Introduction to Digital Image Processing

Ranga Rodrigo

November 13, 2011

1. Introduction
2. Digital Images
3. Matlab or Octave Tutorial
4. Programming in Matlab
   - Flow Control
   - Scripts and Functions
   - Other Data Structures

Outline

What Is Digital Image Processing?

- Digital image processing helps us enhance images to make them visually pleasing, or accentuate regions or features of an image to better represent the content.
- For example, we may wish to enhance the brightness and contrast to make a better print of a photograph, similar to popular photo-processing software.
- In a magnetic resonance image (MRI) of the brain, we may want to accentuate a certain region of image intensities to see certain parts of the brain.

- Image analysis and computer vision, which go beyond image processing, helps us to make decisions based on the contents of the image.
- This course, vision for automation, will give you the basic knowledge required to enter into this exciting field, and equip you with basic tools to do image processing and computer vision and apply the knowledge in automation.

What Is Computer Vision?

- The goal is the emulation of the visual capability of human beings using computers.

What Is Computer Vision?

- The goal is the emulation of the visual capability of human beings using computers.
- In other words, computer vision is making the machine see as we do!

What Is Computer Vision?

- The goal is the emulation of the visual capability of human beings using computers.
- In other words, computer vision is making the machine see as we do!
- It is challenging.

What Is Computer Vision?

- The goal is the emulation of the visual capability of human beings using computers.
- In other words, computer vision is making the machine see as we do!
- It is challenging.
- Steps:
  1. Image acquisition
  2. Image manipulation
  3. Image understanding
  4. Decision making

Main Driving Technologies

- Signal processing.
- Multiple view geometry.
- Optimization.
- Pattern recognition and machine learning.
- Hardware and algorithms.

Applications [1]

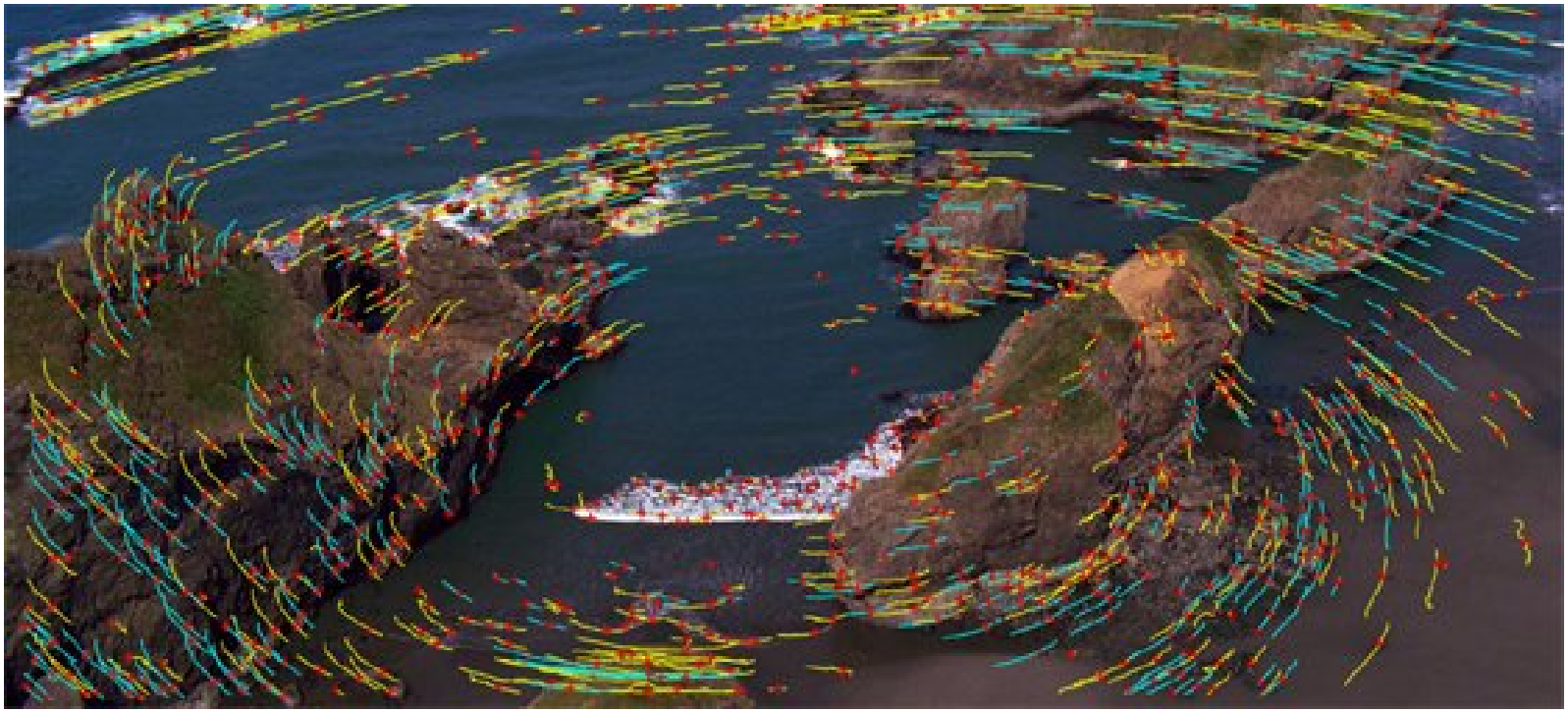
- Automotive:
  - Lane departure warning systems.
  - Head tracking systems for drowsiness detection.
  - Driver assistance systems.
  - Reading automobile license plates, and traffic management.
- Photography:
  - In camera face detection [2], red eye removal, and other functions.
  - Automatic panorama stitching [1].

[1] (From http://www.cs.ubc.ca/spider/lowe/vision.html)

Applications

- Movie and video (a very big industry):
  - Augmented reality.
  - Tracking objects in video or film and solving for 3-D motion to allow for precise augmentation with 3-D computer graphics.
  - Multiple cameras to precisely track tennis and cricket balls.
  - Human expression recognition.
  - Software for 3-D visualization for sports broadcasting and analysis.
  - Tracking consistent regions in video and insert virtual advertising.
  - Tracking for character animation.
  - Motion capture, camera tracking, panorama stitching, and building 3D models for movies.

Camera Tracking



Show 2d3 video.

Applications

- Games:
  - Tracking human gestures for playing games or interacting with computers.
  - Tracking the hand and body motions of players (to control the Sony Playstation).
  - Image-based rendering, vision for graphics.
- General purpose:
  - Inspection and localization tasks, people counting, biomedical, and security. etc.
  - Object recognition and navigation for mobile robotics, grocery retail, and recognition from cell phone cameras.
  - Laser-based 3D vision systems for use on the space shuttles and other applications.
  - Image retrieval based on content.

Applications

- Industrial automation (a very big industry):
  - Vision-guided robotics in the automotive industry.
  - Electronics inspection systems for component assembly.
- Medical and biomedical (maturing):
  - Vision to detect and track the pose of markers for surgical applications, needle insertion, and seed planting.
  - Teleoperations.
  - Quantitative analysis of medical imaging, including diagnosis such as cancer.
- Security and biometrics (thriving):
  - Intelligent video surveillance.
  - Biometric face, fingerprint, and iris recognition.
  - Behavior detection.

Minimal Invasive Surgery

Areas of Advancement

- Hardware.
- Image segmentation.
- 3-D reconstruction.
- Object detection.
- Navigation.
- Scene understanding.

Outline

What Is a Digital Image?

Definition

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity of gray level of f of the image at that point.

Definition

When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image.

Digital Image Representation

- A digital image can thus be treated as a 2-D array of integers. Let's denote a digital image as $f(i, j)$. The variables take following values:
  - $i \in [0, h-1]$, where h is the height of the image.
  - $j \in [10, w-1]$, where w is the width of the image.
  - $f(i, j) \in [0, L-1]$, where $L-1 = 255$ for an 8-bit image.
- We can formally write this as

$$f : [0, h-1] \times [0, w-1] \mapsto [0, L-1]. \tag{1}$$

- In Matlab or Octave indices i takes the values form 1 to h, and index j takes the values from 1 to w.

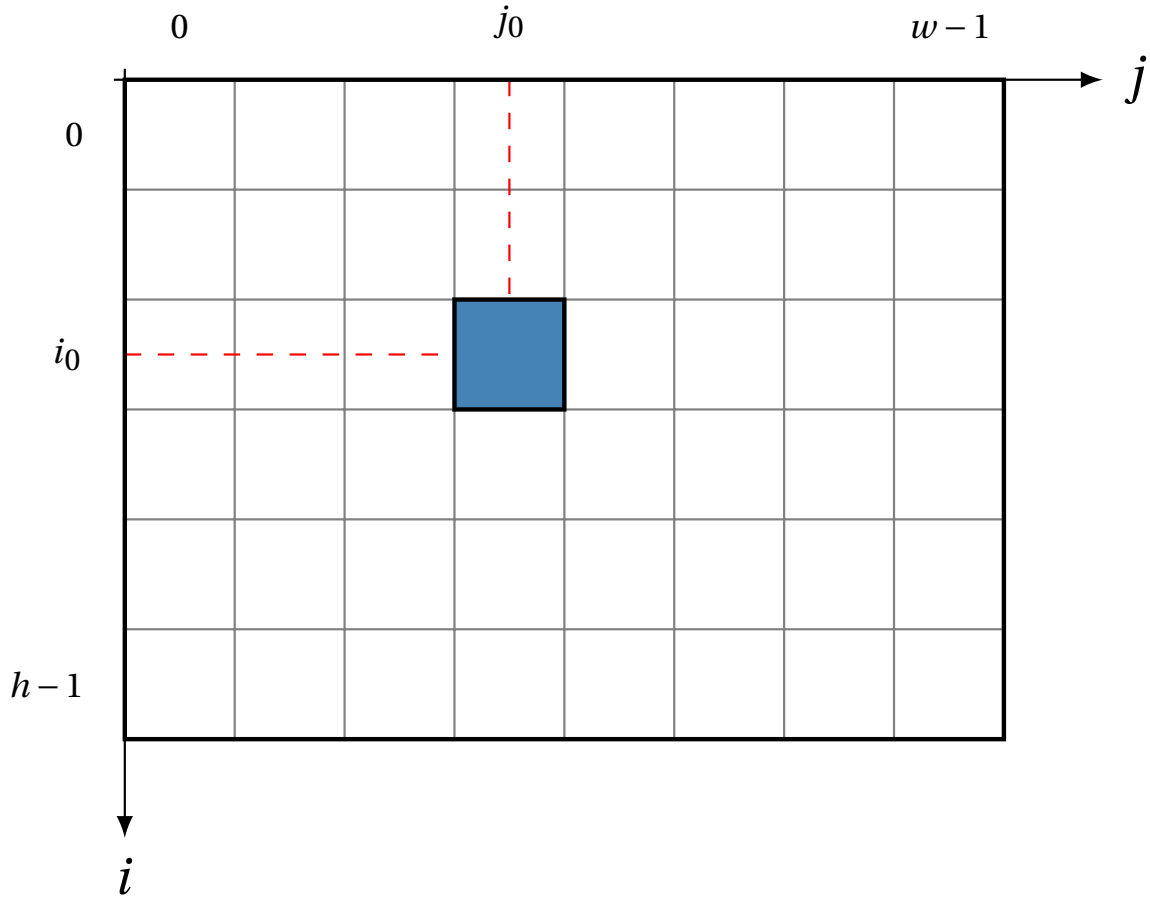


Figure 1: Digital image coordinates. Here, $i \in [0, h-1]$ and $j \in [0, w-1]$. In Matlab and Octave, $i \in [1, h]$ and $j \in [1, w]$. $f(i, j)$ is the value of the pixel at coordinate (i, j). For convenience, we say $f(i, j)$ is the values of pixel (i, j).

Pixels

- A digital images consists of a finite number of elements.
- These elements are referred to as picture elements, or pixels.
- A grayscale digital image can be seen as a two-dimensional array of pixels.
- $f(i_0, j_0)$ is the grayscale values of the pixel at coordinate (i_0, j_0).

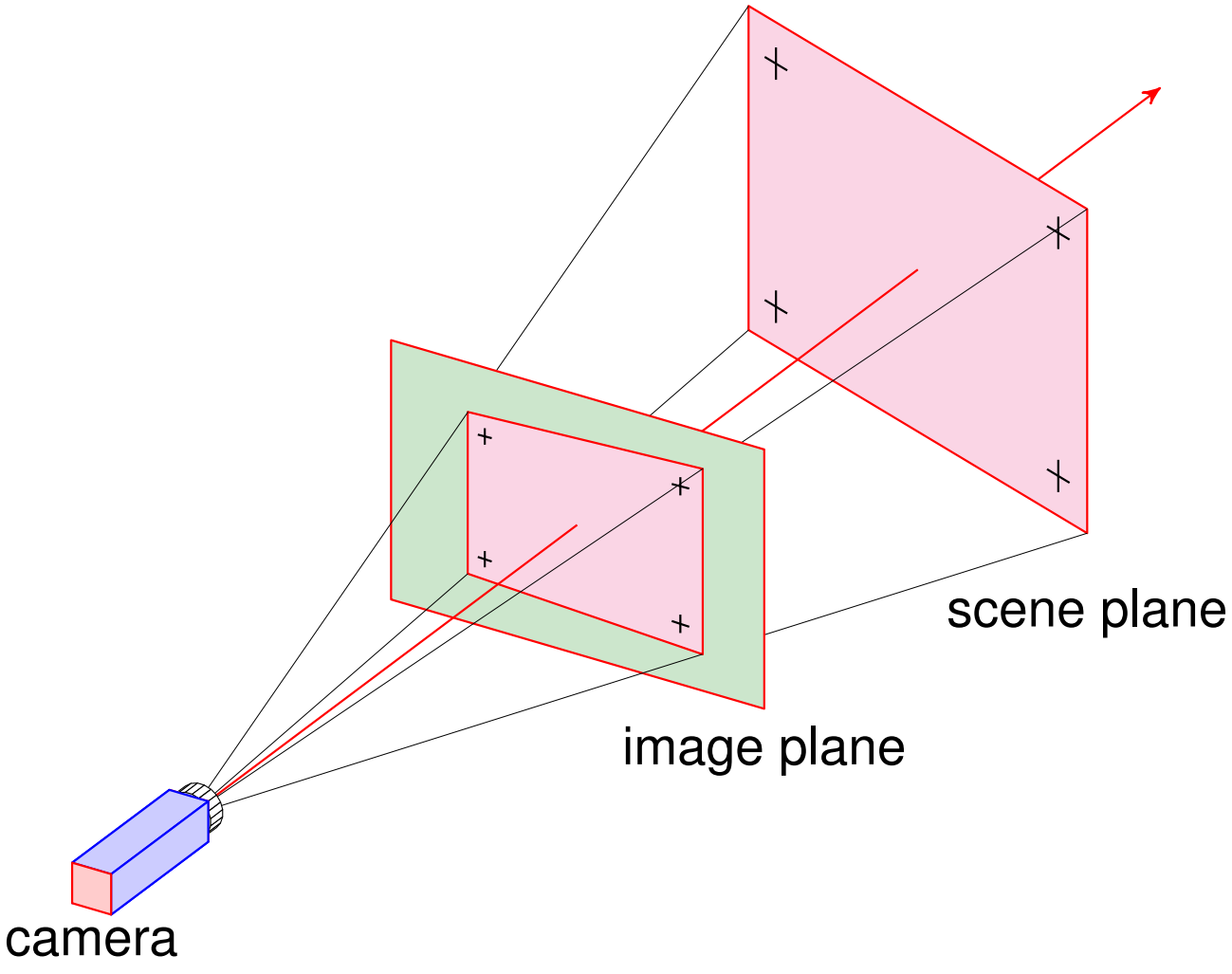
Color Images

- A color image can be represented using three functions $f_R(x, y)$, $f_G(x, y)$, and $f_B(x, y)$, representing the red, green and blue values respectively.
- We can interpret a digital color image as consisting of three two-dimensional arrays.

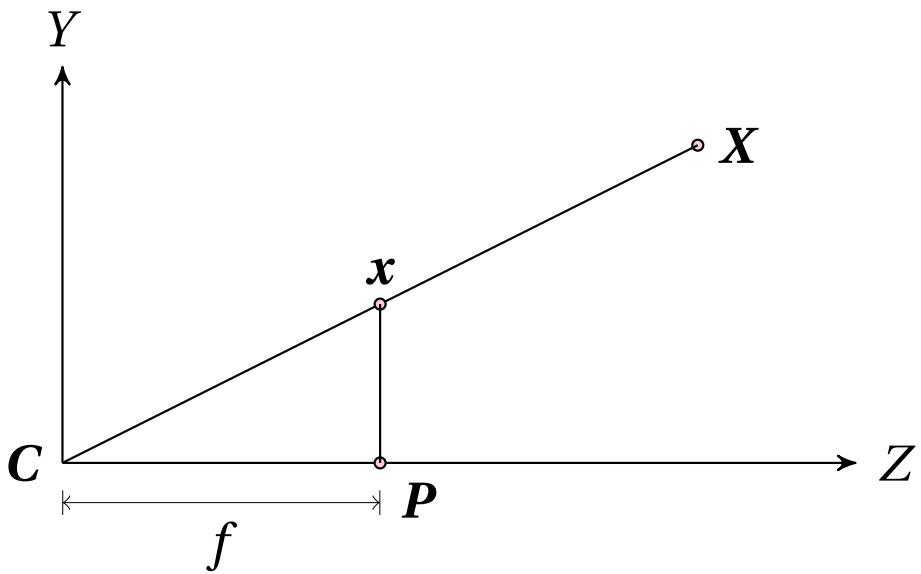
Imaging Modalities

- Optical.
- Other bands: infrared, radio.
- Gamma: PET (positron emission tomography).
- Ultrasound.
- X-ray and CT.
- MRI.

Image Formation



Camera Geometry



$\boldsymbol{C}$ is the camera center and $\boldsymbol{C}Z$ is the optical axis. $\boldsymbol{x}$ is the image of the world point $\boldsymbol{X}$. f is the focal length and $\boldsymbol{x}$ has image coordinates $(f\frac{X}{Z}, f\frac{Y}{Z})$ if the image center is $\boldsymbol{P}$ and image coordinate axes are X and Y.

Image Resolution and Dots Per Inch

- Resolution is the number of pixels in the image, e.g., 800×600.
- The same image with a particular resolution can be displayed with a certain number of dots per inch (DPI).
- If we display an 800×600 image at 100 DPI, the image will be 8 inches wide and 6 inches tall.
- Typically at least 300 DPI is needed for good print quality.

Figure 2 shows the effects of changing image resolution.

(a) 320×240, 300 DPI

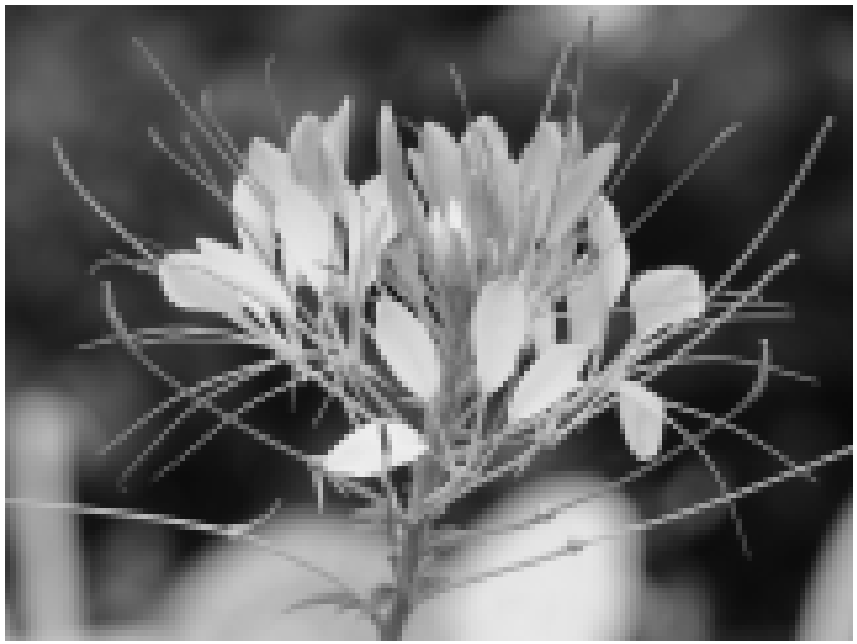
(b) 160×120, 300 DPI

(c) 80×60, 300 DPI

(d) 40×30, 300 DPI

Figure 2: Effect of resolution with the same DPI.

Figure 3 shows the effects of changing dots-per-inch value.



(a) 160×120, 150 DPI

(b) 160×120, 300 DPI

Figure 3: Effect of DPI with the same resolution.

Zooming Images

- If we have a 400×300 image and want a 600×450 image, we need to zoom.
- In order to zoom, we need interpolation.
  1. Nearest-neighbor.
  2. Bilinear.
  3. Cubic.

Image Processing, Analysis, and Computer Vision

- There is no clear cut boundary between these.
- One categorization is low-, mid-, and high-level processing.

Low-Level Processing

- Preprocessing to remove noise.
- Contrast enhancement.
- Image sharpening.

Mid-Level Processing

- Segmentation.
- Edge detection.
- Object extraction.

High-Level Processing

- Image analysis.
- Scene interpretation.

Outline

What Is Matlab?

- “The Matlab high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.”
- Therefore, Matlab is a powerful and friendly environment for image processing.

Toolboxes in Matlab

- Matlab helps us to do matrix manipulations. This is the basic functionality of Matlab. There are a large number of toolboxes built on top of this basic facility.
- Image Processing Toolbox is one of these toolboxes. However, we try to use the basic functionality and just minimally use the Image Processing Toolbox. This is because our aim is to be able to write our own image processing programs in Matlab.
- Octave is a free tool that can do a lot of tasks Matlab is capable of doing. Its toolboxes are not as comprehensive as Matlab, or it is not as user-friendly as Matlab.

Entering Matrices

In order to enter a matrix in Matlab or Octave command window, type the following: [2]

```
A = [1 2 3; 4 5 6; 7 8 9]
```

Matlab responds with the following:

```
A =
    1    2    3
    4    5    6
    7    8    9
```

If we add a semicolon, Matlab will not display the value of the variable `A`.

```
A = [1 2 3; 4 5 6; 7 8 9];
```

[2] In the following tutorial, I follow Matlab's Getting Started Guide. When I say Matlab, I refer to both Matlab and Octave.

We can sum the columns by using

```
A = [1 2 3; 4 5 6; 7 8 9];
sum(A)
```

Matlab responds with the following:

```
ans =
    12    15    18
```

The apostrophe operator (e.g., `A'`) performs transposition[3].

```
A = [1 2 3; 4 5 6; 7 8 9];
B = A'
```

Matlab responds with the following:

```
B =
     1     4     7
     2     5     8
     3     6     9
```

[3] Actually, it does the complex conjugate transposition.

Example

Write a Matlab program to obtain the row sums of matrix `A`.

We can do this by first transposing `A` and then obtaining the column sums.

```
A = [1 2 3; 4 5 6; 7 8 9];
B = A'
sum(B)'
```

We can do this by first transposing `A` and then obtaining the column sums.

```
A = [1 2 3; 4 5 6; 7 8 9];
B = A'
sum(B)'
```

We can also write this as

```
A = [1 2 3; 4 5 6; 7 8 9];
sum(A')'
```

Example

Write a Matlab program to obtain the sum of all the elements in `A`.

We can do this by first obtaining the column sums, and then summing the elements in the resulting vector.

```
A = [1 2 3; 4 5 6; 7 8 9];
c = sum(A);
sum(c)
```

We can do this by first obtaining the column sums, and then summing the elements in the resulting vector.

```
A = [1 2 3; 4 5 6; 7 8 9];
c = sum(A);
sum(c)
```

We can also write this as

```
A = [1 2 3; 4 5 6; 7 8 9];
sum(sum(A))
```

We can read the elements of the main diagonal by using the following:

```
A = [1 2 3; 4 5 6; 7 8 9];
diag(A)
```

Example

Write a Matlab program to obtain the sum of the diagonal of `A`.

We can do this by first obtaining the diagonal and then summing.

```
A = [1 2 3; 4 5 6; 7 8 9];
sum(diag(A))
```

The element in row i and column j of A is denoted by $\mathrm{A}(i, j)$. For example, A(1,2) is the number in the first row and second column.

Example

Write a Matlab program to extract the element in the second row, and third column, i.e, element at $(2,3)$.

```
A = [1 2 3; 4 5 6; 7 8 9];
A(2,3)
```

- It is also possible to refer to the elements of a matrix with a single subscript, $\mathrm{A}(k)$.
- This is the usual way of referencing row and column vectors. But it can also apply to a fully two-dimensional matrix, in which case the array is regarded as one long column vector formed from the columns of the original matrix.
- So, for our matrix, $\mathrm{A}(6)$ is another way of referring to the value 8 stored in $\mathrm{A}(3,2)$.

As A is a 3×3 matrix, the following code is illegal:

```
A = [1 2 3; 4 5 6; 7 8 9];
A(2,4)
```

Matlab will generate an error message such as “Index exceeds matrix dimensions.”

However, the following code is legal.

```
A = [1 2 3; 4 5 6; 7 8 9];
A(2,4) = 4
```

Matlab will enlarge A to accommodate a new column. The output is

```
A =
     1     2     3     0
     4     5     6     4
     7     8     9     0
```

.

The Colon Operator

The colon operator helps us to generates a row vector of numbers. For example

```
1:5
```

generates

```
1  2  3  4  5
```

and

```
100:-10:50
```

generates

```
100  90  80  70  60  50
```

Subscript expressions involving colons can refer to portions of a matrix. For example,

```
B = [1 2 3 4; 4 6 7 8; 9 10 11 12]
B(1:3,2)
```

gives

```
ans =
     2
     6
    10
```

Using the colon operator, we were able to access elements from 1 to 3 of column 2.

Example

Write a Matlab program to sum the elements of the second row of matrix B.

The solution is

```
B = [1 2 3 4; 4 6 7 8; 9 10 11 12]
sum(B(2,1:4))
```

The solution is

```
B = [1 2 3 4; 4 6 7 8; 9 10 11 12]
sum(B(2,1:4))
```

The following are alternative methods:

```
B = [1 2 3 4; 4 6 7 8; 9 10 11 12]
sum(B(2,1:end))
```

```
B = [1 2 3 4; 4 6 7 8; 9 10 11 12]
sum(B(2,:))
```

The keyword end refers to the last row or column.

M-Files

- M-files are text files containing the same statements we would type at the Matlab command line. We save the file with the extension .m.
- We can use Matlab's editor, Octave's editor, or even the Notepad to write an m-file.
- M-files can be run by typing the name of the m-file in the command window, or clicking the run button in the editor.
- This way, we can save our programs.

Numbers in Matlab

Some examples of legal numbers are

3	-99	0.0001
9.6397238	1.60210e-20	6.02252e23
1i	-3.14159j	3ee5i

Scientific notation uses the letter e to specify a power-of-ten scale factor.
Imaginary numbers use either i or j as a suffix.

Matlab Operators

+	Addition
−	Subtraction
*	Multiplication
/	Division
\	Left division (solving linear systems etc.)
^	Power
′	Complex conjugate transpose
()	Specify evaluation order

Matlab Functions

- There is a large number of standard elementary mathematical functions, including abs, sqrt, exp, and sin.
- For a list of the elementary mathematical functions, we can type help elfun. For a list of more advanced mathematical and matrix functions, we can type help specfun and help elmat[4].
- Some of the functions, like sqrt and sin, are built in. Built-in functions are part of the Matlab core so they are very efficient, but the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in m-files.

[4] Only for Matlab.

Several Special Functions

Several special functions provide values of useful constants.

pi	$3.14159265\ldots$
i	Imaginary unit $\sqrt{-1}$
j	Same as i
eps	Floating-point relative precision
realmin	Smallest floating-point number
realmax	Largest floating-point number
Inf	Infinity
NaN	Not-a-number

Expressions

- In Matlab, expressions are written similar to any other high level language. Here are the operators:

Example

Solve $x^2 + 2x + 3 = 0$ by using the quadratic formula.

```
a = 1;
b = 2;
c = 3;
delta = b^2 - 4*a*c;
x1 = (-b + sqrt(delta))/(2*a)
x2 = (-b - sqrt(delta))/(2*a)
```

```
a = 1;
b = 2;
c = 3;
delta = b^2 - 4*a*c;
x1 = (-b + sqrt(delta))/(2*a)
x2 = (-b - sqrt(delta))/(2*a)
```

The output is

```
x1 = -1.0000 + 1.4142i
x2 = -1.0000 - 1.4142i
```

Generating Matrices

- There are four function that generate basic matrices:
  1. zeros: All zeros
  2. ones: All ones
  3. rand: Uniformly distributed random elements
  4. randn: Normally distributed random elements

The following code generates a 2×3 matrix of 4s.

```
A = 4*ones(2,3)
```

The output is

```
A =
     4     4     4
     4     4     4
```

Concatenation

Concatenation is the process of joining small matrices to make bigger ones. For example, the code

```
A = ones(2,3)
B = zeros(2)
C = 2*ones(1,3)
D = 3*ones(1,2)
E = [A B; C D]
```

produces the following output:

A =

1	1	1
1	1	1

B =

0	0
0	0

C =

2	2	2

D =

3	3

E =

1	1	1	0	0
1	1	1	0	0
2	2	2	3	3

Deleting Rows and Columns

We can use a pair of empty brackets to delete rows and columns. For example, the following code deletes the second row of A:

```
A = [1 2 3; 4 5 6; 7 8 9];
A(2,:) = []
```

Deleting Rows and Columns

We can use a pair of empty brackets to delete rows and columns. For example, the following code deletes the second row of A:

```
A = [1 2 3; 4 5 6; 7 8 9];
A(2,:) = []
```

Example

Write a Matlab program to delete the last column of A.

```
A = [1 2 3; 4 5 6; 7 8 9];
A(:,end) = []
```

Example

Write a Matlab program to delete the first two rows of A.

```
A = [1 2 3; 4 5 6; 7 8 9];
A(1:2,:) = []
```

Linear Algebra Applications

Matrix multiplication:

```
A = [1 2 3; 4 5 6; 7 8 9]
B = [1 1; 2 2 ; 3 3]
C = A*B
```

```
A =
     1     2     3
     4     5     6
     7     8     9

B =
     1     1
     2     2
     3     3

C =
    14    14
    32    32
    50    50
```

Note that in matrix multiplication, as in $C = A \times B$, the number of columns in A and the number of rows in B must be equal.

det(A)	Determinant
inv(A)	Inverse
rref (A)	Reduced row echelon form
eig(A)	Eigenvalues
poly(A)	Coefficients of the characteristic polynomial

Operations on Arrays

$+$	Addition
$-$	Subtraction
$.*$	Element-by-element multiplication
$./$	Element-by-element division
$.\backslash$	Element-by-element left division
$.\hat{}$	Element-by-element power
$.'$	Unconjugated array transpose

Example

Write a Matlab program to generate a table of squares and cubes of integers from 1 to 10.

```
n = [1:10]'
table = [n n.^2 n.^3]
```

```
table =
     1     1     1
     2     4     8
     3     9    27
     4    16    64
     5    25   125
     6    36   216
     7    49   343
     8    64   512
     9    81   729
    10   100  1000
```

The find Function

The find function determines the indices of array elements that meet a given logical condition.

The find Function

The find function determines the indices of array elements that meet a given logical condition.

Example

Write a Matlab program to replace all the 5s in the following matrix by 0s.

```
A = [1 2 3 4 5; 5 6 3 9 1; 9 5 3 1 5]
```

```
A = [1 2 3 4 5; 5 6 3 9 1; 9 5 3 1 5]
indices = find(A == 5)
A(indices) = 0
```

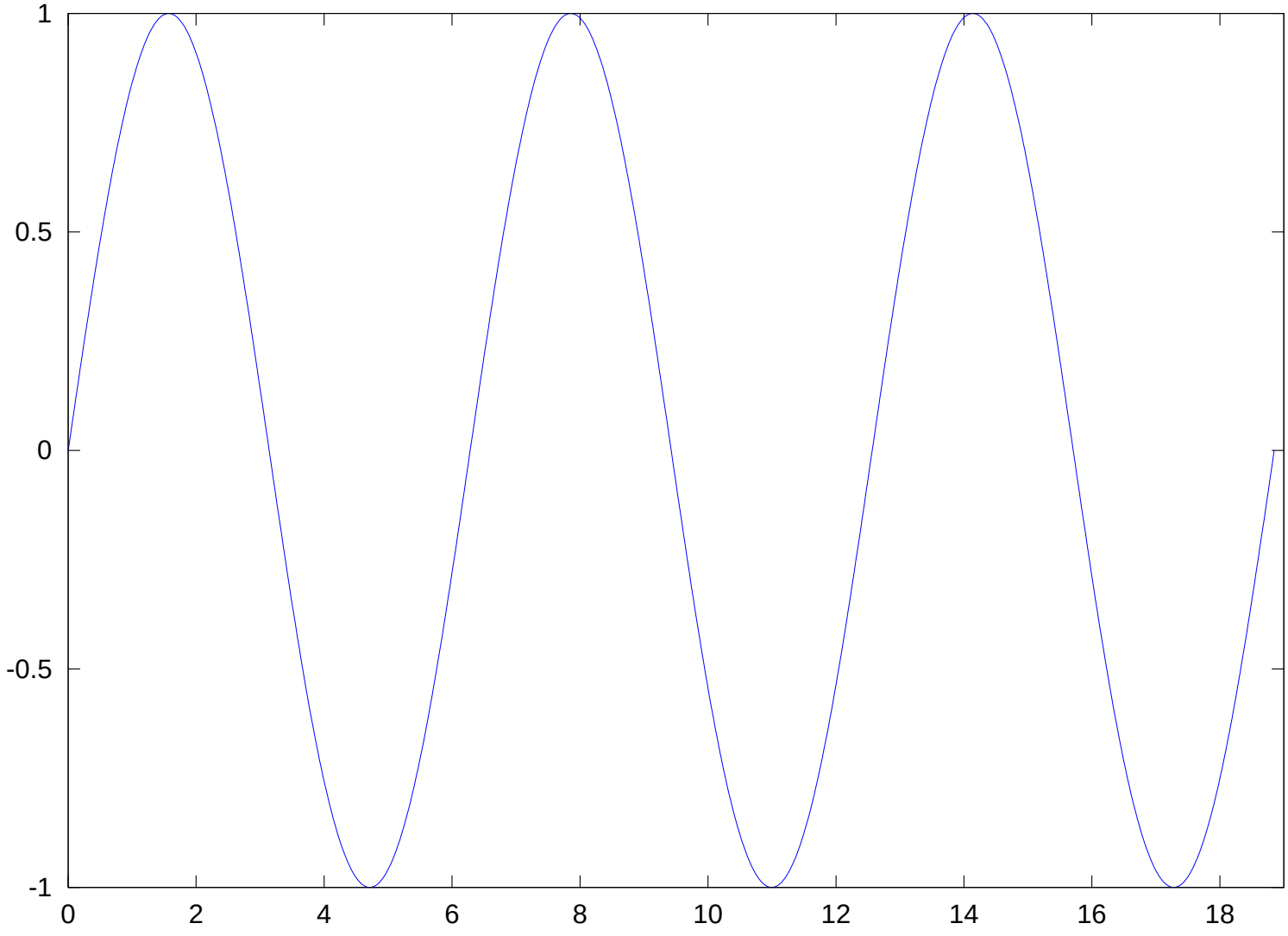
Alternatively,

```
A = [1 2 3 4 5; 5 6 3 9 1; 9 5 3 1 5]
A(find(A==5)) = 0
```

Plotting

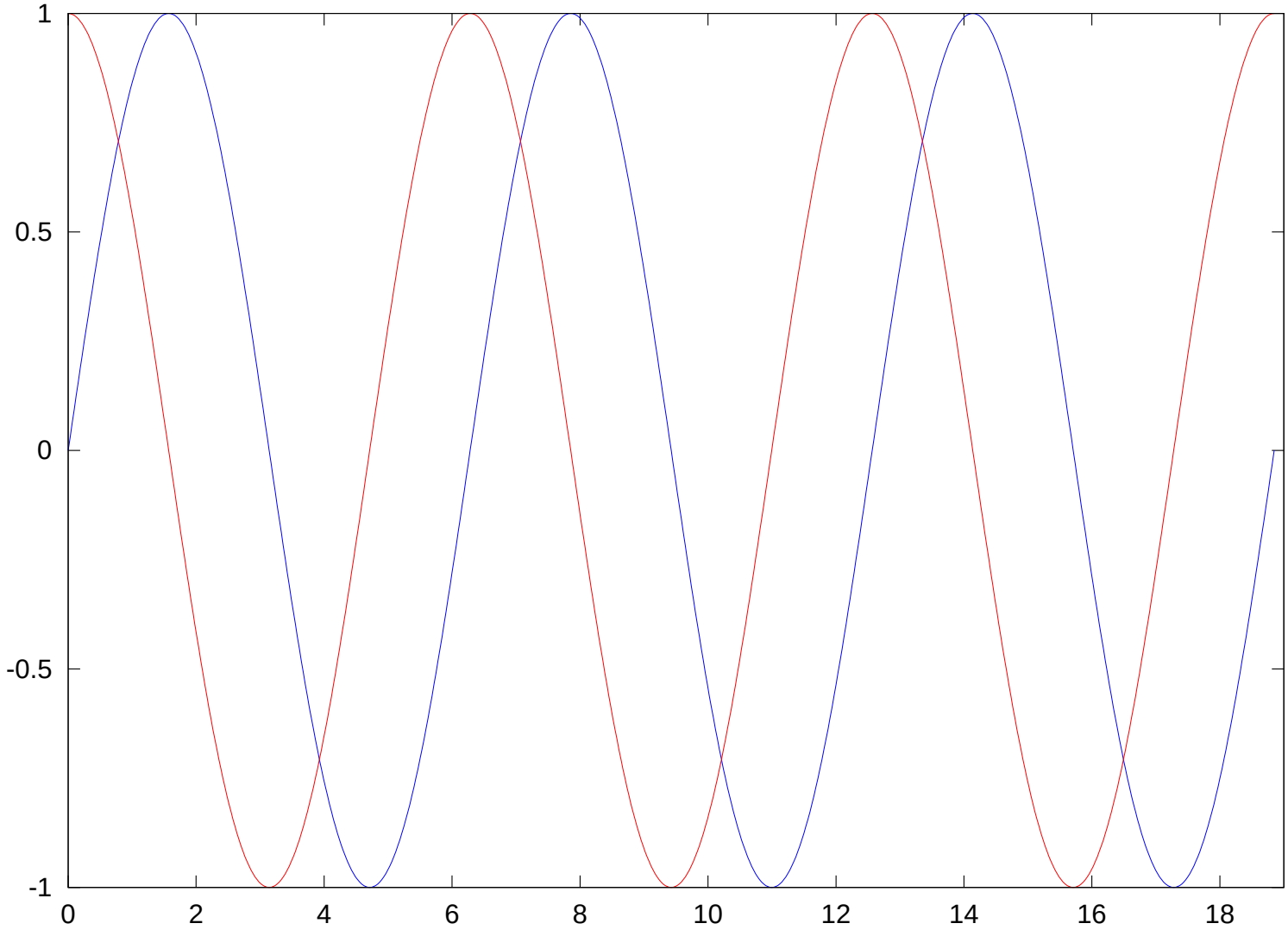
Matlab provides easy-to-use ways of displaying results. In the following example, we plot a sinusoid.

```
theta = 0:pi/100:6*pi;
y = sin(theta);
plot(theta, y)
```


1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14
16
18

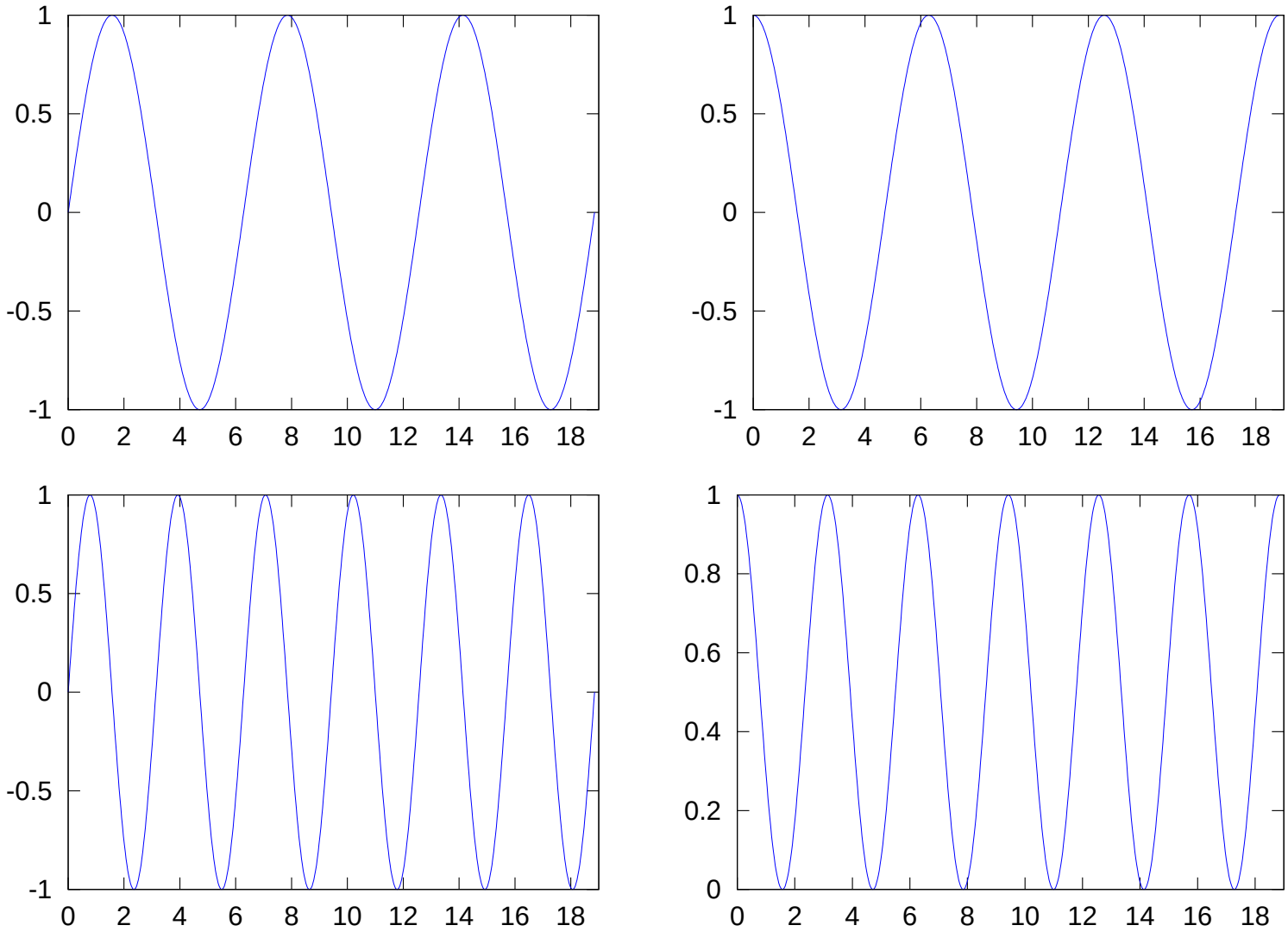
This example shows how to reuse an existing figure.

```
theta = 0:pi/100:6*pi;
y1 = sin(theta);
plot(theta, y1)
hold on
y2 = cos(theta);
plot(theta, y2, 'color', 'r')
hold off
```


1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14
16
18

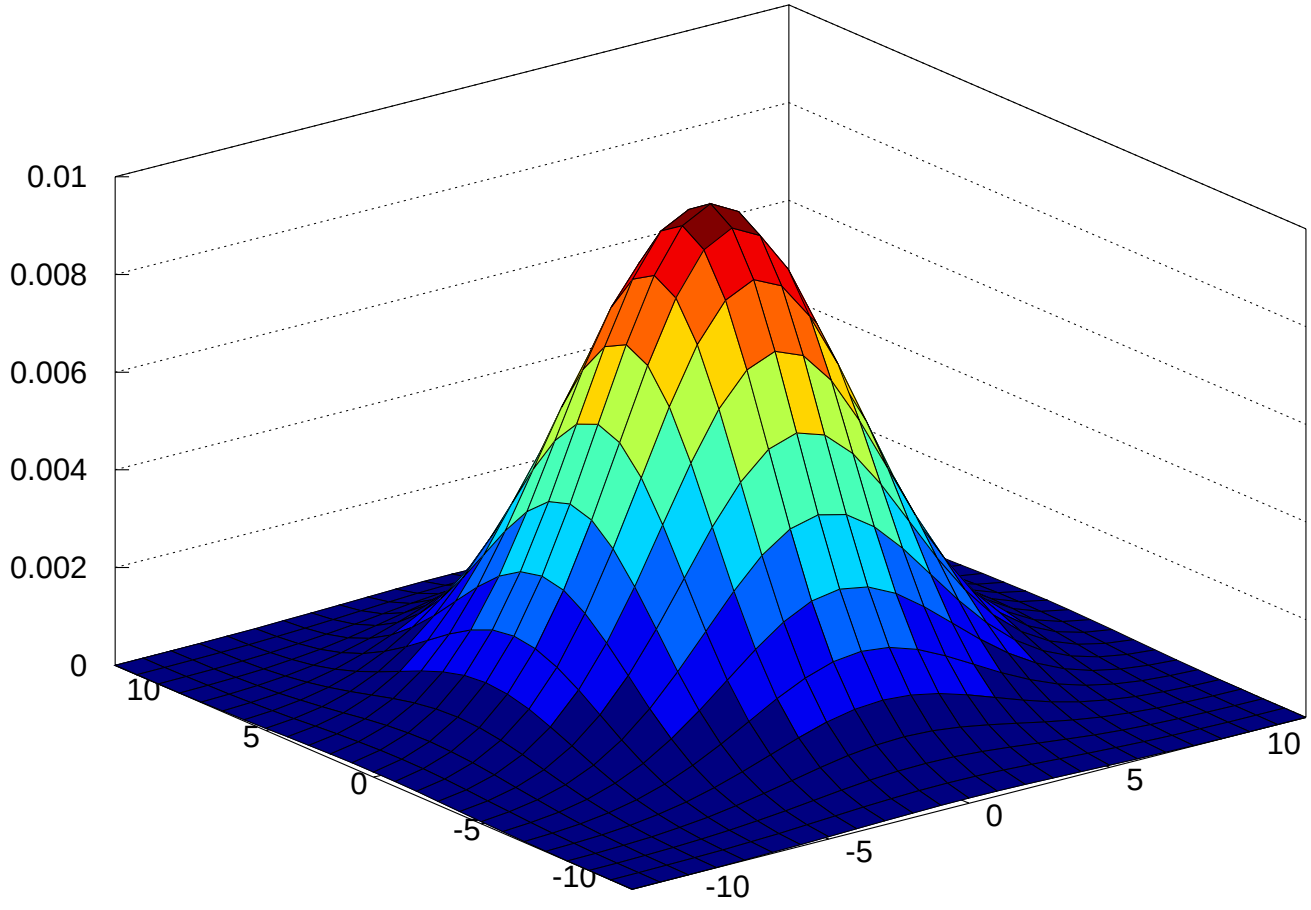
subplot is useful to display plots side-by-side.

```
theta = 0:pi/100:6*pi;
y1 = sin(theta);
plot(theta, y1)
y2 = cos(theta);
y3 = sin(2*theta);
y4 = cos(theta).*cos(theta);
subplot(2,2,1)
plot(theta, y1)
subplot(2,2,2)
plot(theta, y2)
subplot(2,2,3)
plot(theta, y3)
subplot(2,2,4)
plot(theta, y4)
```


1
0.5
0
-0.5
-1
0 2 4 6 8 10 12 14 16 18
1
0.5
0
-0.5
-1
0 2 4 6 8 10 12 14 16 18
1
0.5
0
-0.5
-1
0 2 4 6 8 10 12 14 16 18
1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18

A 3-D plot:

```
close all;
w = 25;
hw = floor(w/2);
u = linspace(-hw,hw,w);
v = linspace(-hw,hw,w);
[uu, vv] = meshgrid(u,v);
sigma = 4;
g = exp(-(uu.^2 + vv.^2)/(2*sigma^2));
g /= sum(g(:));
colormap(jet (10));
surf(uu,vv,g);
```


0.01
0.008
0.006
0.004
0.002
0
10
5
0
-5
-10
-10
-5
0
5
10

Outline

Language[5]

- Matlab's language is a matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.
- It allows both "programming in the small" to rapidly create quick programs, and "programming in the large" to create large and complex application programs.

[5] From Matlab's getting started guide.

- Major functionalities include
  1. Flow control
  2. Data structures
  3. Functions and scripts
  4. Object oriented programming

Outline

if , else, and elseif

The following code prints equal if variables A and B are equal.

```
A = ones(2);
B = A;
B(1,2) = 0;

if isequal(A,B)
  disp('A and B are equal')
else
  disp('A and B are not equal')
end
```

Here we check for the equality of two *matrices* not scalars. Therefore, we cannot use A==B.

The following code is meaningful for scalars A and B. For unequal matrices, Unexpected situation will be the output.

```
A = ones(2);
B = A;
B(1,2) = 0;


if A > B
  'greater'
elseif A < B
  'less'
elseif A == B
  'equal'
else
  error('Unexpected situation')
end
```

- Following functions are useful
  1. isequal
  2. isempty
  3. all
  4. any

switch

```
x = input("Input a choice (1 or 0): ")
switch x
case 0
  'You input 0.'
case 1
  'You input 1.'
otherwise
  error('Not a valid input.')
end
```

Unlike the C language switch statement, the Matlab switch does not fall through. If the first case statement is true, the other case statements do not execute. So, break statements are not required.

for

```
for i=1:10
    i^2
end
```

Example

Write a Matlab program print the multiplicative table form 1 to 12 using a for loops.

```
n = 12;
m = 12;
table = [];
for i=1:n
  row = [];
  for j=1:m
    row = [row i*j];
  end
  table = [table; row];
end
table
```

```
n = 12;
m = 12;
table = [];
for i=1:n
  row = [];
  for j=1:m
    row = [row i*j];
  end
  table = [table; row];
end
table
```

We can generate the table without using for loops. Loops are said to be slow in Matlab.

```
n = 12;
v = 1:n;
I = repmat(v', 1, n);
J = repmat(v, n, 1);
table = I.*J
```

Here is an example of printing odd number from 1 to 20.

```
i = 0;
while i < 20
    if mod(i,2)
        i
    end
    i++;
end
```

Example

Use Nowton-Raphson method to find the square-root of 625. According to Nowton-Raphson method, a better approximation of x_n, a root of $f(x)$ is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

In order to find the square-root of 625, we need to find x if $x^2 = 625$. Therefore, $f(x) = x^2 - 625$, and $f'(x) = 2x$.

```
error = Inf;
tolarence = 1e-5;
iterations = 100;

i = 0;
x0 = 10;
xn = x0;
while error > tolarence & i <= iterations
  i++;
  xnplus1 = xn - (xn^2 - 625)/(2*xn);
  error = abs(xnplus1 - xn)
  xn = xnplus1
end
i
error
xnplus1
```

Alternatively, roots([1,0,−625]) gives the result. $(f(x) = 1 \times x^2 + 0 \times x + (-625) \times 1)$.

Other constructs:

- continue passes control to the next iteration of a loop in which it appears, skipping any remaining statements in the body of the loop.
- break lets you exit early from a loop. In nested loops, break exits from the innermost loop only.
- return terminates the current sequence of commands and returns control to the invoking function or to the keyboard.
- try-catch-end provides error handling control.

Outline

M-Files

There are two kinds of M-files:

- Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace.
- Functions, which can accept input arguments and return output arguments. Internal variables are local to the function.

We have already used many scripts. Now, we study functions.

Functions

Here is an example of a function that computes the square of a given variable.

```
% Return the square of a scalar or element-by-element squa
function s = l01e20_square(x)
  s = x.^2;
return
```

We must save this with the file-name “l01e20_square.m”. We can call this function as, for example,

```
l01e20_square(5)
```

Functions

Here is an example of a function that returns the mean and the standard deviation of a set of numbers in a vector.

```
% Returns the mean and standard deviation of the elements in v
function [mu, sigma] = l01e21_mean_and_std(v)
  mu = mean(v);
  sigma = std(x);
return
```

Outline

Cells

- Cell arrays in Matlab are multidimensional arrays whose elements are copies of other arrays.
- A cell array of empty matrices can be created with the cell function. But, more often, cell arrays are created by enclosing a miscellaneous collection of things in curly braces, {}.
- The braces are also used with subscripts to access the contents of various cells. For example,

```
A = [1 2 3; 4 5 6; 7 8 9];
C = {A sum(A) prod(prod(A))}
C
```

produces a 1-by-3 cell array.

The element (2,1) of A in the cell array can be accessed as

```
C{1}(2,1)
```

Multidimensional Arrays

These are arrays with more than two subscripts. For example,

```
A = zeros(2,3,4)
```

creates a $2 \times 3 \times 4$ array of zeros.

```
A (1, 1, 2) = 3
```

accesses the element at (1,1,2).

```
sum(A,3)
```

sums the elements in the the 3rd dimension.

Images

Color images are loaded as a $h \times w \times 3$ array. Following code loads an image called “fruits.jpg”. Then it zeros-out the red plane and the green plane. As a result the image becomes blue.

```
close all
im = imread('fruits.jpg');
imshow(im)
pause
im(:,:,1) = 0;
im(:,:,2) = 0;
figure
imshow(im)
```

In Octave, I have been successful in using

```
imshow(im/4)
```

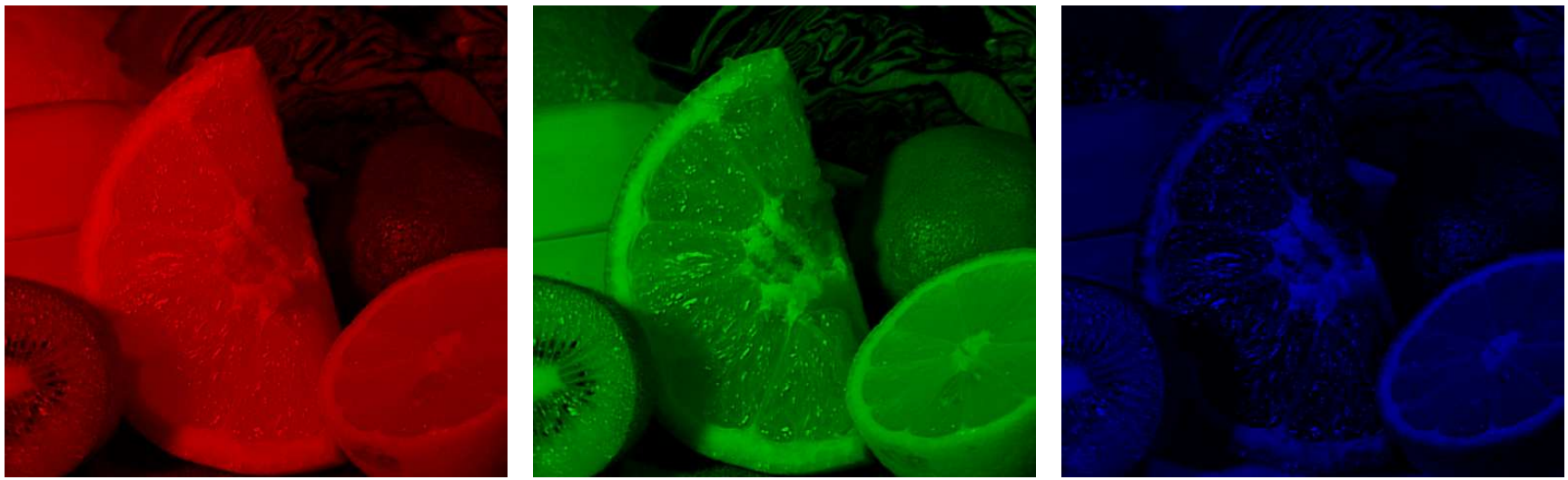
to display an image.

Figure 4 shows the two images.

(a) Color (b) Blue only

Figure 4: Color and blue-only versions.

Figure 5 shows all three colors.



(a) Red only (b) Green only (c) Blue only

Figure 5: Red, green, and blue versions.

Journals

- IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).
- International Journal of Computer Vision (IJCV).
- IEEE Transactions on Image Processing (TIP).
- IEEE Transactions on System, Man, and Cybernetics, Parts A, B, and C (SMC).

Conferences

- IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
- IEEE International Conference on Computer Vision.
- European Conference on Computer Vision.
- British Machine Vision Conference.
- IEEE Asian Conference on Computer Vision.
- IEEE International Conference on Robotics and Automation.
- IEEE International Conference on Image Processing.
- IEEE International Conference on Pattern Recognition.


Matthew Brown and David Lowe.
Recognising panoramas.
In *Proceedings of the 9th International Conference on Computer Vision*, pages 1218–1225, Nice, France, October 2003.


Paul Viola and Michael Jones.
Rapid object selection using a boosted cascade of simple features.
In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 511–518, Hawaii, December 2001.