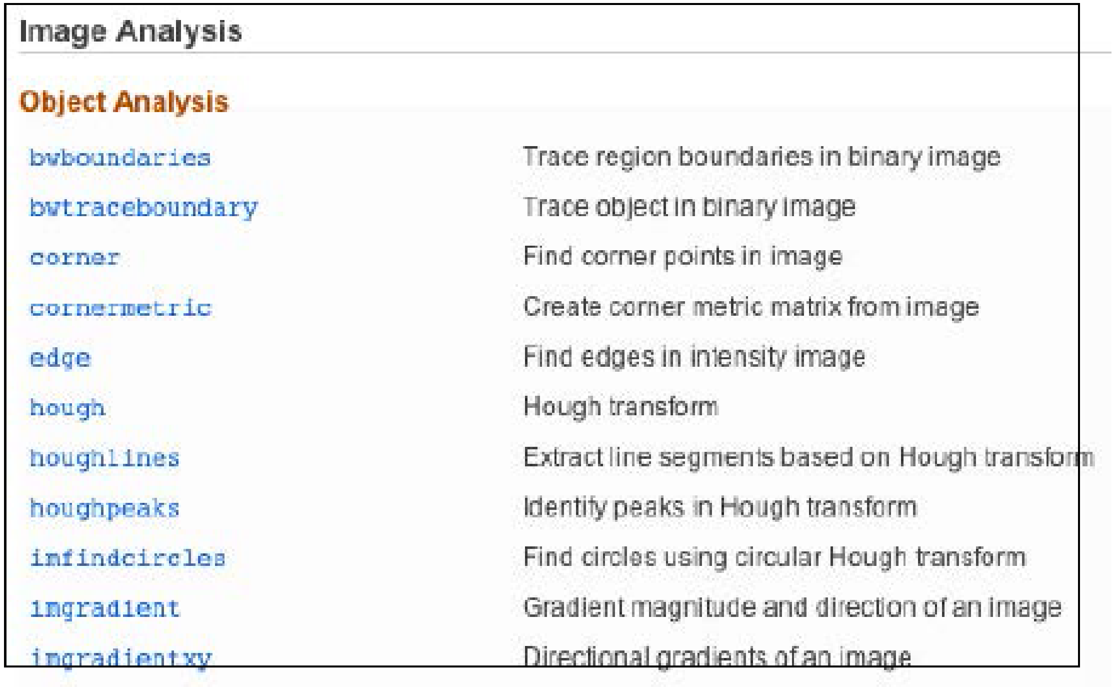


Image Analysis

Object Analysis

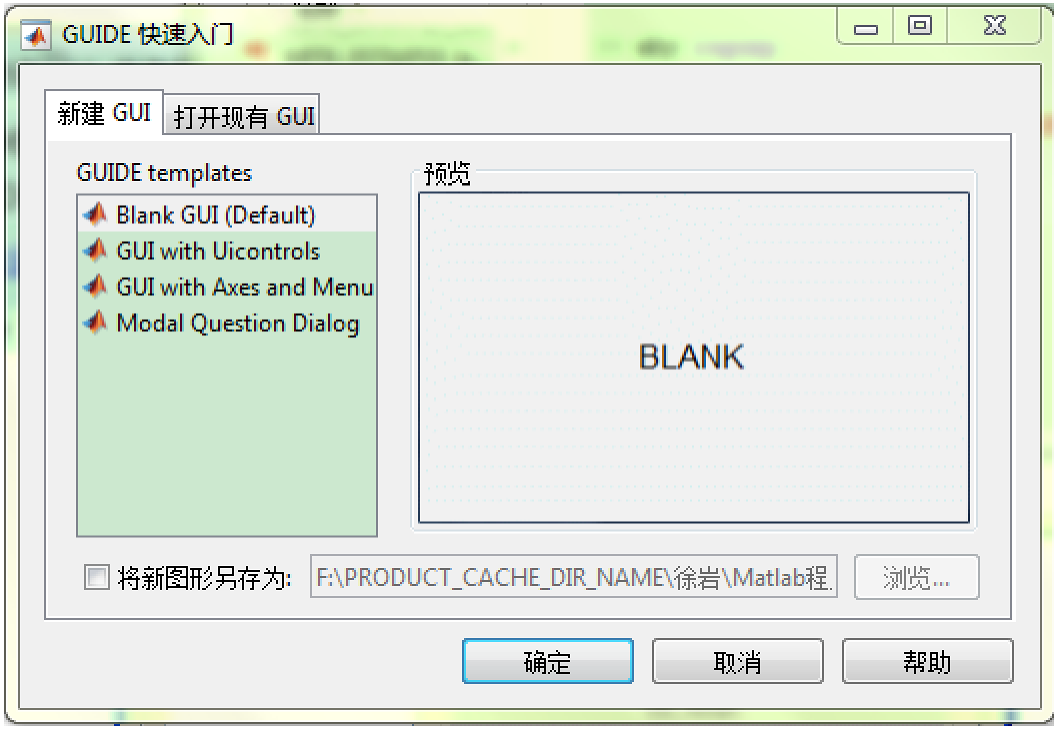
bwboundaries	Trace region boundaries in binary image
bwtraceboundary	Trace object in binary image
corner	Find corner points in image
cornermetric	Create corner metric matrix from image
edge	Find edges in intensity image
hough	Hough transform
houghlines	Extract line segments based on Hough transform
houghpeaks	Identify peaks in Hough transform
imfindcircles	Find circles using circular Hough transform
imgradient	Gradient magnitude and direction of an image
imgradientxy	Directional gradients of an image

Outline

- Symbolic Toolbox
- Simulink
- Image Processing
- Miscellaneous Useful Functions
- **Graphical User Interfaces**

Making GUIs

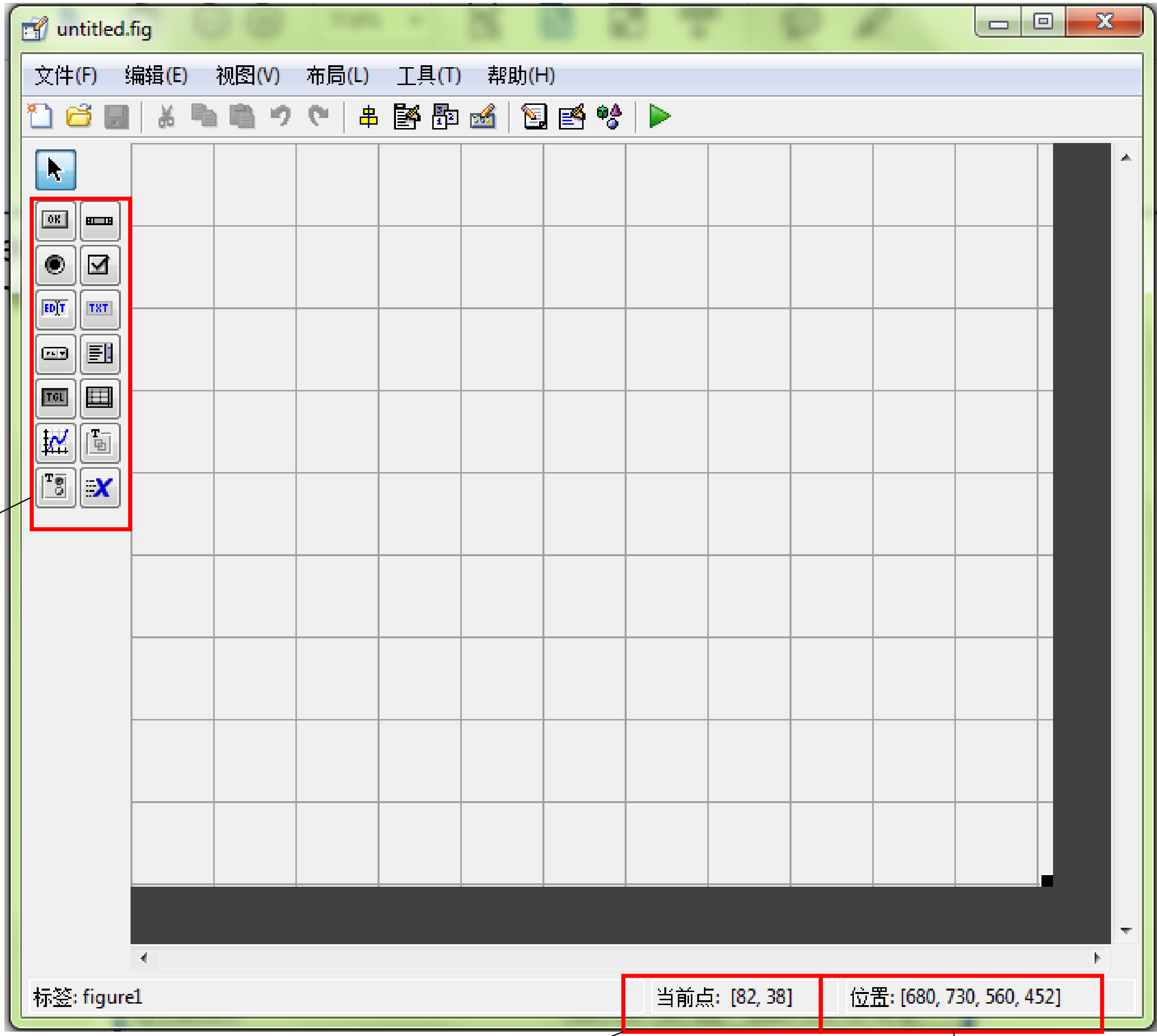
- It's really easy to make a graphical user interface in Matlab
- To open the graphical user interface development environment, type **guide**
  - » **guide**
    - Select Blank GUI



□ 将新图形另存为: F:\PRODUCT_CACHE_DIR_NAME\徐岩\Matlab程

Draw the GUI

- Select objects from the left, and draw them where you want them



标签: figure1

当前点: [82, 38]

位置: [680, 730, 560, 452]

各种对象：
单选、多选、滑动条、
弹出式菜单，等等

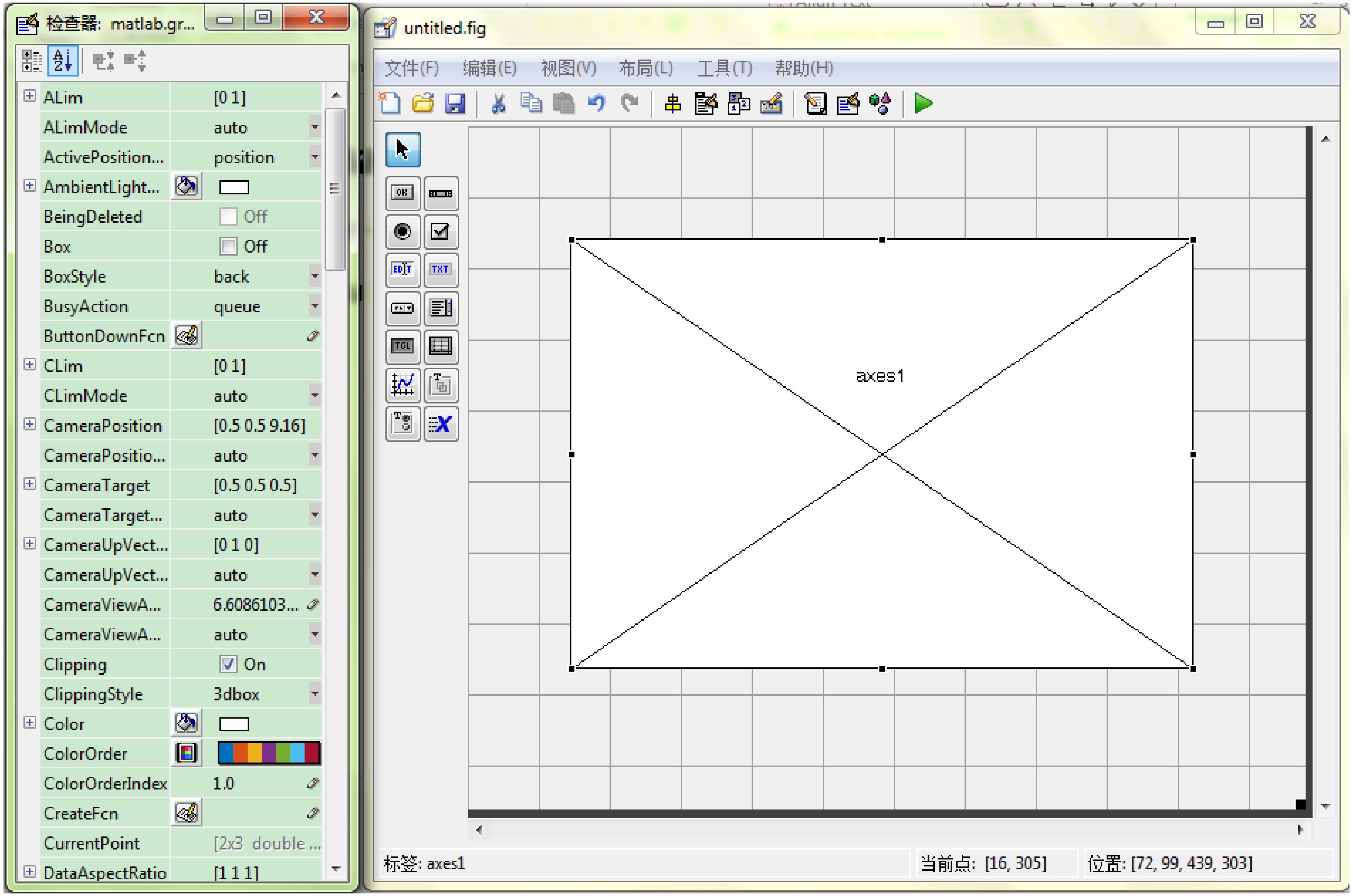
当前鼠标
所在位置

选定对象
所在位置

Change Object Settings

- Double-click on objects to open the **Inspector**.

Here you can change all the object's properties.



ALim	[0 1]
ALimMode	auto
ActivePosition...	position
AmbientLight...	
BeingDeleted	□ Off
Box	□ Off
BoxStyle	back
BusyAction	queue
ButtonDownFcn	
CLim	[0 1]
CLimMode	auto
CameraPosition	[0.5 0.5 9.16]
CameraPositio...	auto
CameraTarget	[0.5 0.5 0.5]
CameraTarget...	auto
CameraUpVect...	[0 1 0]
CameraUpVect...	auto
CameraViewA...	6.6086103...
CameraViewA...	auto
Clipping	☑ On
ClippingStyle	3dbox
Color	
ColorOrder	
ColorOrderIndex	1.0
CreateFcn	
CurrentPoint	[2x3 double ...
DataAspectRatio	[1 1 1]

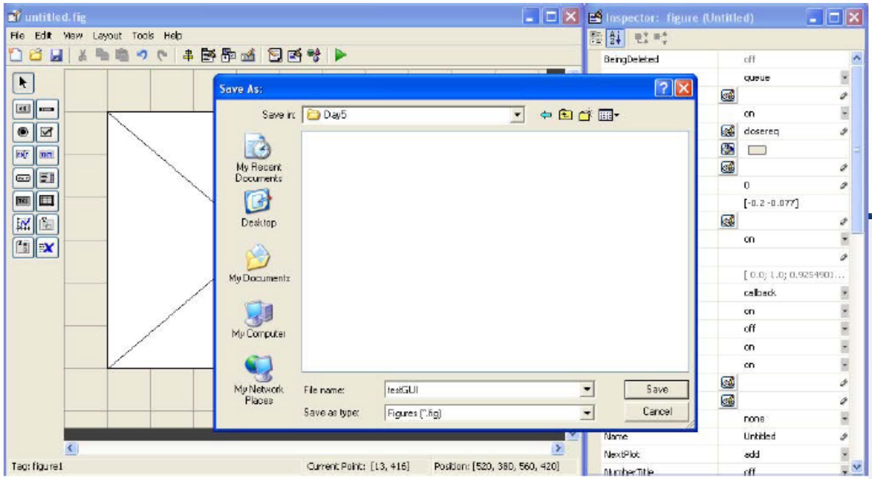
标签: axes1

当前点: [16, 305]

位置: [72, 99, 439, 303]

Save the GUI

- When you have modified all the properties, you can save the GUI
- Matlab saves the GUI as a **.fig** file, and generates an **m-file**!



Save in: Day5

File name: testGUI

Save as type: Figures (*.fig)

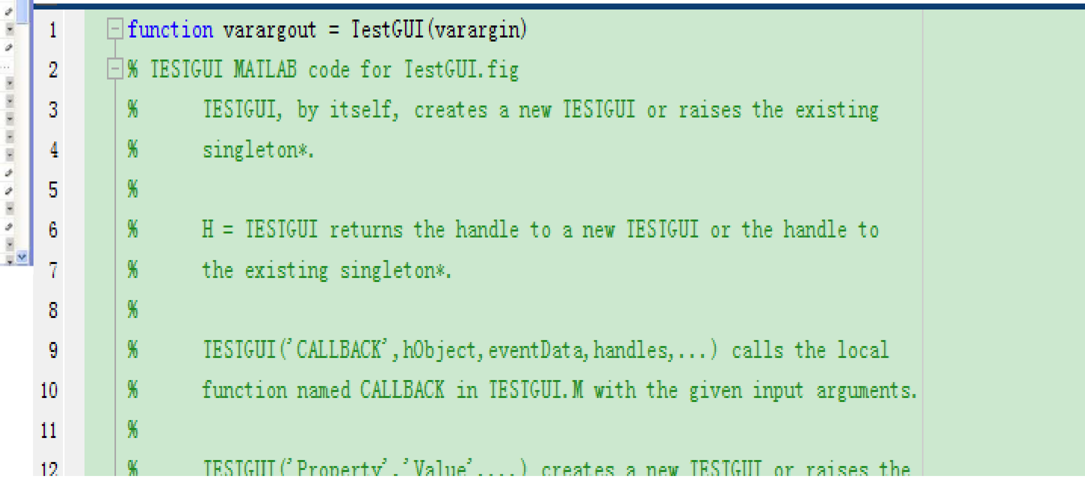
Tag: figure1 Current Point: [13, 416] Position: [520, 380, 560, 420]

BeingDeleted off

Name Untitled

NextPlot add

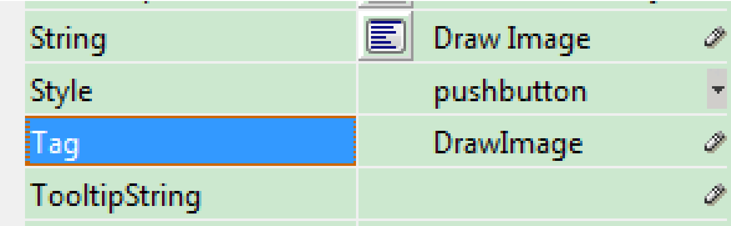
NumberTitle off



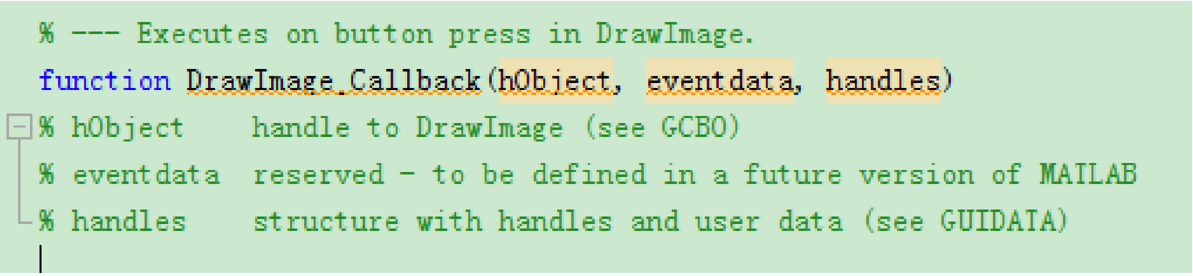
```
function varargout = TestGUI(varargin)
% TESTGUI MATLAB code for TestGUI.fig
%      TESTGUI, by itself, creates a new TESTGUI or raises the existing
%      singleton*.
%
%      H = TESTGUI returns the handle to a new TESTGUI or the handle to
%      the existing singleton*.
%
%      TESTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TESTGUI.M with the given input arguments.
%
%      TESTGUI('Property','Value',...) creates a new TESTGUI or raises the
```

Add Functionality to M-File

- To add functionality to your buttons, add commands to the 'Callback' functions in the m-file.
- For example, when the user clicks the Draw Image button, the **drawimage_Callback** function will be called and executed .
- All the data for the GUI is stored in the handles, so use **set** and **get** to get data and change it if necessary.
- Any time you change the handles, save it using **guidata**
  - » **guidata(handle,data);**



String	Draw Image
Style	pushbutton
Tag	DrawImage
TooltipString	



```
% --- Executes on button press in DrawImage.
function DrawImage_Callback(hObject, eventdata, handles)
% hObject    handle to DrawImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
f=openfig('TestGUI.fig')
data=guihandles(f)
h=get(data.axes1)
set(data.axes1,'XGrid','on')
```

```
data.axes1.YGrid='on'
guidata(f, data)
```

Running the GUI

- To run the GUI, just type its name in the command window and the GUI will pop up.

GUI Helper Functions

- Use keyboard to allow debugging from command window.

GUI variables will appear in the workspace.

Use return to exit debug mode

- Use built-in GUI modals for user input:
  - »uigetfile
  - »uiputfile
  - »inputdlg
- And more… (see help for details)