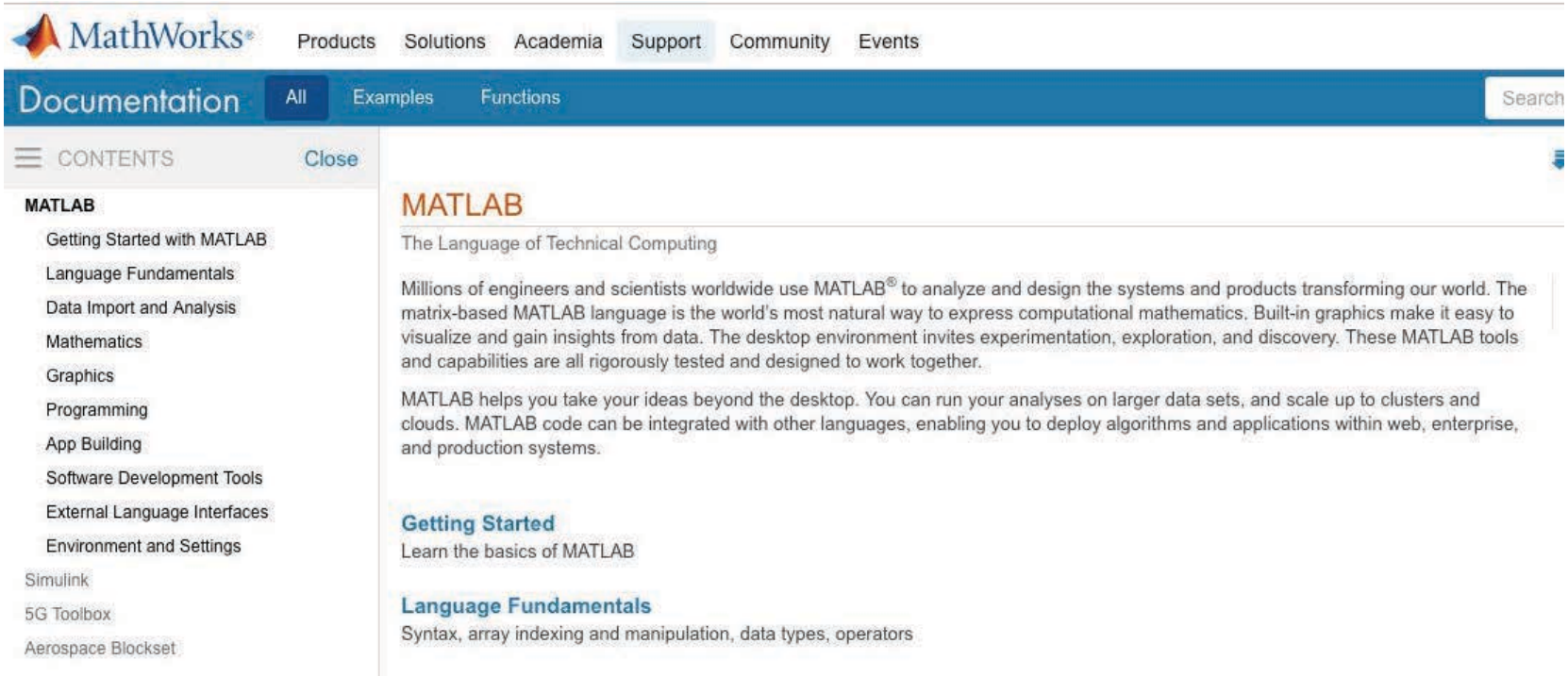
Outline

- Documentation
- Misc. Useful Functions
- Graphical User Interfaces
- Simulink
- Symbolic Toolbox
- Image Processing
- Hardware Interface

Official Documentation

- http://www.mathworks.com/help/matlab/



Miscellaneous Matlab (1)

- The command **deal** can make variable initialization simpler

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

- Execute Perl scripts using the command **perl**

```
» perl('myPerlFile.pl');
```

Miscellaneous Matlab (2)

- Use **regexp** for powerful regular expression operations

```
» str = 'The staff email is example@example.edu';
» pat = '([\w-.])+@([\w-.])+';
» r = regexp(str, pat, 'tokens')
» name = r{1}{1};     % name = '6.057-staff'
» domain = r{1}{2};   % domain = 'mit.edu'
```

- Set the root defaults by using the handle 0

```
» get(0, 'Default')
» set(0, 'DefaultLineLineWidth', 2);
```

- Edit the datatip text display function to show customized information
- You can also import Java classes (but don't)

```
» import java.util.Scanner
```

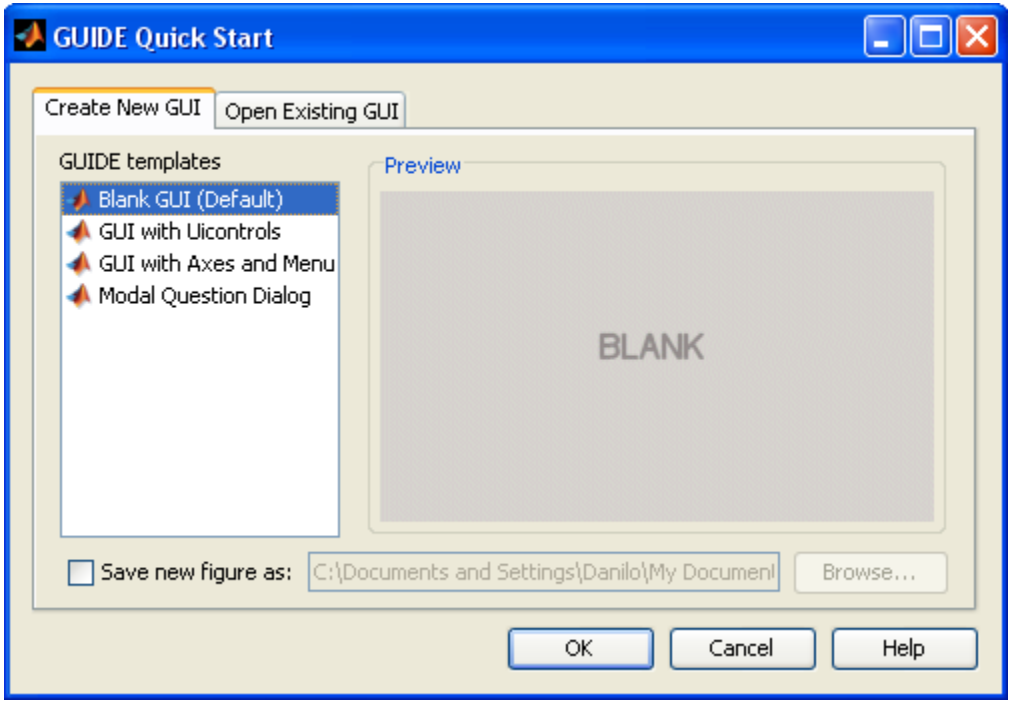
- If you're not sure about something – just ask Matlab **why**

Making GUIs

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

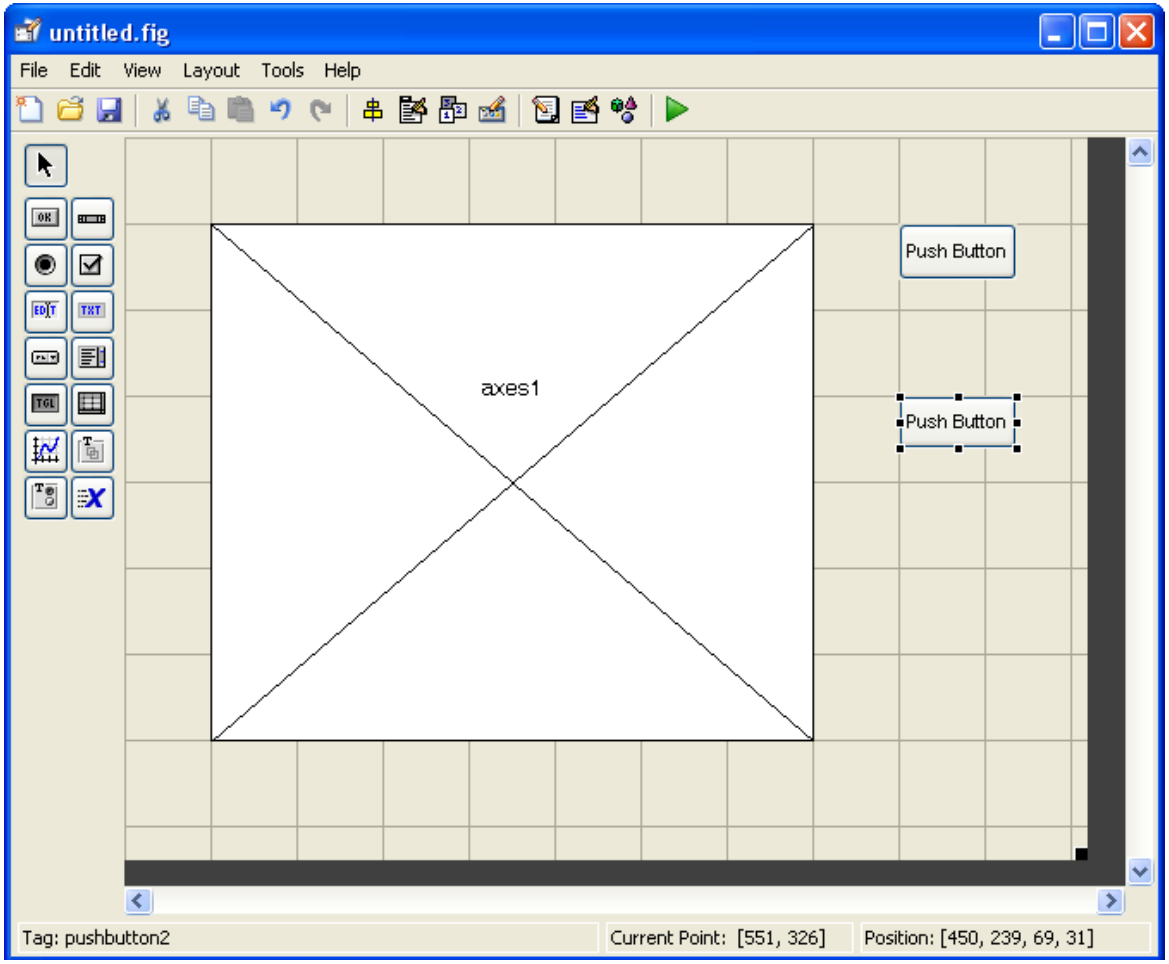
  » **`guide`**

  ➢ Select Blank GUI



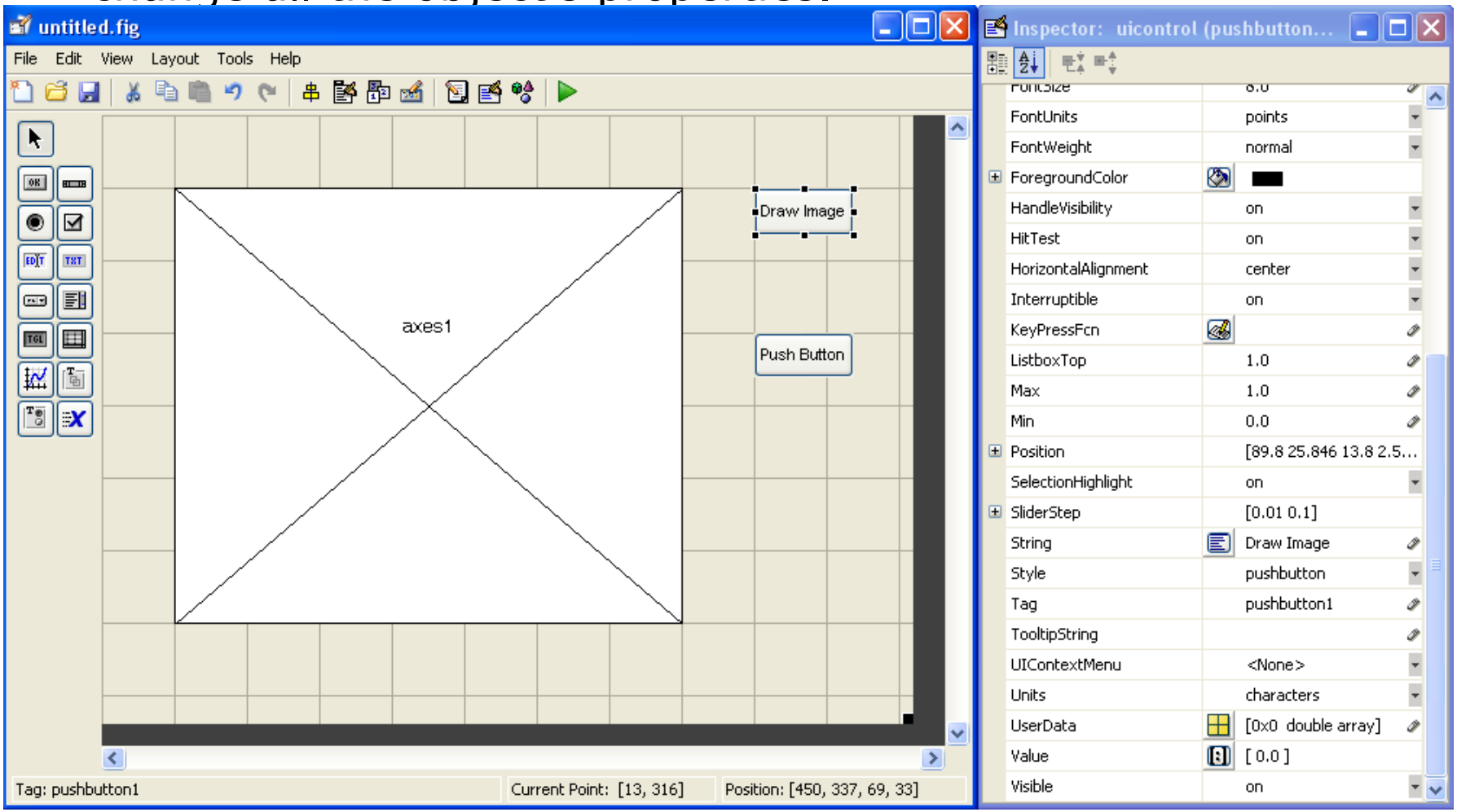
Draw the GUI

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



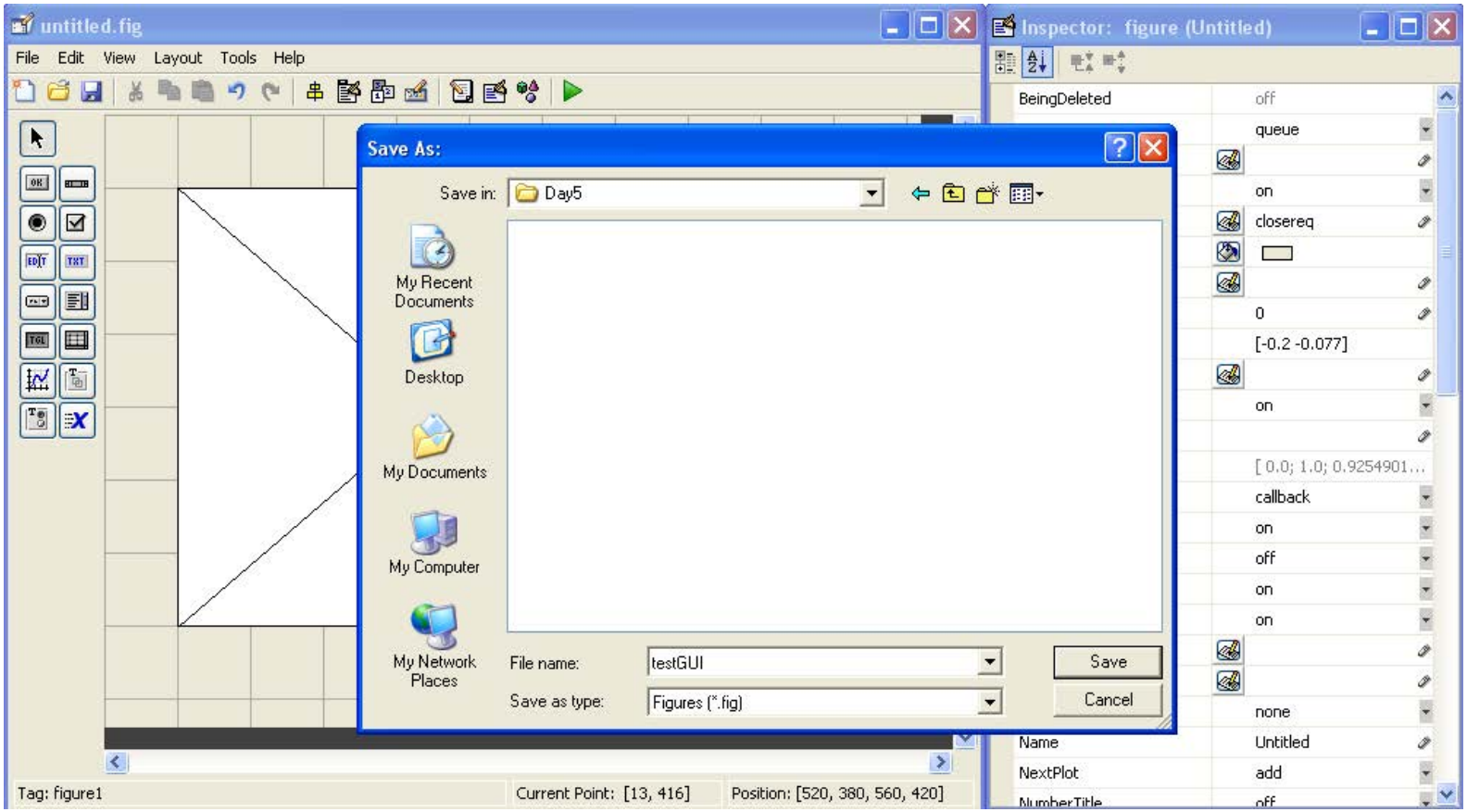
Change Object Settings

- Double-click on objects to open the Inspector. Here you can change all the object's properties.



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!



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(handles.Figure1,handles);
```

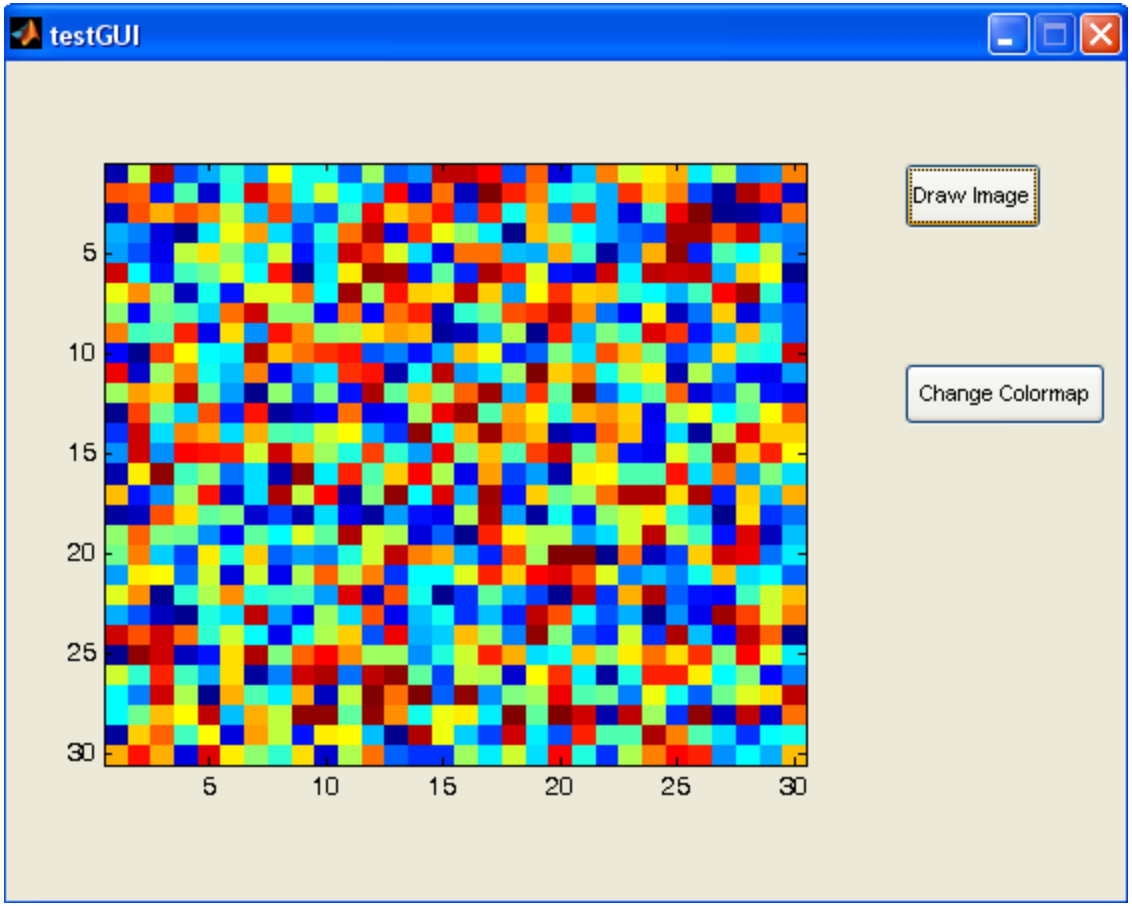
```
75
76      % --- Executes on button press in drawimage.
77      function drawimage_Callback(hObject, eventdata, handles)
78      % hObject    handle to drawimage (see GCBO)
79      % eventdata  reserved - to be defined in a future version of MATLAB
80      % handles    structure with handles and user data (see GUIDATA)
81
82
83      % --- Executes on button press in changeColormap.
84      function changeColormap_Callback(hObject, eventdata, handles)
85      % hObject    handle to changeColormap (see GCBO)
86      % eventdata  reserved - to be defined in a future version of MATLAB
87      % handles    structure with handles and user data (see GUIDATA)
88
```

textFile.txt | numbers.txt | testGUI.m

testGUI

Running the GUI

- To run the GUI, just type its name in the command window and the GUI will pop up. The debugger is really helpful for writing GUIs because it lets you see inside the GUI



GUI Helper Functions

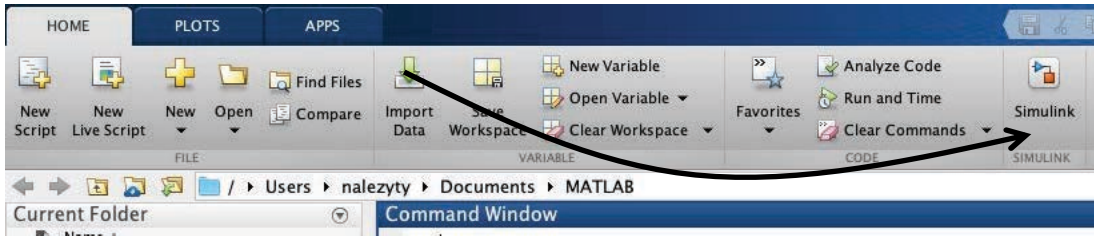
- Use keyboard to allow debugging from command window. GUI variables will appear in the workspace. Use return to exit debug mode
- Use built-in GUI modals for user input:
  - **`uigetfile`**
  - **`uiputfile`**
  - **`inputdlg`**
    - And more… (see help for details)

SIMULINK

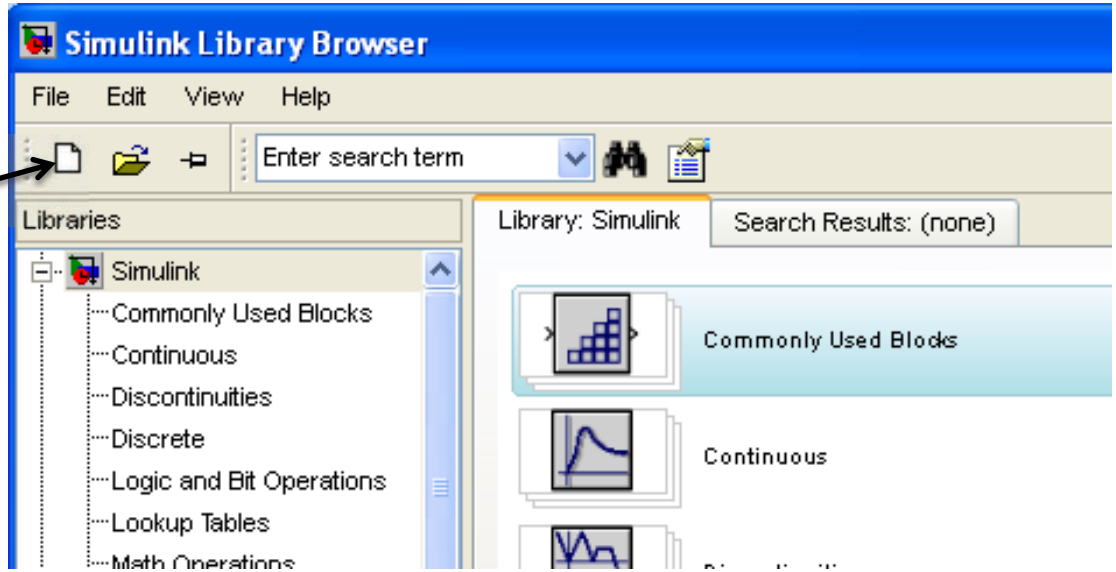
- Interactive graphical environment
- Block diagram based MATLAB add-on environment
- Design, simulate, implement, and test control, signal processing, communications, and other time-varying systems

Getting Started

- In MATLAB, Start Simulink

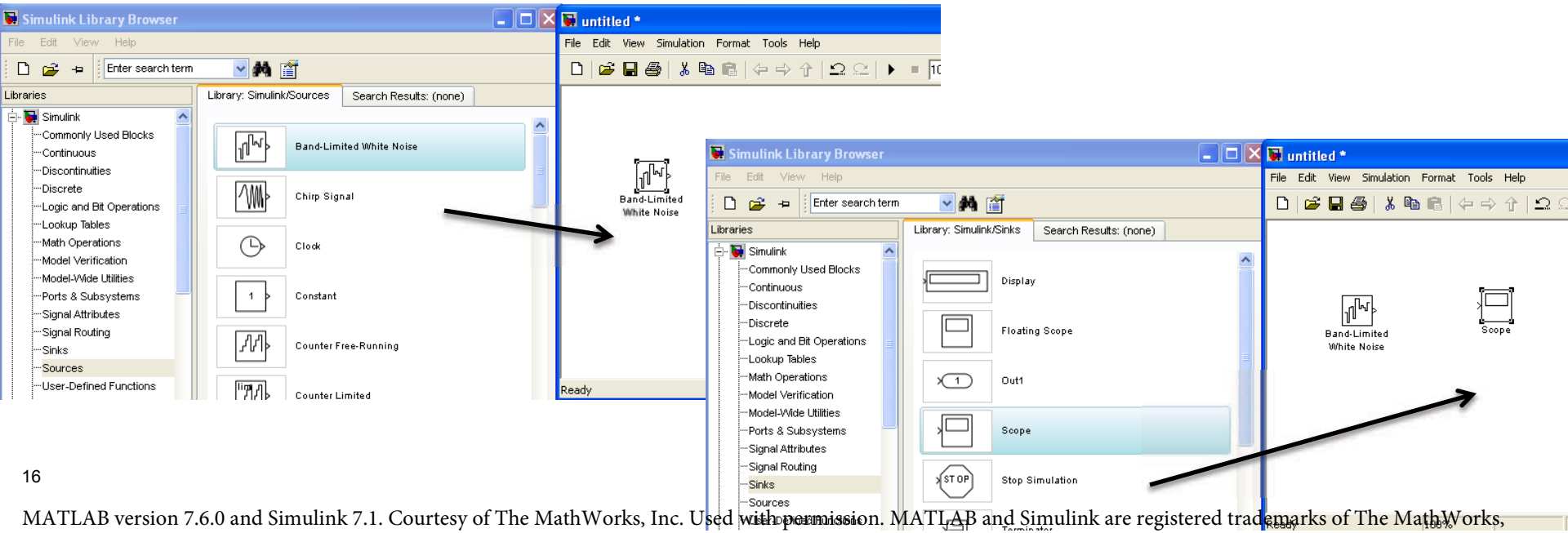


- Create a new Simulink file, similar to how you make a new script



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

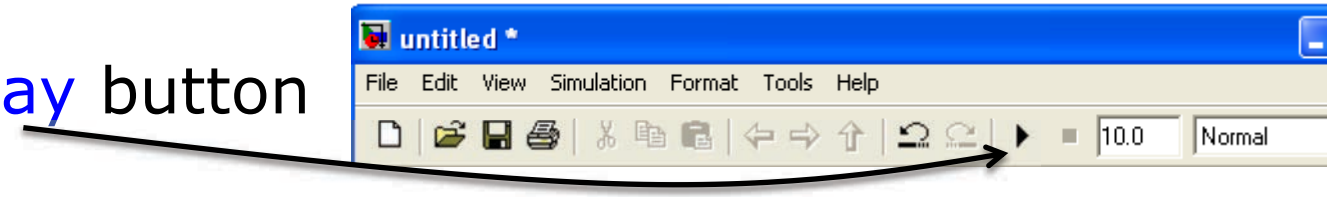


Connections

- Click on the carat/arrow on the right of the band limited white noise box

- Drag the line to the scope
  - You'll get a hint saying you can quickly connect blocks by hitting Ctrl
  - Connections between lines represent signals

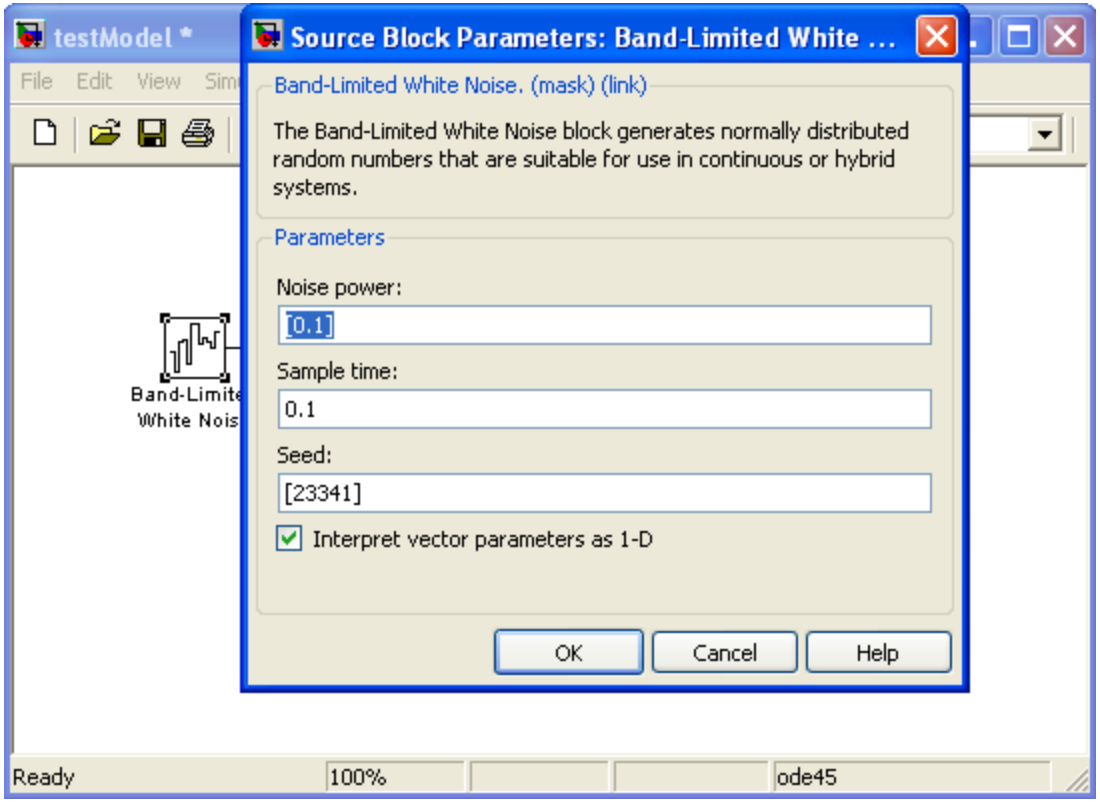
- Click the play button



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

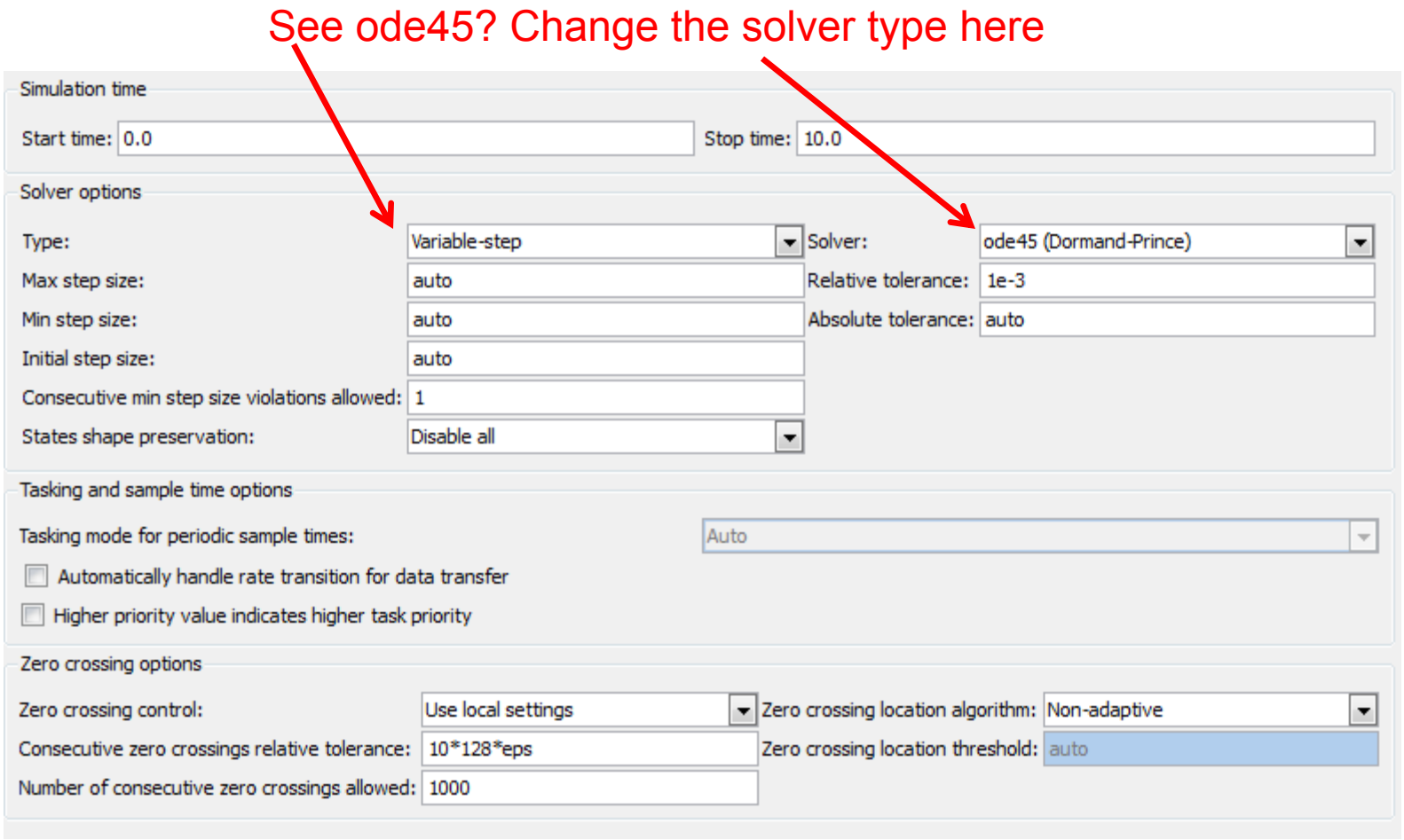
Connections, Block Specification

- 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.



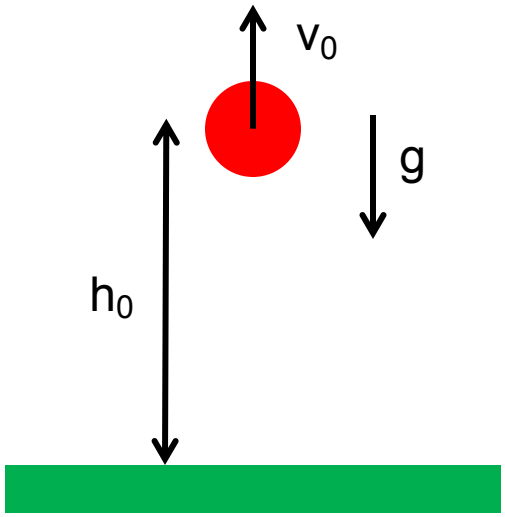
Behind the curtain

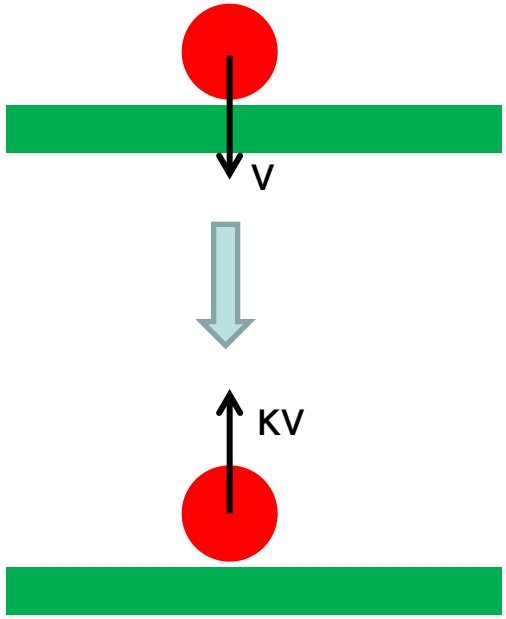
- Go to "Simulation"->"Configuration Parameters" at the top menu



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

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

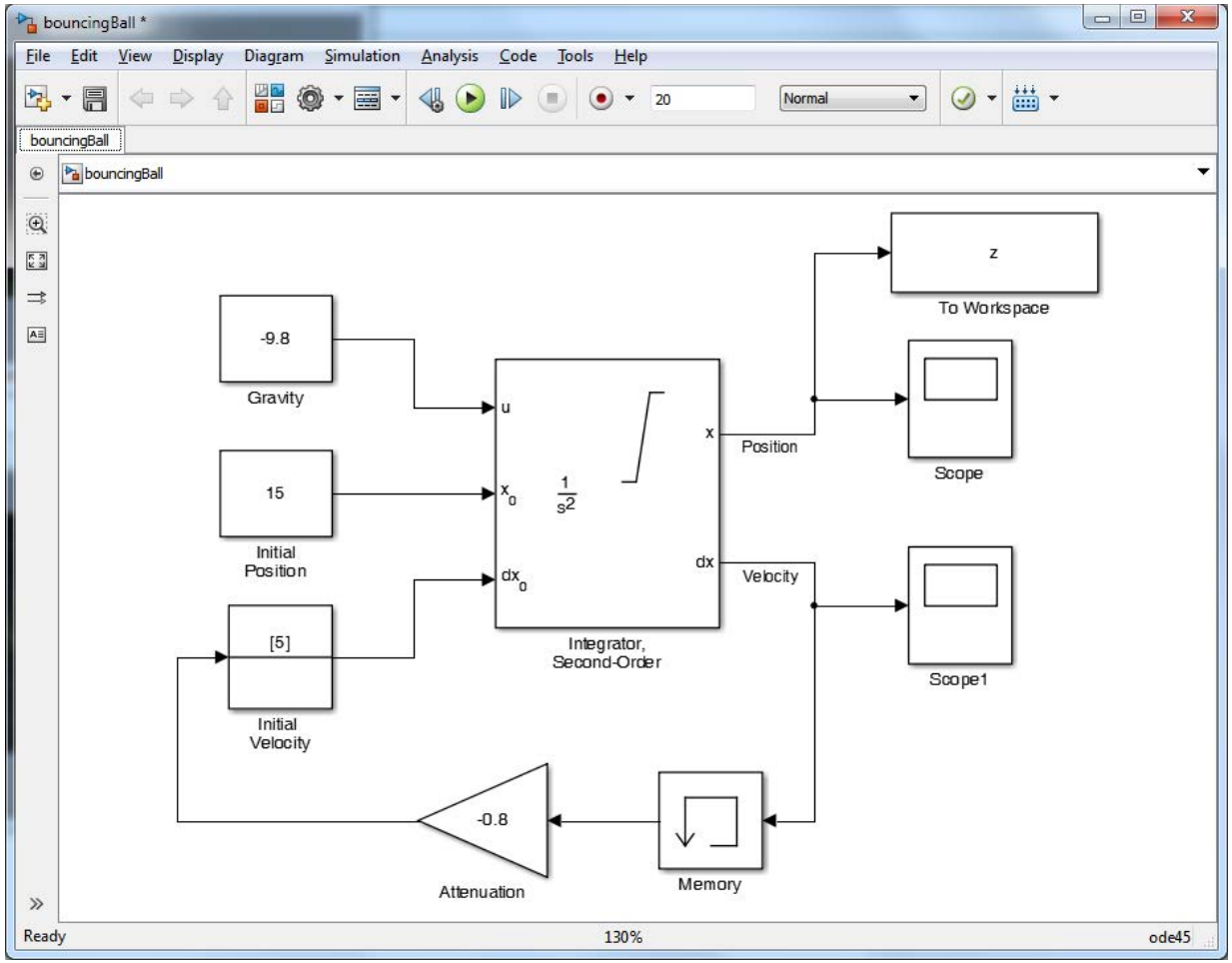
$$z(t=0) = h_0 \quad v(t=0) = v_0$$

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

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

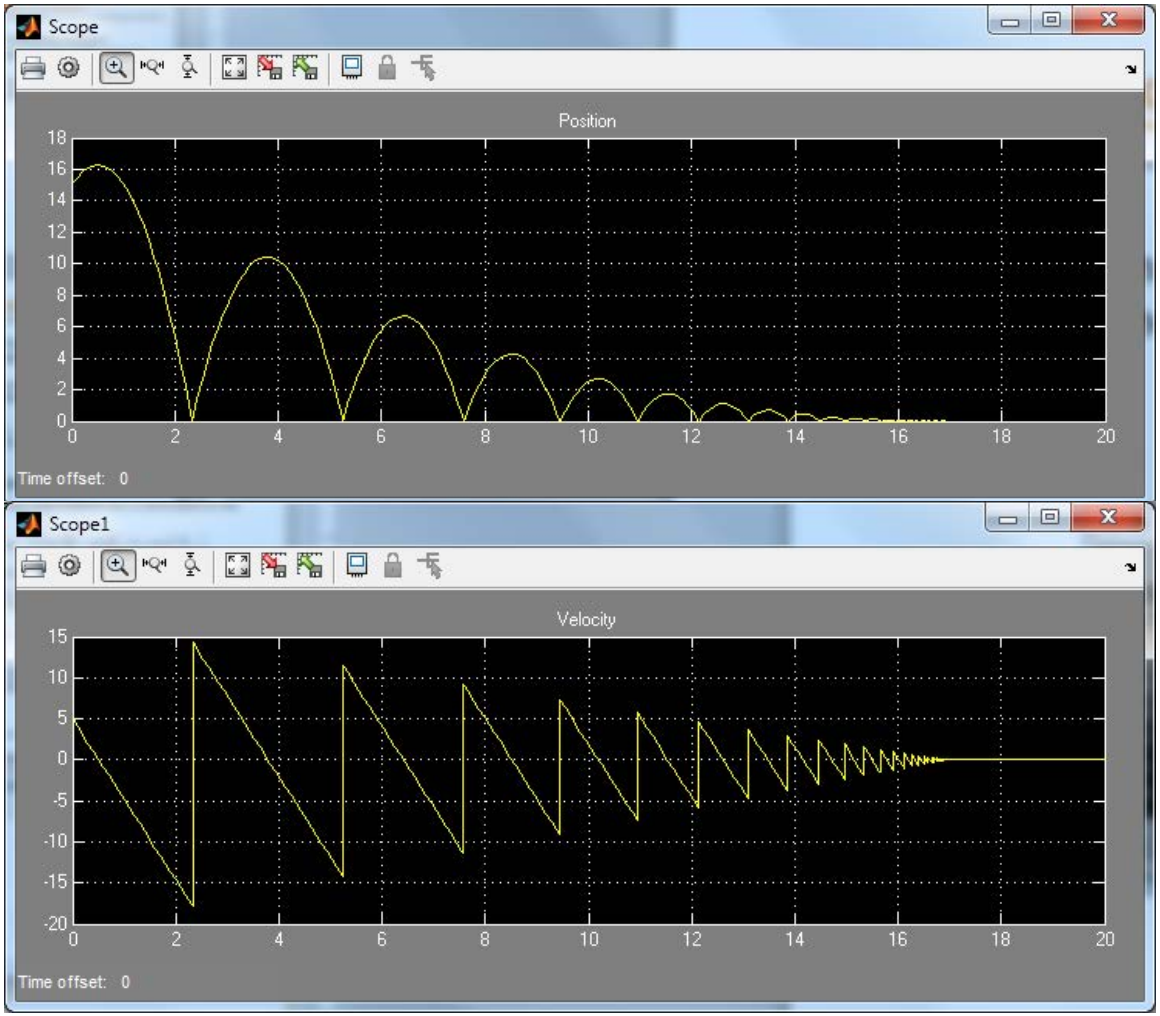
Exercise: Simulink Model

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



Exercise: Simulink Results

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



Toolboxes

- Math
  - Takes the signal and performs a math operation
  - » **`Add, subtract, round, multiply, gain, angle`**
- Continuous
  - Adds differential equations to the system
  - » **`Integrals, Derivatives, Transfer Functions, State Space`**
- Discontinuities
  - Adds nonlinearities to your system
- Discrete
  - Simulates discrete difference equations
  - Useful for digital systems

Building systems

- Sources
  - » **Step input, white noise, custom input, sine wave, ramp input,**
    - ➢ Provides input to your system
- Sinks
  - » **Scope: Outputs to plot**
  - » **simout: Outputs to a MATLAB vector (struct) on workspace**
  - » **Matlab mat file**

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');`
  - » `b=sym('4/5');`
  - » `mat=sym([1 2;3 4]);`
    - ➢ fractions remain as fractions
  - » `c=sym('c','positive');`
    - ➢ can add tags to narrow down scope
    - ➢ 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`** →
    ```
    d =
    4/15
    ```
    - ➢ does 1/3*4/5=4/15;
  - » **`expand((a-c)^2);`** →
    ```
    ans =
    1/9-2/3*c+c^2
    ```
    - ➢ multiplies out
  - » **`factor(ans)`** →
    ```
    ans =
    1/9*(3*c-1)^2
    ```
    - ➢ factors the expression
  - » **`pretty(ans)`** →
    ```
                2
    (3 c - 1)
    ----------
        9
    ```
    - ➢ makes it look nicer

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
```

» **`simplify(cos(x)^2+sin(x)^2)`** →

```
ans =
1
```

➢ simplifies expressions

» **`subs('c^2',c,5)`** →

```
ans =

    25
```

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

» **`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];` →
    - ➢ compute the product

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

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

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

  - » `i=inv(mat)` →
    - ➢ find the inverse

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

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

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

Exercise: Symbolics

- 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

- 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: $\int_a^b xe^x dx$

Exercise: Symbolics

- 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})
```

Image Processing

- http://www.mathworks.com/help/images/index.html

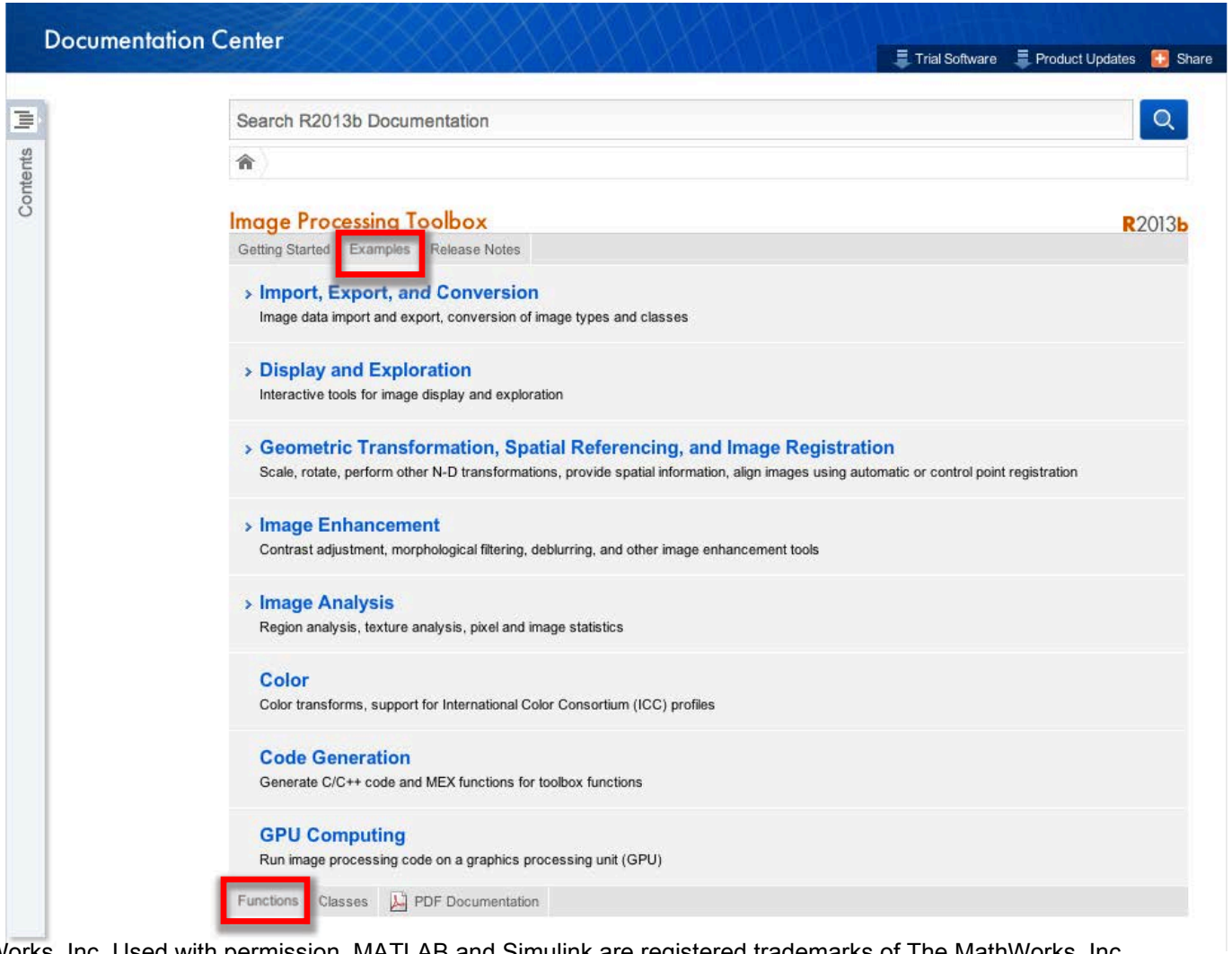


Image Processing

- Image enhancement
  - Adjust image contrast, intensities, etc.
- Filtering and deblurring
  - Convolution and deconvolution

- Finding edges
  - Image gradient, `edge`
- Finding circles
  - Hough transform

- Training an object detector
  - Computer vision toolbox: `trainCascadeObjectDetector`

Image Processing

- Image Restoration
  - Denoising
- Image Enhancement & Analysis
  - Contrast Improvement
    - imadjust, histeq, adapthisteq
  - Edge Detection
    - edge
  - Image Sharpening
  - Image Segmentation
- Image Compression
  - Wavelet toolbox (Chap. 3 of Gonzalez book on DIP)

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

```
»   % Loading the our image into the workspace
»   Image                   = imread('pout.tif');
»
»   % For comparison, it is better to have a predefined width
»   width                   = 200;
»
»   % Resizing the image using bicubic interpolation
»   dim                     = size(Image);
»   Image                   = imresize(Image , width * [dim(1) / dim(2) 1] , 'bicubic');
»
»   % Adjusting the contrast using imadjust
»   Image_imadjust          = imadjust(Image);
»
»   % Adjusting the contrast using histogram equalization
»   Image_histeq            = histeq(Image);
»
»   % Adjusting the contrast using adaptive histogram equalization
»   Image_adapthisteq       = adapthisteq(Image);
»
```

Exercise: Contrast Improvement

```
»   % Displaying the original image and the results in a single figure to compare with each other
»   figure
»   subplot(2 , 2 , 1);
»   imshow(Image);
»   title('Original Image');
»
»   subplot(2 , 2 , 2);
»   imshow(Image_imadjust);
»   title('Enhanced Image using Imadjust');
»
»   subplot(2 , 2 , 3);
»   imshow(Image_histeq);
»   title('Enhanced Image using Histeq');
»
»   subplot(2 , 2 , 4);
»   imshow(Image_adapthisteq);
»   title('Enhanced Image using Adapthisteq');
```

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.

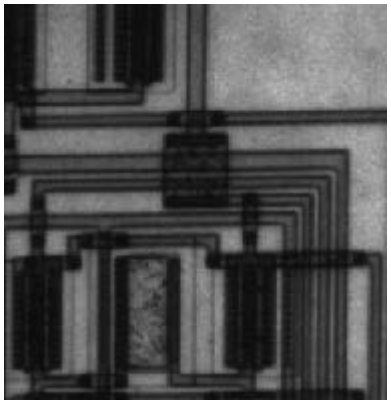
- First load the image "circuit.tif" and then plot the edges in that figure using the function **`edge`** and the filters **`"sobel"`**, **`"prewitt"`**. Also use **`"canny"`** as another method for edge detection using **`edge`**.

Exercise: Edge Detection

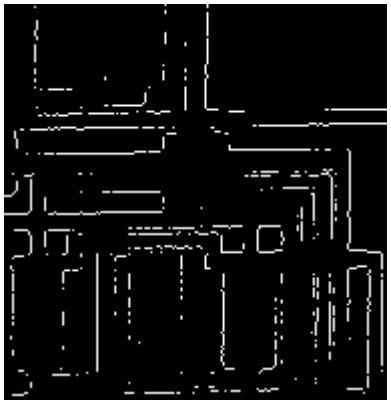
```
» I   = imread('circuit.tif');
» I1  = edge(I , 'sobel');
» I2  = edge(I , 'canny');
» I3  = edge(I , 'prewitt');
»
» figure
» subplot(2 , 2 , 1);
» imshow(I);
» title('Original Image');
»
» subplot(2 , 2 , 2);
» imshow(I1);
» title('Edges found using sobel filter');
»
» subplot(2 , 2 , 3);
» imshow(I2);
» title('Edges found using the "canny" method');
»
» subplot(2 , 2 , 4);
» imshow(I3);
» title('Edges found using prewitt filter');
```

Exercise: Edge Detection

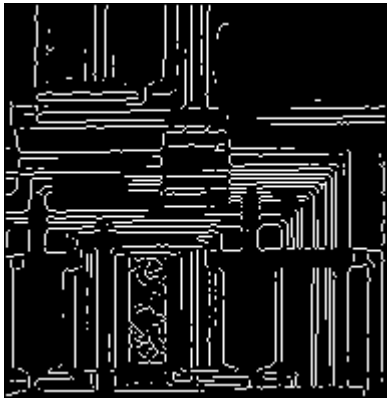
Original Image



Edges found using sobel filter



Edges found using the "canny" method



Edges found using prewitt filter

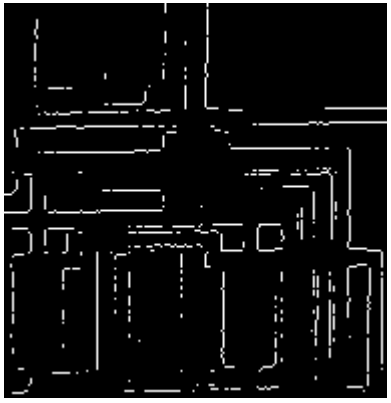


Image Enhancement

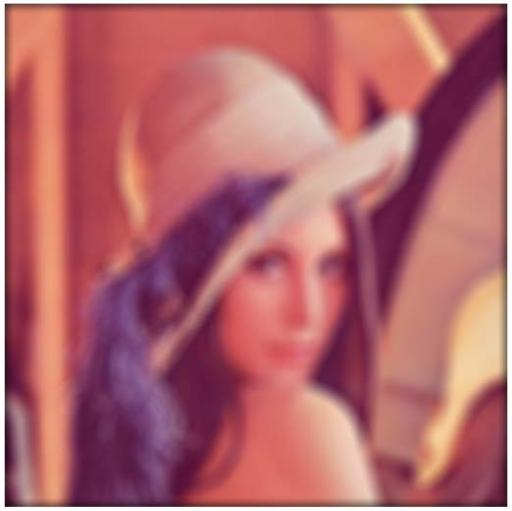
- Commonly-used: **`imread`**, **`imwrite`**, **`imshow`**, **`imresize`**
  - » `im = imread('pout.tif');`
    `% image included in toolbox`
  - » `imtool(im);`
    - ➢ Convenient for editing in figure window

- 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

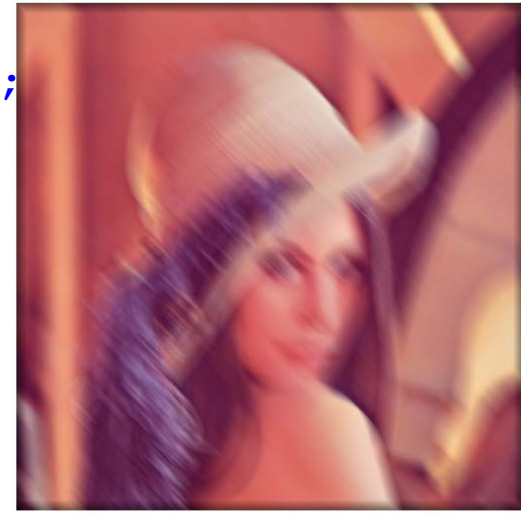
Pillbox filter:

```
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

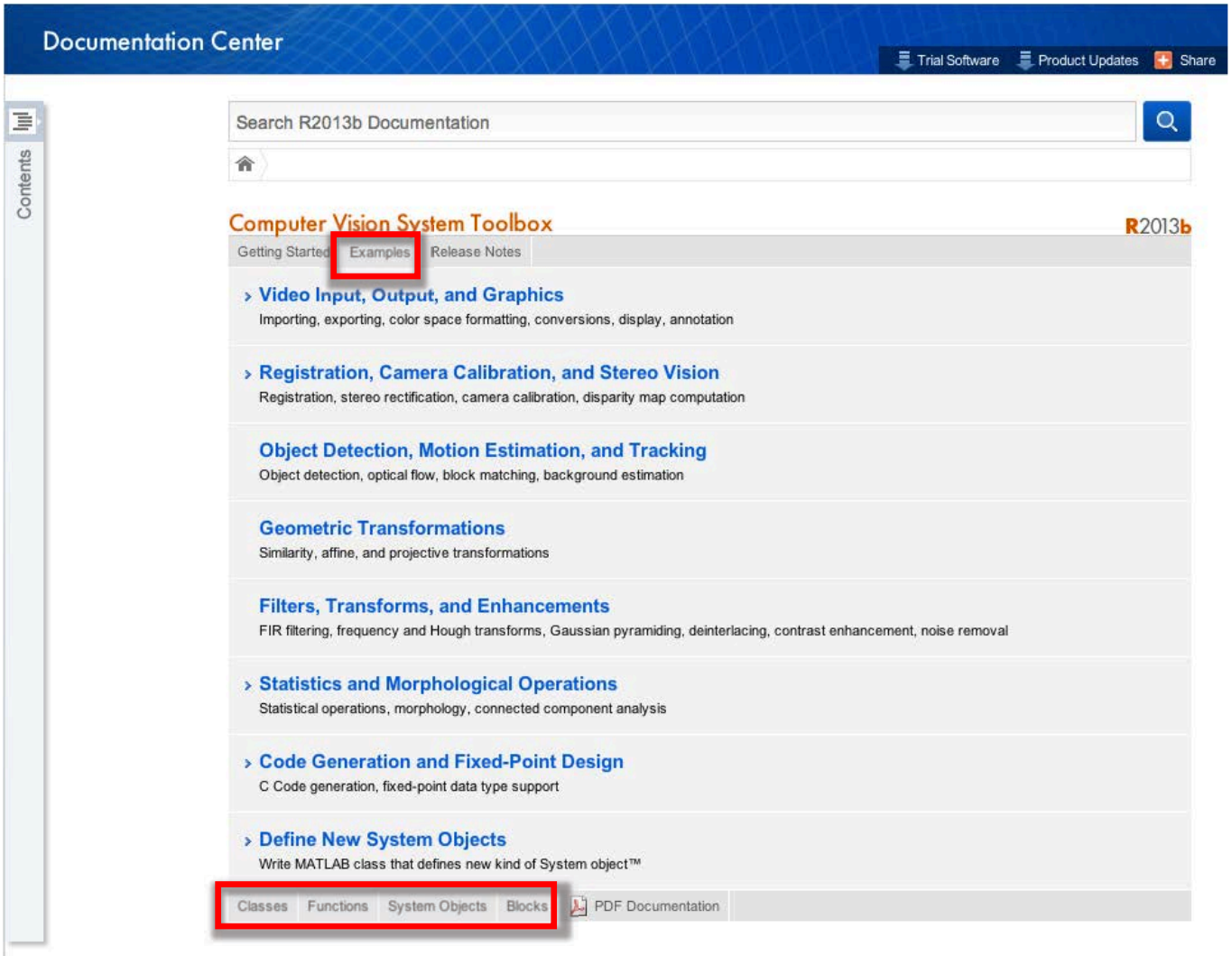
Image Analysis

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

… and also Computer Vision

- http://www.mathworks.com/help/vision/index.html



… and also Computer Vision

- http://www.mathworks.com/help/vision/functionlist.html

Feature Detection, Extraction, and Matching

detectFASTFeatures	Find corners using FAST algorithm
detectHarrisFeatures	Find corners using Harris–Stephens algorithm
detectMinEigenFeatures	Find corners using minimum eigenvalue algorithm
detectMSERFeatures	Detect MSER features
detectSURFFeatures	Detect SURF features
extractFeatures	Extract interest point descriptors
extractHOGFeatures	Extract Histograms of Oriented Gradients (HOG) features
matchFeatures	Find matching features
showMatchedFeatures	Display corresponding feature points
binaryFeatures	Object for storing binary feature vectors
cornerPoints	Object for storing corner points
SURFPoints	Object for storing SURF interest points
MSERRegions	Object for storing MSER regions
vision.BoundaryTracer	Trace object boundary
vision.CornerDetector	Detect corner features
vision.EdgeDetector	Find object edge

Object Detection, Motion Estimation, and Tracking

configureKalmanFilter	Create Kalman filter for object tracking
disparity	Disparity map between stereo images
trainCascadeObjectDetector	Train cascade object detector model
detectFASTFeatures	Find corners using FAST algorithm
detectHarrisFeatures	Find corners using Harris–Stephens algorithm
detectMinEigenFeatures	Find corners using minimum eigenvalue algorithm
detectMSERFeatures	Detect MSER features
detectSURFFeatures	Detect SURF features
extractFeatures	Extract interest point descriptors
extractHOGFeatures	Extract Histograms of Oriented Gradients (HOG) features
insertObjectAnnotation	Annotate truecolor or grayscale image or video stream
assignDetectionsToTracks	Assign detections to tracks for multiobject tracking
matchFeatures	Find matching features
cornerPoints	Object for storing corner points
SURFPoints	Object for storing SURF interest points
MSERRegions	Object for storing MSER regions
vision.KalmanFilter	Kalman filter for object tracking
vision.BlockMatcher	Estimate motion between images or video frames
vision.CascadeObjectDetector	Detect objects using the Viola-Jones algorithm
vision.ForegroundDetector	Detects foreground using Gaussian mixture models
vision.HistogramBasedTracker	Histogram-based object tracking
vision.OpticalFlow	Estimate object velocities
vision.PeopleDetector	Detect upright people using HOG features
vision.PointTracker	Track points in video using Kanade-Lucas-Tomasi (KLT) algorithm
vision.TemplateMatcher	Locate template in image

Also consider OpenCV+MATLAB
http://www.mathworks.com/discovery/matlab-opencv.html

Object Detection

- Train a cascade object detector (introduced in R2013a)
- http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html
- http://www.mathworks.com/help/vision/ref/traincascadeobjectdetector.html

- Inputs to **`trainCascadeObjectDetector`**:
  - Image files with bounding boxes for positive instances
  - Image files of negative instances ('background')
  - Optional: FP/TP rates, # cascade stages, feature type
- Output: An XML file with object detector parameters

```
» detector=vision.CascadeObjectDetector('my.xml');
```

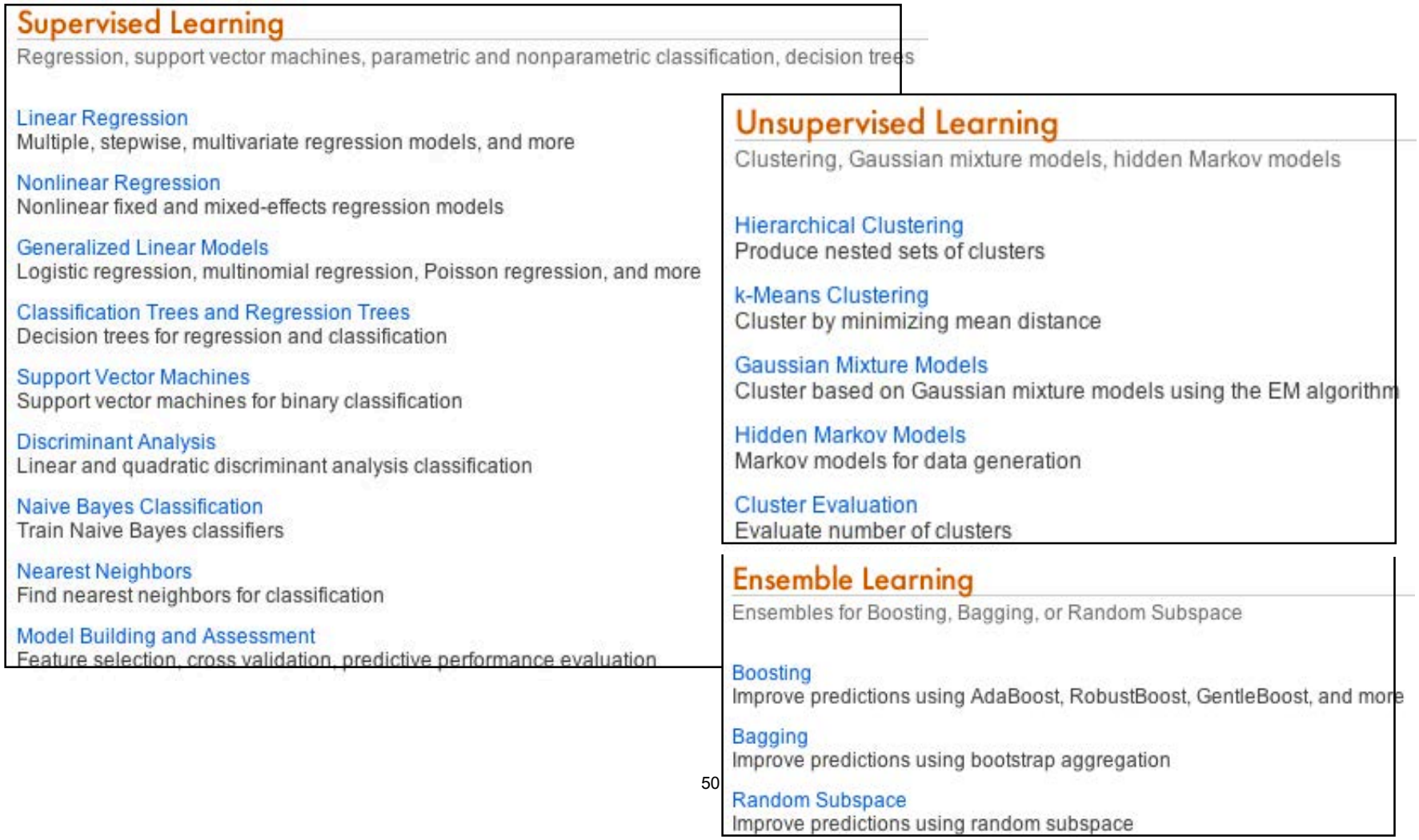
- Use the detector on new images:

```
» bbox=step(detector, imread('testImage.jpg'));
```

- See links above for full example

Machine Learning (Stats Toolbox)

- http://www.mathworks.com/help/stats/index.html



Supervised Learning

Regression, support vector machines, parametric and nonparametric classification, decision trees

Linear Regression
Multiple, stepwise, multivariate regression models, and more

Nonlinear Regression
Nonlinear fixed and mixed-effects regression models

Generalized Linear Models
Logistic regression, multinomial regression, Poisson regression, and more

Classification Trees and Regression Trees
Decision trees for regression and classification

Support Vector Machines
Support vector machines for binary classification

Discriminant Analysis
Linear and quadratic discriminant analysis classification

Naive Bayes Classification
Train Naive Bayes classifiers

Nearest Neighbors
Find nearest neighbors for classification

Model Building and Assessment
Feature selection, cross validation, predictive performance evaluation

Unsupervised Learning

Clustering, Gaussian mixture models, hidden Markov models

Hierarchical Clustering
Produce nested sets of clusters

k-Means Clustering
Cluster by minimizing mean distance

Gaussian Mixture Models
Cluster based on Gaussian mixture models using the EM algorithm

Hidden Markov Models
Markov models for data generation

Cluster Evaluation
Evaluate number of clusters

Ensemble Learning

Ensembles for Boosting, Bagging, or Random Subspace

Boosting
Improve predictions using AdaBoost, RobustBoost, GentleBoost, and more

Bagging
Improve predictions using bootstrap aggregation

Random Subspace
Improve predictions using random subspace

Hardware Interface

- Matlab can interact directly with many forms of external hardware, from lab equipment to standalone micro-controllers
- Interaction can be done at various levels of abstraction
- Ideal when processor intensive DSP is required and target system cannot handle it on it’s own
- Probably not suitable for real-time systems due to the communication overhead

Low Level

- Most basic link – through the serial port using **`serial`**
  - **`s = serial('com3')`**
    - Can also provide additional properties, see **`help serial`**
- From here on, treat **`s`** as a file handler
  - **`fopen(s)`**
  - **`fwrite(s, data)`**
  - **`fprintf(s, 'string');`**
  - **`res = fscanf(s);`**
- Don't forget to close!
  - **`fclose(s);`**

GPIB

- GPIB – General Purpose Interface Bus (IEEE-488)
- Created by HP in the 1960's, but highly adopted today in many lab instruments
- A standardized communication protocol for sending and receiving information
- Simply create using the command gpib
  - » `g = gpib('agilent', 7, 1);`
    - ➢ See `help gpib` for option details
    - ➢ From now on, treat as file handler
  - » `fopen(g);`
  - » `fprintf(g, '*IDN?')`
  - » `idn = fscanf(g);`
- Don't forget to close!
  - » `fclose(g);`