SCSV3213

FUNDAMENTAL OF IMAGE PROCESSING

MATLAB TUTORIAL

Tutorial 1

Dr. Md Sah Hj Salam

Acknowledgements

- Slides Materials comes from below sources and re-organized for class suitability by Dr. Md Sah Hj Salam
  - Matlab helps
  - An Introduction to Matlab by John Sebeson from DeVry University lecture on matlab
  - MATLAB Tutorial by Qian Wang, Penn State University
  - Introduction to MATLAB slides by Markus Kuhn, university of Cambridge.

OUTLINE

- Intro to matlab
  - The MATLAB System / Environment
- The basic MATLAB programming
  - Fundamental expression
  - Matrix operation
- Use Matlab to solve linear equations
- M – file and function
- Conditional and Loop
- Image Processing Tools

1. MATLAB SYSTEM AND ENVIRONMENT

What is MATLAB?

- MATLAB stands for MATrix LABoratory.
- MATLAB is a high-performance language for technical computing.
  - Math and computation
  - Algorithm development (optimized for DSP)
  - Data acquisition
  - Modeling, simulation, and prototyping
  - Data analysis, exploration, and visualization
  - Scientific and engineering graphics
  - Application development, including graphical user interface building

What MATLAB is NOT ..

→ not a computer algebra system

→ not a strong general purpose programming language

- limited support for other data structures
- few software-engineering features; typical MATLAB programs are only a few lines long
- not suited for teaching OOP
- limited GUI features

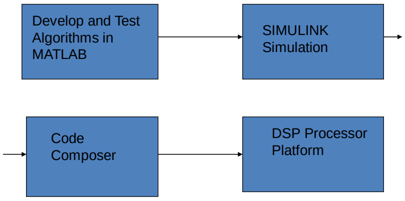
→ not a high-performance language (but fast matrix operators)

→ not freely available

Why Learn and Use MATLAB?

- Extensive built-in commands for scientific and engineering mathematics
- Easy way to generate class demonstrations and test examples
- Simple and intuitive programming for more complex problems
- Standard and widely-used computational environment with many features, extensions, and links to other software**.**

MATLAB in DSP Product Development



Develop and Test Algorithms in MATLAB

SIMULINK Simulation

Code Composer

DSP Processor Platform

MATLAB + PC = DSP Processor!! (just less efficient)

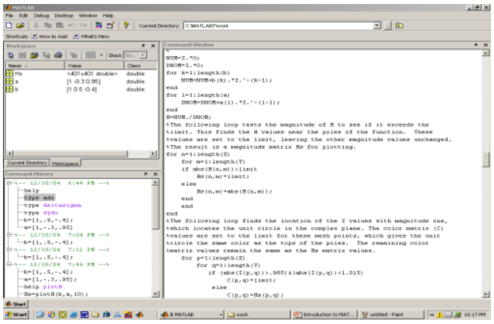
Why Learn MATLAB (and DSP)?

- Digital Signal Processing (DSP) is the dominant technology today, and into the future, for small-signal electronic systems (i.e., just about everything)
- MATLAB has become one of the standard design environments for DSP engineering
- Students need to be literate and skilled in this environment: knowledgeable in both DSP *and* MATLAB

The MATLAB System

- Development Environment.
  - MATLAB desktop
  - Editor and debugger for MATLAB programs (“m-files”)
  - Browsers for help, built-in and on-line documentation
  - Extensive demos
- The MATLAB Mathematical Function Library.
  - Elementary functions, like sum, sine, cosine, and complex arithmetic
  - More sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.
  - “Toolboxes” for special application areas such as Signal Processing
- The MATLAB Language.
  - “Programming in the small" to rapidly create quick and dirty throw-away programs, or
  - “Programming in the large" to create large and complex application programs.
- Graphics.
  - 2D and 3D plots
  - Editing and annotation features
- The MATLAB Application Program Interface (API).
  - A library that allows you to write C and Fortran programs that interact with MATLAB.

MATLAB Development Environment (Workspace)


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work
Shortcuts How to Add What's New
Workspace
Stack: Ba...
Current Directory Workspace
Command History
Command Window
Start
3 MATLAB
work
Introduction to MAT...
untitled - Paint
10:17 PM

Name	Value	Class
Hs	<401x401 double>	double
a	[1 -0.3 0.95]	double
b	[1 0.5 -0.4]	double

```
%-- 12/30/04  6:44 PM --%
   help
   type ado
   type deltasigma
   type dydx
   b=[1,.5,-.4];
   a=[1,-.3,.95]
%-- 12/30/04  7:04 PM --%
   b=[1,.5,-.4];
%-- 12/30/04  7:32 PM --%
   b=[1,.5,-.4];
%-- 12/30/04  7:46 PM --%
   b=[1,.5,-.4];
   a=[1,-.3,.95];
   help plotH
   Hs=plotH(b,a,10);
```

```
%
NUM=Z.*O;
DNOM=Z.*O;
for k=1:length(b)
    NUM=NUM+b(k).*Z.^-(k-1);
end
for l=1:length(a)
    DNOM=DNOM+a(l).*Z.^-(l-1);
end
H=NUM./DNOM;
%The following loop tests the magnitude of H to see if it exceeds the
%limit. This finds the H values near the poles of the function.  These
%values are set to the limit, leaving the other magnitude values unchanged.
%The result is a magnitude matrix Hs for plotting.
for n=1:length(X)
    for m=1:length(Y)
    if abs(H(n,m))>limit
        Hs(n,m)=limit;
    else
        Hs(n,m)=abs(H(n,m));
    end
    end
end
%The following loop finds the location of the Z values with magnitude one,
%which locates the unit circle in the complex plane. The color matrix (C)
%values are set to the limit for these mesh points, which gives the unit
%circle the same color as the tops of the poles.  The remaining color
%matrix values remain the same as the Hs matrix values.
    for p=1:length(X)
        for q=1:length(Y)
            if (abs(Z(p,q))>.985)&(abs(Z(p,q))<1.015)
                C(p,q)=limit;
             else
                C(p,q)=Hs(p,q);
```

DEMO 1 : OPENING MATLAB ENVIRONMENT

- Open your PC
- Go the start button
- Choose Matlab program.
- See the environment and find the windows for ..
  - – Command windows
  - – History
  - – Workspace
  - – Editors
  - – Current folder

MATLAB “Help” Utilities

- MATLAB is so rich that ‘help’ is ***essential***
  - – Command name and syntax
  - – Command input/output parameters
  - – Usage examples
- Help command
  - – help *command_name*
  - – help [partial_name] *tab*
- Help documents
- Demos

MATLAB Function Library (A Subset)

- matlab\general - General purpose commands.
- matlab\ops - Operators and special characters.
- matlab\lang - Programming language constructs.
- matlab\elmat - Elementary matrices and matrix manipulation.
- matlab\elfun - Elementary math functions.
- matlab\specfun - Specialized math functions.
- matlab\matfun - Matrix functions - numerical linear algebra.
- matlab\datafun - Data analysis and Fourier transforms.
- matlab\polyfun - Interpolation and polynomials.
- matlab\funfun - Function functions and ODE solvers.
- matlab\sparfun - Sparse matrices.
- matlab\scribe - Annotation and Plot Editing.
- matlab\graph2d - Two dimensional graphs.
- matlab\graph3d - Three dimensional graphs.
- matlab\specgraph - Specialized graphs.
- matlab\graphics - Handle Graphics.

Some Elementary Functions

- **Exponential.**

- exp - Exponential.
- expm1 - Compute exp(x)-1 accurately.
- log - Natural logarithm.
- log1p - Compute log(1+x) accurately.
- log10 - Common (base 10) logarithm.
- log2 - Base 2 logarithm and dissect floating point number.
- pow2 - Base 2 power and scale floating point number.
- realpow - Power that will error out on complex result.
- reallog - Natural logarithm of real number.
- realsqrt - Square root of number greater than or equal to zero.
- sqrt - Square root.
- nthroot - Real n-th root of real numbers.
- nextpow2 - Next higher power of 2.

Some Specialized Functions

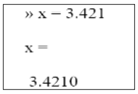
Number theoretic functions.

```
    factor      - Prime factors.
    isprime     - True for prime numbers.
    primes      - Generate list of prime numbers.
    gcd         - Greatest common divisor.
    lcm         - Least common multiple.
    rat         - Rational approximation.
    rats        - Rational output.
    perms       - All possible permutations.
    nchoosek    - All combinations of N elements taken K at a time.
    factorial   - Factorial function.
```

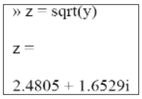
2. BASIC MATLAB PROGRAMMING

Fundamental Expression / Operation

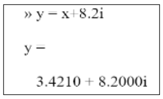
- MATLAB uses conventional decimal notion
- Builds expression with usual arithmetic operators and precedence rules :



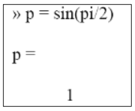
```
» x – 3.421

x =

 3.4210
```



```
» z = sqrt(y)

z =

2.4805 + 1.6529i
```

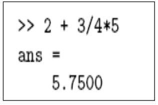


```
» y – x+8.2i

y –

  3.4210 + 8.2000i
```

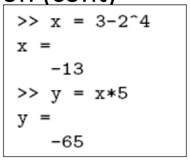


```
» p = sin(pi/2)

p =

        1
```

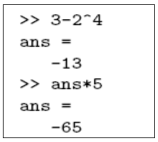
Fundamental Expression / Operation (cont)



```
>> 2 + 3/4*5
ans =
    5.7500
```



```
>> x = 3-2^4
x =
   -13
>> y = x*5
y =
   -65
```



```
>> 3-2^4
ans =
   -13
>> ans*5
ans =
   -65
```

Arithmetic Expression Priority

- Operation priorities is similar to c arithmetic

1. Brackets
2. Powers
3. * and / working from left to right
4. + and - working from left to right

- Example:

2+3/(4*5) → 2+ 3/20 → 2+0.150 → 2.150

2 + (3/4) * 5 → 2 + 0.75 * 5 → 2 + 3.750 → 5.750

Suppressing Output

- The result of an expression can be hidden (not display) on the command window by terminating the expression with semi-colon (;).
- We can also place several statements in one line separated by comma
- Example ..

```
>> x=-13; y = 5*x, z = x^2+y
y =
   -65
z =
   104
>>
```

Exercise :

- Find the answer for the following expression by hand and then compare your answer using MATLAB.

i) -2^3+9 ii) 2/3*3

iii) 3*2/3 iv) 3*4-5^2*2-3

v) (2/3^2*5)*(3-4^3)^2 vi) 3*(3*4-2*5^2-3)

Build in Functions

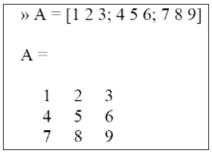
- MATLAB offer build in functions for easiness in calculation.
- For examples
  - cos, sin, acos, asin, sqrt, exp, log and more
- If you want to know the usage of the function just type help <the name of the function> in command windows)

>> help asin

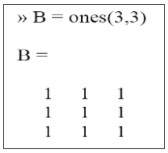
- Explore to know more..

Matrix operations

- Matrix operations are fundamental to MATLAB.
- Within a matrix, columns are separated by space and rows are separated by semicolon (;).
- For example:



```
» A = [1 2 3; 4 5 6; 7 8 9]

A =

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



```
» B = ones(3,3)

B =

    1    1    1
    1    1    1
    1    1    1
```

Matrix Operations (continue)

- matrix in MATLAB can do operations like
  - + addition
  - - subtraction
  - * multiplication
  - ^ power
  - ‘ transpose

Exercise.

1. create the following matrix

```
A =                B =
 1  2  3            2   4   6
 4  5  6            8  10  12
 7  8  9           14  16  18
```

2. Do these operations on the matrix. See the output

   i. A + B ii. A * B

   iii. B ‘ iv. A^B

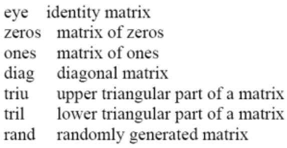
   v. A^2

Question ?

- What are the different between
  - A * B and A*2 in previous example.
- Assuming A = [1 2 3; 4 5 6; 7 8 9] and B = [3 2; 5 4; 7 6]
- Are these operation valid
  - i . A * B
  - ii. B * A
  - iii. A*3
  - iv. B*2

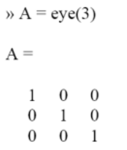
BUILDING MATRIX FUNCTIONS

- There are build in matrix functions for examples ..

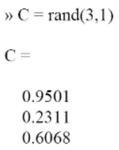


eye	identity matrix
zeros	matrix of zeros
ones	matrix of ones
diag	diagonal matrix
triu	upper triangular part of a matrix
tril	lower triangular part of a matrix
rand	randomly generated matrix

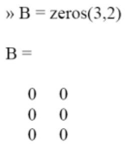
BUILDING MATRIX FUNCTIONS (cont)



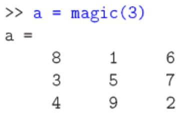
```
» A = eye(3)

A =

    1   0   0
    0   1   0
    0   0   1
```



```
» C = rand(3,1)

C =

    0.9501
    0.2311
    0.6068
```



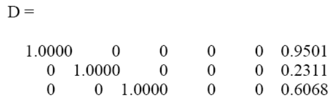
```
» B = zeros(3,2)

B =

    0   0
    0   0
    0   0
```



```
>> a = magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
```

BUILDING MATRIX FUNCTIONS (cont)

- Matrices can be build from block. For example.
- Using previous definition of Matrices A,B and C
- >> D = [A B C] will result to

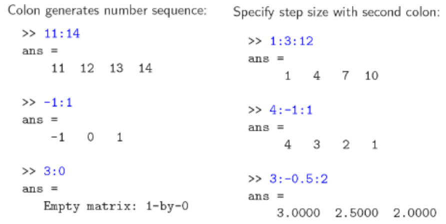


```
D =

   1.0000        0        0        0        0   0.9501
        0   1.0000        0        0        0   0.2311
        0        0   1.0000        0        0   0.6068
```

- The operation applies if the they have the same rows.

Colon Notation “ : “

- Colon notation can be used for various operation of matrices.



Colon generates number sequence:

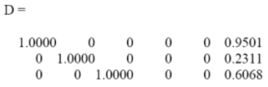
```
>> 11:14
ans =
    11   12   13   14

>> -1:1
ans =
    -1    0    1

>> 3:0
ans =
   Empty matrix: 1-by-0
```

Specify step size with second colon:

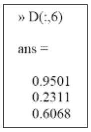
```
>> 1:3:12
ans =
     1    4    7   10

>> 4:-1:1
ans =
     4    3    2    1

>> 3:-0.5:2
ans =
    3.0000    2.5000    2.0000
```

Colon Notation “ : “

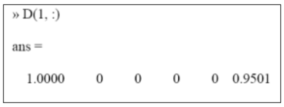
- We can also used colon for retrieving elements in a Matrix. For example..



```
D =

   1.0000        0        0        0        0   0.9501
        0   1.0000        0        0        0   0.2311
        0        0   1.0000        0        0   0.6068
```

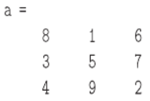


```
» D(:,6)

ans =

    0.9501
    0.2311
    0.6068
```

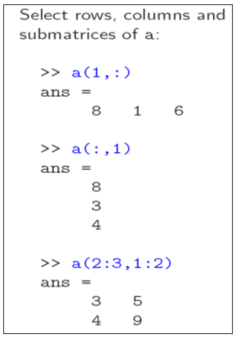


```
» D(1, :)

ans =

   1.0000        0        0        0        0   0.9501
```

Colon Notation “ : “ (more examples)



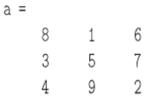
```
a =

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



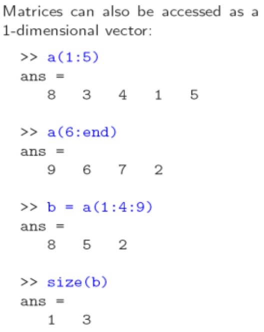
Select rows, columns and submatrices of a:

```
>> a(1,:)
ans =
     8     1     6

>> a(:,1)
ans =
     8
     3
     4

>> a(2:3,1:2)
ans =
     3     5
     4     9
```

Colon Notation “ : “ (more examples)



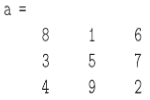
```
a =

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



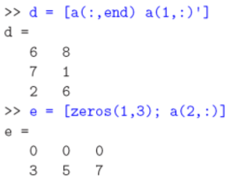
Matrices can also be accessed as a 1-dimensional vector:

```
>> a(1:5)
ans =
     8     3     4     1     5

>> a(6:end)
ans =
     9     6     7     2

>> b = a(1:4:9)
ans =
     8     5     2

>> size(b)
ans =
     1     3
```

More examples on matrices

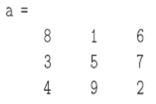


```
a =

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

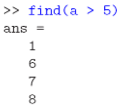


```
>> d = [a(:,end) a(1,:)']
d =
     6     8
     7     1
     2     6
>> e = [zeros(1,3); a(2,:)]
e =
     0     0     0
     3     5     7
```

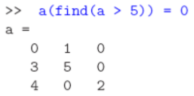
More examples on matrices



```
a =

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

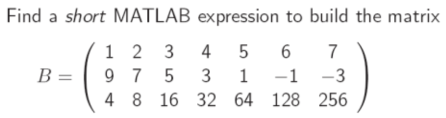


```
>> find(a > 5)
ans =
    1
    6
    7
    8
```



```
>>  a(find(a > 5)) = 0
a =
    0    1    0
    3    5    0
    4    0    2
```

Exercise 1



Find a *short* MATLAB expression to build the matrix

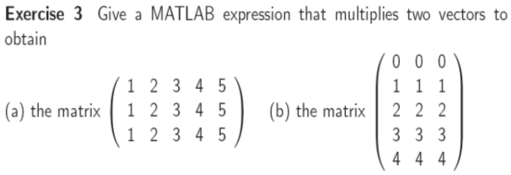
$$B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 7 & 5 & 3 & 1 & -1 & -3 \\ 4 & 8 & 16 & 32 & 64 & 128 & 256 \end{pmatrix}$$

EXERCISE 2

Based on matrix B write in a single line MATLAB command to

- Find the sum of column 5 and 7 of B
- Display the last row of B

exercise



Exercise 3 Give a MATLAB expression that multiplies two vectors to obtain

(a) the matrix $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$ (b) the matrix $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{pmatrix}$

3. SOLVING LINEAR EQUATION

System of Linear Equations

- A general system of linear equation can be expressed in term of coefficient matrix A.

  Ax = b component wise as

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots a_{n,n}x_n &= b_n \end{aligned}$$

- if A is (n *x* n) equations, then x can be find using this $x = A^{-1}b$ or x is the product of inverse A and b.
- In MATLAb function *inv* can be used for inverse function

Solve this linear equation

Find X and Y given ..

X + Y = 5;

2X + Y = 7;

```
>> A =[ 1 1 ;2 1]; C = [5 7];
>>  inv(A)*C'

ans =

     2
     3
```

Find X and Y and Z given ..

2X + 3Y + 4Z = 20;

10X + 5Y + 3Z = 30;

X + 5Y + 3Z = 10;

```
>> Z = [2 3 4;10 5 3; 1 5 3]; Y = [20 30 10];
>> inv(Z)*Y'

ans =

    2.2222
   -1.4141
    4.9495
```

4. M-FILE AND FUNCTION

M - File

- Last Time we do all the scripting in command windows.
  - Run at once
  - Not saved for later use
- MATLAB provide a platform for us to write code and save it.
- M-files are macros of MATLAB commands that are stored as ordinary text files with the extension "m", that is *filename*.m
- We can write command in M-File just like we write in command window or create function in M-File to be called.

Example 1 :

- Open your MATLAB environment.
- Open editor window and write these code.
- Save it as Test1.m

```
disp('This is a Test');
reply = input(‘ Y/N [Y] ?: ', 's');
 if isempty(reply)
     reply = 'Y';
 end
reply
```

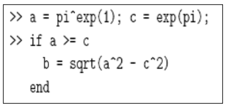
Understand the code

- Identify the used of functions in the code
  - disp
  - input
  - isempty
- The code shows the use of conditional statement if … end (noticed the syntax)

5. CONDITIONAL AND LOOP

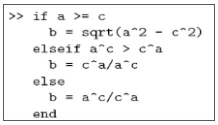
Conditional statement

- if .. end



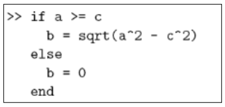
```
>> a = pi^exp(1); c = exp(pi);
>> if a >= c
     b = sqrt(a^2 - c^2)
   end
```

- if .. elseif .. end



```
>> if a >= c
     b = sqrt(a^2 - c^2)
   elseif a^c > c^a
     b = c^a/a^c
   else
     b = a^c/c^a
   end
```

- if ..else ..end



```
>> if a >= c
     b = sqrt(a^2 - c^2)
   else
     b = 0
   end
```

Exercise1 : conditional

- Rewrite the code in example 1 so that it will ask user whether he like morning class. If [Y] display “ I like ”, if [N] “ I hate” else or no answer “ehmm” ..

Example 2.

- Write and run these code.

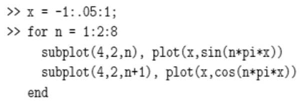
```
x = -1:.05:1;
for n = 1:2:8
subplot(4,2,n), plot(x,sin(n*pi*x))
subplot(4,2,n+1), plot(x,cos(n*pi*x))
end
```

Understand the code

- Identify the used of functions in the code
  - subplot
  - plot
- The code shows the use of loop statement for .. end (noticed the syntax)

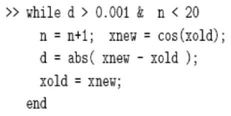
LOOP

- MATLAB has loop command similar to c
  - For loop



```
>> x = -1:.05:1;
>> for n = 1:2:8
     subplot(4,2,n), plot(x,sin(n*pi*x))
     subplot(4,2,n+1), plot(x,cos(n*pi*x))
   end
```

- While loop



```
>> while d > 0.001 &  n < 20
     n = n+1;  xnew = cos(xold);
     d = abs( xnew - xold );
     xold = xnew;
   end
```

Exercise 2:

- Find the highest value in a matrix of 6 x 6 generated using rand() function.

function

- In MATLAB, function name is the same name as the M-File name.
- The first line of the function file need to be written as follow

  function [list of output] = function_name [list of input]

- For example.
  - Function [A] = Area[a,b,c];
- Then the function can be called by its name for example
  - Area(3,5,6);
  - Area1 = Area(3,5,6);

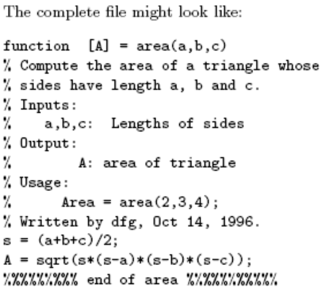
Example 3

- Write a function named Area that will compute the area of a triangle given length of side a,b,c.
- Solution :
  - Function name Area
  - Input parameter .. a,b,c
  - Output the area calculated , let says, A
  - The calculation / process in getting area use formula

$$A = \sqrt{s(s-a)(s-b)(s-c)},$$

whe

solution



The complete file might look like:

```
function  [A] = area(a,b,c)
% Compute the area of a triangle whose
% sides have length a, b and c.
% Inputs:
%    a,b,c:  Lengths of sides
% Output:
%        A: area of triangle
% Usage:
%      Area = area(2,3,4);
% Written by dfg, Oct 14, 1996.
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
%%%%%%%%% end of area %%%%%%%%%%%
```

Exercise 4

- Write a function that will do equation of y+ x = z
- Write a function that will solve problem of multiplication of y and x.
- Write a function that will receive a matrix of any size and return a new matrix with Each element being powered by 2.

*For each of the question call the function to test them in a file.

6. IMAGE PROCESSING APPLICATIONS

Images in Matlab

- Matlab is optimised for operating on matrices
- Images are matrices!
- Many useful built-in functions in the Matlab Image Processing Toolbox
- Very easy to write your own image processing functions

Loading and displaying images

```
>> I=imread('mandrill.bmp','bmp');          % load image
```

Matrix with image data

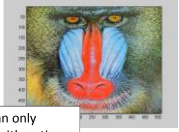
image filename as a string

image format as a string

```
>> image(I)   % display image
>> whos I

Name      Size          Bytes     Class

 I      512x512x3      786432    uint8 array

Grand total is 786432 elements using 786432 bytes
```

Dimensions of I (red, green and blue intensity information)

Matlab can only perform arithmetic operations on data with class double!



Display the left half of the mandrill image

Loading and displaying images

- Try using imshow(I) to display the image.
- Use different image of grayscale on both function imshow() and image();
- What are the differences ?

Representation of Images

- Images are just an array of numbers

```
>> I    % ctrl+c to halt output!
```

- Intensity of each pixel is represented by the pixel element's value in the red, green and blue matrices

```
>> I(1,1,:)         % RGB values of element (1,1)
ans(:,:,1) =
   135
ans(:,:,2) =
    97
ans(:,:,3) =
    33
```

Red

Green

Blue

Images where the pixel value in the image represents the intensity of the pixel are called intensity images.

Indexed images

- An indexed image is where the pixel values are indices to elements in a **colour map** or **colour lookup table**.
- The colour map will contain entries corresponding to red, green and blue intensities for each index in the image.

```
>> jet(20)  % Generate a jet colourmap for 20 indices
ans =
      0         0    0.6000
      0         0    0.8000
      0         0    1.0000
      0    0.2000    1.0000
      0    0.4000    1.0000
      0    0.6000    1.0000
      0    0.8000    1.0000
      0    1.0000    1.0000
 0.2000    1.0000    0.8000
 0.4000    1.0000    0.6000
 0.6000    1.0000    0.4000
 0.8000    1.0000    0.2000
 1.0000    1.0000         0
 1.0000    0.8000         0
 1.0000    0.6000         0
 1.0000    0.4000         0
 1.0000    0.2000         0
 1.0000         0         0
 0.8000         0         0
 0.6000         0         0
```

RGB Entry for index value 3

Values can range from 0.0 to 1.0

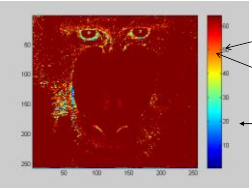
```
3  4   7   3  6  19   8  9   1  2
5  6  14   4  2   5   6  1   4  5
2  8   9   4  2  13   7  8   4  5
5  1  11   5  6   4   1  7   4  4
1  9   5   6  5   5  14  4   6  5
5  9   2   1 11   1   3  6   1  9
7  6   8  18  1   8   1  9  13  3
9  2   3   7  2   9   8 16   6  4
7  8   6   7  4  15   8  2   1  3
7  5  10   8  4  10   4  3   6  4
```

Red, green and blue intensities of the nearest index in the colourmap are used to display the image.

Displaying indexed images

```
>> I2=I(:,:,2);  % green values of I
>> image(I2)
>> colorbar  % display colourmap
```

Matlab considers I2 as an indexed image as it doesn't contain entries for red, green and blue entries



Index

Associated color

Colour Lookup Table

Displaying indexed images (continued)

- change colourmap

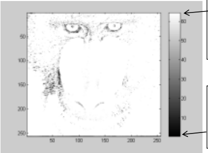
```
>> colormap(gray)
```

🔍 Type >>help graph3d to get a list of built-in colourmaps. Experiment with different built-in colourmaps.

🔍 Define your own colourmap mymap by creating a matrix (size m x 3) with red, green, blue entries. Display an image using your colourmap.

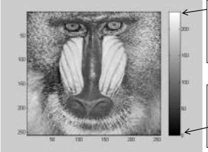
- scale colourmap

```
>> imagesc(I2)
```



Red =1.0, Green = 1.0, Blue =1.0, corresponds to index 64

Red =0.0, Green = 0.0, Blue = 0.0, corresponds to index 1

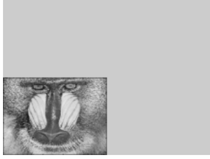


Red =1.0, Green = 1.0, Blue =1.0, corresponds to index 255

Red =0.0, Green = 0.0, Blue = 0.0, corresponds to index 0

Useful functions for displaying images

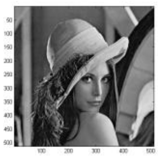
```
>> axis image   % plot fits to data
>> h=axes('position', [0 0 0.5 0.5]);
>> axes(h);
>> imagesc(I2)
```

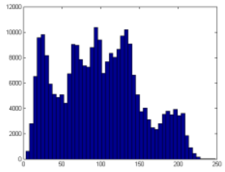


🔍 Investigate axis and axes functions using Matlab's help

Histograms

- Frequency of the intensity values of the image
- Quantise frequency into intervals (called bins)
- (Un-normalised) probability density function of image intensities





Computing histograms of images in Matlab

```
>>hist(reshape(double(Lena(:,:,2)),[512*512 1]),50)
```

Histogram function

Convert image into a 262144 by 1 distribution of values

Number of bins

Generate the histograms of the green channel of the Lena image using the following number of bins : 10, 20, 50, 100, 200, 500, 1000

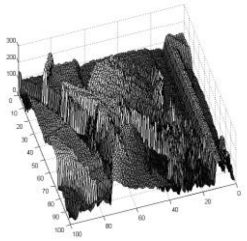
Histogram equalisation works by equitably distributing the pixels among the histogram bins. Histogram equalise the green channel of the Lena image using Matlab's histeq function. Compare the equalised image with the original. Display the histogram of the equalised image. The number of pixels in each bin should be approximately equal.

Visualising the intensity surface

```
>>surf(double(imresize(Lena(:,:,2),[50 50])))
```

Change type to double precision

Remember to reduce size of image!



Use Matlab's built-in mesh and shading surface visualisation functions

Useful functions for manipulating images

- Convert image to grayscale

>>Igray=rgb2gray(I);

- Resize image

>>Ismall=imresize(I,[100 100], 'bilinear');

- Rotate image

>>I90=imrotate(I,90);

Other useful functions

Convert polar coordinates to cartesian coordinates >>pol2cart(rho,theta)	Check if a variable is null >>isempty(I)	Trigonometric functions sin, cos, tan
Convert polar coordinates to cartesian coordinates >>cart2pol(x,y)	Find indices and elements in a matrix >>[X,Y]=find(I>100)	Fast Fourier Transform fft2(I)
Get size of matrix >>size(I)	Change the dimensions of a matrix >>reshape(rand(10,10),[100 1])	Discrete Cosine Transform dct(I)
Add elements of a Matrix (columnwise addition in matrices) >>sum(I)	Exponentials and Logarithms exp log log10	

Convolution

Bit of theory! Convolution of two functions f(x) and g(x)

$$h(x) = f(x) \otimes g(x) = \int_{-\infty}^{+\infty} f(r)g(x-r)dr$$

Output filtered image

Image

convolution operator

Filter (mask/kernel)

Support region of filter where g(x-r) is nonzero

Discrete image processing 2D form

$$H(x,y) = \sum_{j=1}^{height} \sum_{i=1}^{width} I(i,j)M(x-i,y-j)$$

Compute the convolution where there are valid indices in the kernel

Convolution example

Image (I)

197	199	195	194	189	190	132	90	112	101
194	194	198	201	189	196	150	85	87	97
194	194	198	195	186	191	109	90	90	124
197	187	195	198	185	186	115	78	81	96
194	190	198	193	187	177	88	86	94	69
194	194	190	190	179	177	93	99	95	100
201	194	191	186	186	181	74	110	82	76
196	194	195	191	183	164	77	119	84	88
192	194	199	192	191	174	89	164	103	129
201	190	187	189	178	168	90	82	88	84

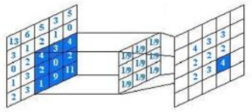
Filter (M)

j	1/9	1/9	1/9
	1/9	1/9	1/9
	1/9	1/9	1/9
	i		

=

0	0	0	0	0	0	0	0	0	0
0	196	196	194	192	170	137	105	97	0
0	195	196	194	192	167	133	98	92	0
0	194	194	193	189	158	124	92	90	0
0	193	193	191	186	154	122	92	89	0
0	194	192	189	184	149	121	91	90	0
0	194	192	188	182	146	122	93	95	0
0	195	193	190	183	147	128	100	106	0
0	194	192	189	181	146	125	100	105	0
0	0	0	0	0	0	0	0	0	0

$$H(x,y) = \sum_{j=1}^{height} \sum_{i=1}^{width} I(i,j)M(x-i,y-j)$$



http://www.s2.chalmers.se/undergraduate/courses0203/ess060/PDFdocuments/ForScreen/Notes/Convolution.pdf

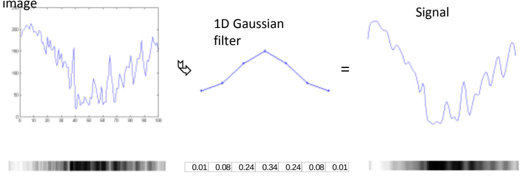
Write your own convolution function myconv.m to perform a convolution. It should accept two parameters – the input matrix (image) and convolution kernel, and output the filtered matrix.

Convolution example in 1D

Horizontal slice from Mandrill image

1D Gaussian filter

Filtered Signal



0.01	0.08	0.24	0.34	0.24	0.08	0.01

Common convolution kernels

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

Arithmetic mean filter (smoothing)
>>fspecial('average')

-0.17	-0.67	-0.17
-0.67	3.33	-0.67
-0.17	-0.67	-0.17

Laplacian (enhance edges)
>>fspecial('laplacian')

-0.17	-0.67	-0.17
-0.67	4.33	-0.67
-0.17	-0.67	-0.17

Sharpening filter
>>fspecial('unsharp')

0.01	0.08	0.01
0.08	0.62	0.08
0.01	0.08	0.01

Gaussian filter (smoothing)
>>fspecial('gaussian')

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Sobel operators (edge detection in x and y directions)
>>fspecial('sobel')
>>fspecial('sobel')'

Investigate the listed kernels in Matlab by performing convolutions on the Mandrill and Lena images. Study the effects of different kernel sizes (3x3, 9x9, 25x25) on the output.

The median filter is used for noise reduction. It works by replacing a pixel value with the median of its neighbourhood pixel values (vs the mean filter which uses the mean of the neighbourhood pixel values). Apply Matlab's median filter function medfilt2 on the Mandrill and Lena images. Remember to use different filter sizes (3x3, 9x9, 16x16).

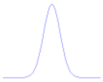
Useful functions for convolution

- Generate useful filters for convolution

>>fspecial('gaussian',[kernel_height kernel_width],sigma)

- 1D convolution

>>conv(signal,filter)



- 2D convolution

>>conv2(double(I(:,:,2)),fspecial('gaussian‘,[kernel_height kernel_width] ,sigma),'valid')

image

kernel

Border padding options

Perform the convolution of an image using Gaussian kernels with different sizes and standard deviations and display the output images.

End of MATLAB Tutorial

SCSV 3213