M'hamed Bougara University - Boumerdès
Faculty of Sciences
Computer Science Department

Course: Programming Tools for Mathematics Matlab
L1

Dr RAHMOUNE Nabila

Chapter 4
Numerical analysis and image processing with Matlab

Outline

- Polynomial Evaluation
- Resolution of Systems of Equations M-File
- Probability and Statistics
- MATLAB and Image Processing
- Applications

Polynomial Evaluation

y = polyval(p,x): returns the value of a polynomial of degree n evaluated at x. The input argument p is a vector of length n+1 whose elements are the coefficients in descending powers of the polynomial to be evaluated.

y = $p_1 x^n + p_2\ x^{n-1} + \ldots + p_n\ x \quad + p_{n+1}$

x can be a matrix or a vector. In either case, polyval evaluates p at each element of x.

Example :

The polynomial $p(x) = 1 - 2x + 4x^3$ is represented by the list:

```
>> p = [4  0  -2  1]
      p = 4    0     -2      1
>> y=polyval(p, 2)
      y= 29
```

```
polyval(p, 2)
  ans =     29
```

Polynomial Evaluation

The polynomial p(x)=3x^2+2 x +1 is evaluated at x = 5, 7, and 9 with p = [3 2 1];

```
>> polyval(p,[5 7 9])
```

which results in

```
ans =

    86   162   262
```

```
>> polyval(p,[5 7 9;1 2 3 ])

ans =

    86   162   262
     6    17    34
```

Polynomial roots

The **roots** function solves polynomial equations of the form

$p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1} = 0$

Polynomial equations contain a single variable with non negative exponents.

To find the roots of other types of equations, use **fzero**.

Syntax

r = roots(p)

Example:

Determine the roots of the polynomial $p(x)$.

$p(x)=3x^2+2x+1$

```
>>p = [3 2 1];
>> r=roots(p)
r =
  -0.3333 + 0.4714i
  -0.3333 - 0.4714i
```

Polynomial derivative

The polyder function calculates the derivative of polynomials, polynomial products, and polynomial quotients. The operands a, b, and p are vectors whose elements are the coefficients of a polynomial in descending powers.

- k = polyder(p) returns the derivative of the polynomial p.
- k = polyder(a,b) returns the derivative of the product of the polynomials a and b.
- [q,d] = polyder(b,a) returns the numerator q and denominator d of the derivative of the polynomial quotient b/a.

Polynomial derivative

Example

The derivative of the product

$(3x^2+6\ x\ +9)(x^2+2\ x)$

is obtained with

```
>> a = [3 6 9];
>> b = [1 2 0];
>> k = polyder(a,b)
        k = 12 36 42 18
```

This result represents the polynomial

$12x^3+36x^2+42\ x\ +18$

Nonlinear function Roots

x = fzero(fun,x0) tries to find a point x where fun(x) = 0.

This solution is where fun(x) changes sign.

- The first argument is a function handle.
- The second argument is the initial guess. If we provide a different initial guess, we get a different root.

Nonlinear function Roots

```
function res = func(x)                    %  M-File name is also func
    res = x^2 - 2*x -3;
end
```

You can call func from the Command Window, and confirm that there are zeros at 3 and -1.

```
>> func(3)
        ans = 0
>> func(-1)
        ans = 0
```

When we don't know exactly where the roots are; we only know that one of them is near 4. We can call fzero like this:

```
>> fzero(@func, 4)
     ans = 3.0000
```

```
>> fzero(@func, -2)
     ans = -1
```

Matrix determinant

d = det(A) returns the determinant of square matrix A.

Examples

Calculate Determinant of Matrix
Create a 3-by-3 square matrix, A.
A = [1 -2 4; -5 2 0; 1 0 3]

```
A =
     1    -2     4
    -5     2     0
     1     0     3
d = det(A)
d =
    -32
```

The determinant of A is -32.

Matrix inverse

Y = **inv(X)** returns the inverse of the **square** matrix X.

```
M = [3 2 ; 2 1];

% Creating a 2 X 2 square matrix

>> I = inv(M)

    ans =

      -1.0000    2.0000
       2.0000   -3.0000
```

Resolution of Systems of Equations with MATLAB

A system of n linear equations in p variables is called the following system:

$$(S)\begin{cases} a_{11}x_1+a_{12}x_2+a_{13}x_3+\ldots+a_{1p}x_p=b_1 & \text{(eq1)} \\ a_{21}x_1+a_{22}x_2+a_{23}x_3+\ldots+a_{2p}x_p=b_2 & \text{(eq2)} \\ a_{n1}x_1+a_{n2}x_2+a_{n3}x_3+\ldots+a_{np}x_p=b_n & \text{(eqn)} \end{cases}$$

Matrix representation

$$(S)\begin{cases} a_{11}x_1+a_{12}x_2+a_{13}x_3+\ldots+a_{1p}x = b_{p\ 1} & \text{(eq1)} \\ a_{21}x_1+a_{\ 22}x_2+a_{23}x_3+\ldots+a_{2p}x_p = b_{\ 2} & \text{(eq2)} \\ a_{n1}x_1+a_{\ n2}x_2+a_{n3}x_3+\ldots+a_{\ np}x_p = b_{\ n} & \text{(eqn)} \end{cases}$$

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1p} \\ a_{21} & a & \ldots_{11} & a_{2p} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & & a_{np} \end{bmatrix} \times \begin{bmatrix} x1 \\ x2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

A **X** **B**

A*X = B · X = A^{-1}*B

Resolution of Systems of Equations

Example 1

Write the following system S1 in matrix form.

$$\textbf{(S1)} \begin{cases} x_1 + 3x_2 + 4x_3 = 50 \\ 3x_1 + 5x_2 - 4x_3 = 2 \\ 4x_1 + 7x_2 - 2x_3 = 31 \end{cases}$$

$$A = \begin{bmatrix} 1 & 3 & 4 \\ 3 & 5 & -4 \\ 4 & 7 & -2 \end{bmatrix} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad B = \begin{bmatrix} 50 \\ 2 \\ 31 \end{bmatrix}$$

A * X = B

Resolution of Systems of Equations

Example 2

Write the following system S1 in matrix form.

$$\textbf{(S2)} \begin{cases} x+y+z+t=5 \\ x+2y+z+t=6 \\ 2x+y+z+t=7 \\ x+y+z+2t=8 \end{cases}$$

$$A=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \quad X=\begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} \quad B=\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$$

A * X = B

Existence of solutions of a system of linear equation

If the determinant of a matrix A is equal to 0, then:

The associated system either does not have a solution or has infinitely many solutions.

the determinant is different from 0, then:

The associated system has a unique solution.

Existence of solutions of a system of linear equation

$$A*X = B \Rightarrow X = A^{-1} * B$$

$$A * X = B$$

$$A^{-1} * A * X = A^{-1} * B$$

$$I * X = A^{-1} * B$$

$$X = A^{-1} * B$$

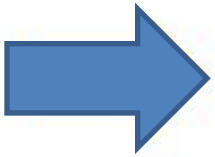
A^{-1} Is the inverse matrix of A.

This matrix is calculated using the inv function

.

Existence of solutions of a system of linear equation

The command creates a column vector containing each solution of the system.

A*X = B  **X = A^{-1} *B**

The calculation of solutions is therefore written as follows:

```
>> X = inv(A) * B ;
```

The command creates a column vector containing each solution of the system.

Existence of solutions of a system of linear equation

$$(S1)\begin{cases} x_1 + 3x_2 + 4x_3 = 50 \\ 3x_1 + 5x_2 - 4x_3 = 2 \\ 4x_1 + 7x_2 - 2x_3 = 31 \end{cases}$$

$$A = \begin{pmatrix} 1 & 3 & 4 \\ 3 & 5 & -4 \\ 4 & 7 & -2 \end{pmatrix} \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad B = \begin{pmatrix} 50 \\ 2 \\ 31 \end{pmatrix}$$

```
>> A = [1 3 4 ; 3  5 -4 ; 4 7 -2];
>> B = [ 50 ; 2 ; 31 ];
>> X = inv(A) * B
X =  3
     5
     8
```

One solution of the system is the vector X: $X = \begin{pmatrix} 3 \\ 5 \\ 8 \end{pmatrix}$ x_1 x_2 x_3

Existence of solutions of a system of linear equation

Provide the MATLAB commands to solve S2.

$$(S2)\begin{cases} x+y+z+t=5 \\ x+2y+z+t=6 \\ 2x+y+z+t=7 \\ x+y+z+2t=8 \end{cases}$$

$$A=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \quad X=\begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} \quad B=\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$$

```
>> A = [1 1 1 1 ; 1 2 1 1 ; 2 1 1 1 ; 1 1 1 2];
>> B = [5 ; 6 ; 7 ; 8];
>> X = inv(A)*B
X = 2.0000
    1.0000
   -1.0000
    3.0000
```

The solution of the system is the vector X:

$$X=\begin{bmatrix} 2 \\ 1 \\ -1 \\ 3 \end{bmatrix} \begin{matrix} \leftarrow \mathbf{x} \\ \leftarrow \mathbf{y} \\ \leftarrow \mathbf{z} \\ \leftarrow \mathbf{t} \end{matrix}$$

Image

Loading an image:

```
a = imread('picture.jpg')
imshow(a);
```

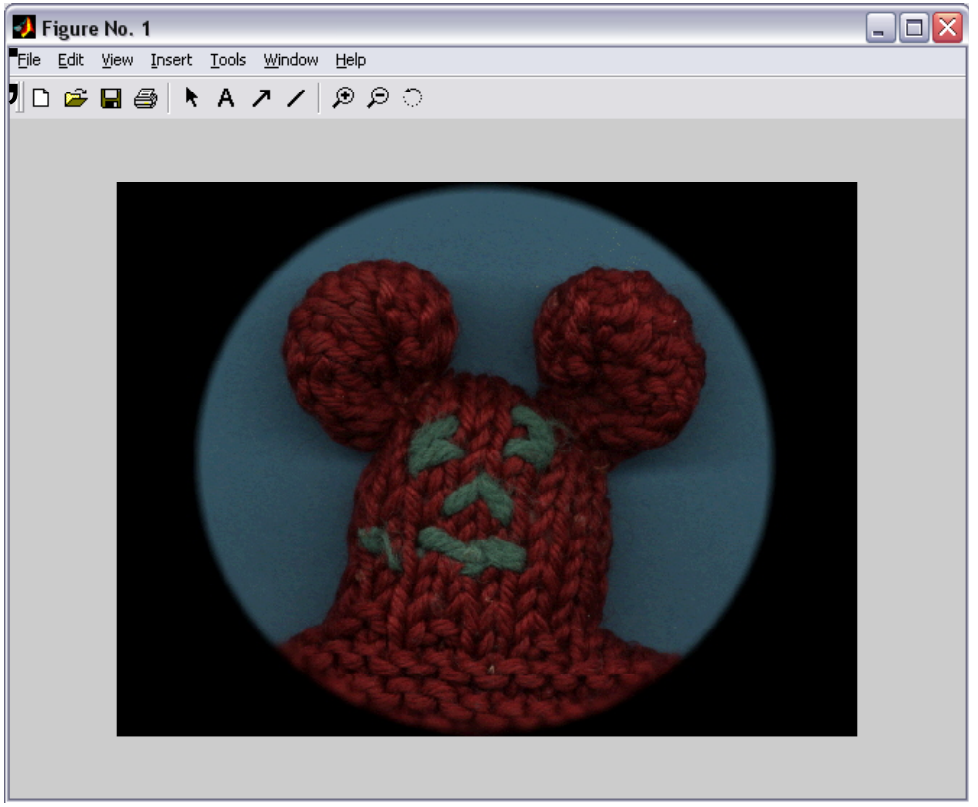
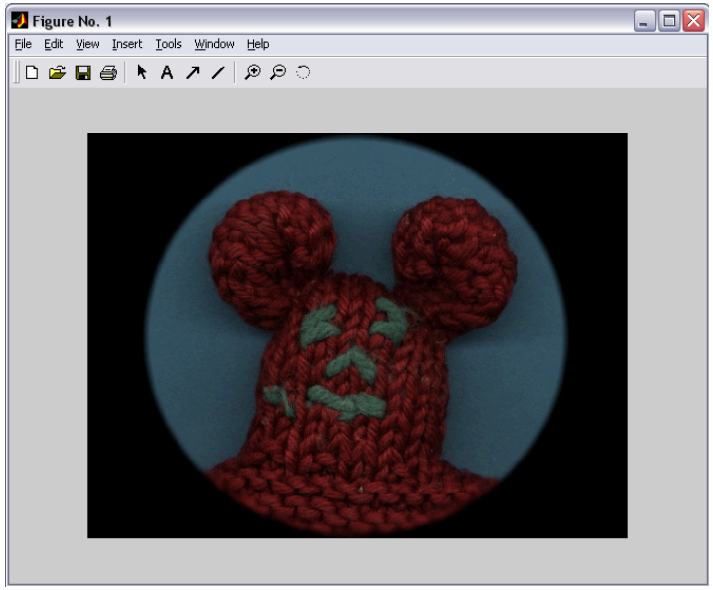


Image (=matrix) size:

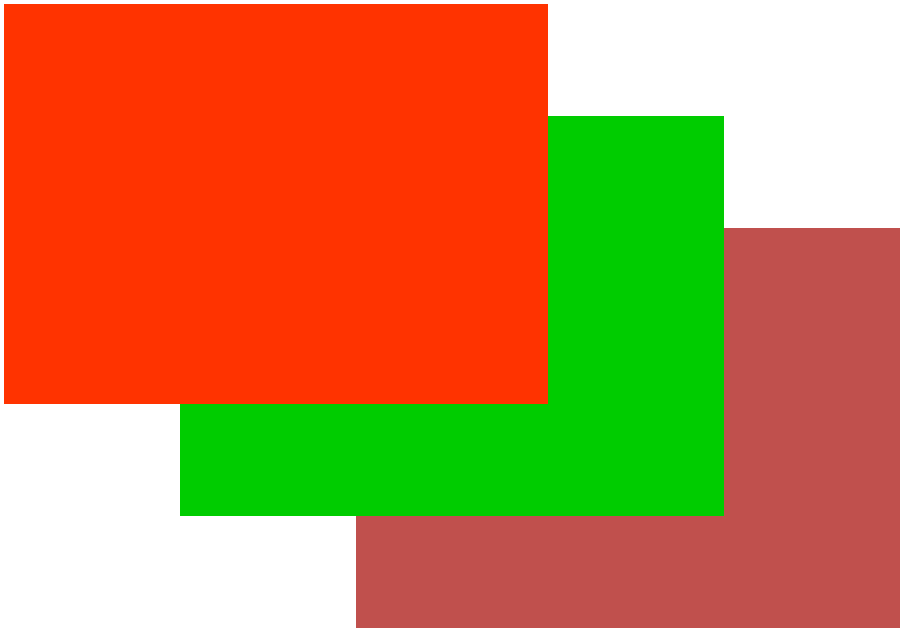
```
size(a): 384   512   3
```

R G B

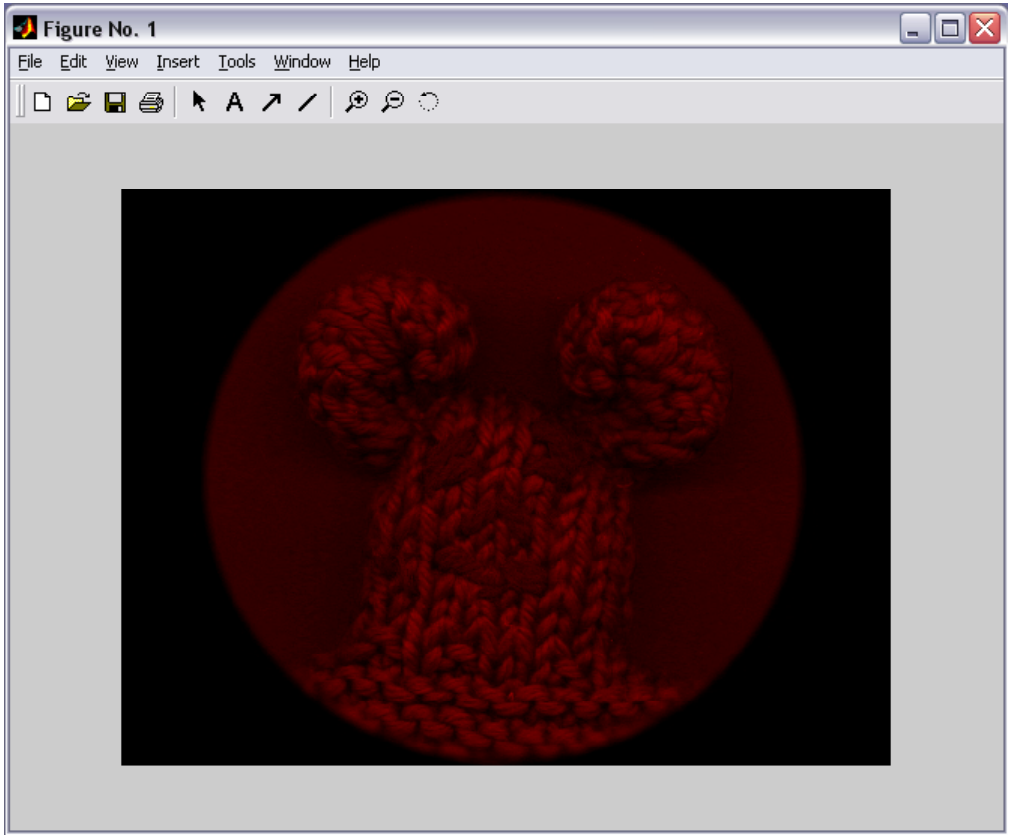
Image



Color image:
3D Matrix of RGB planes



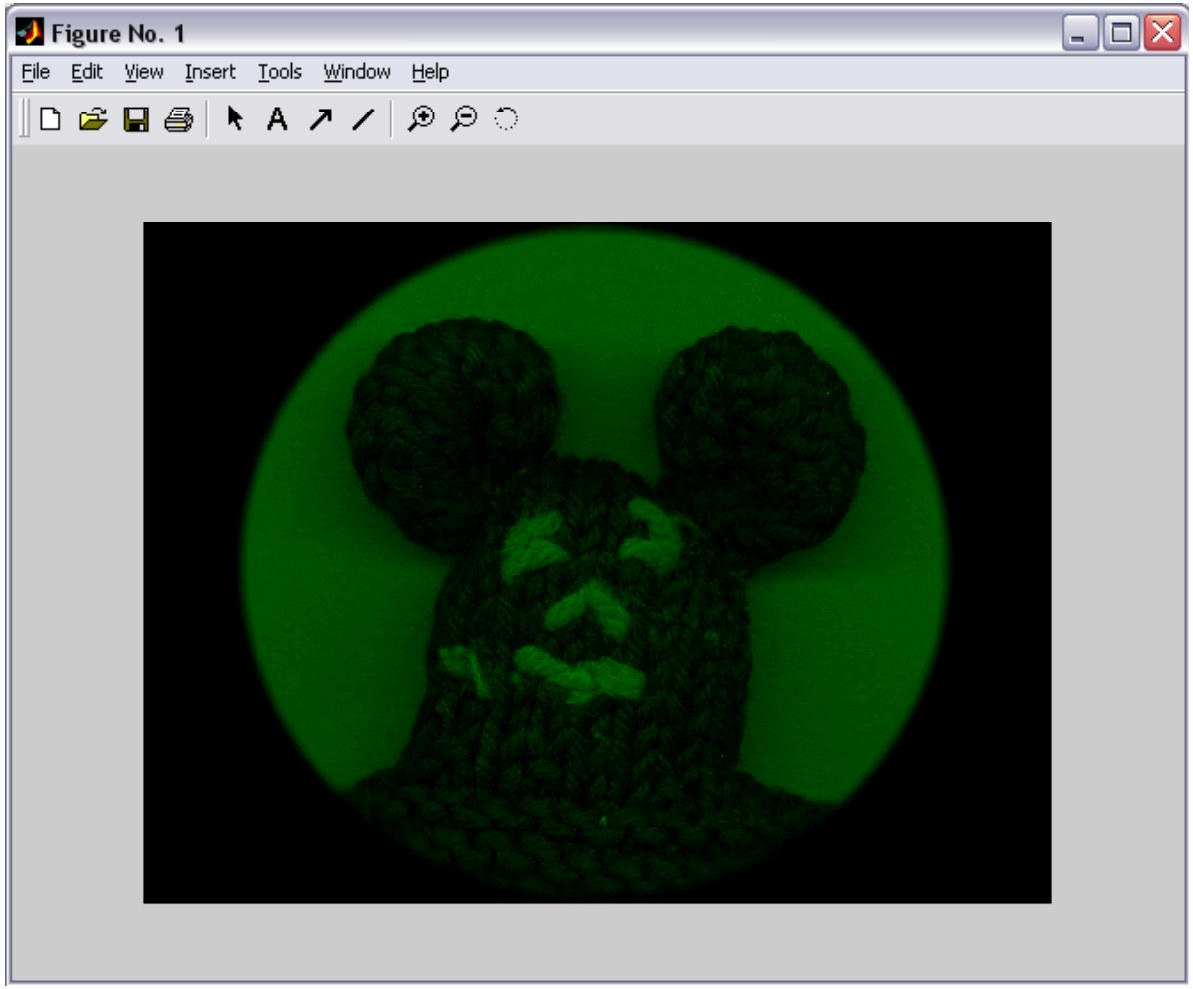
Image



Show RED plane:

```
a(:,:,2:3) = 0;
imshow(a);
```

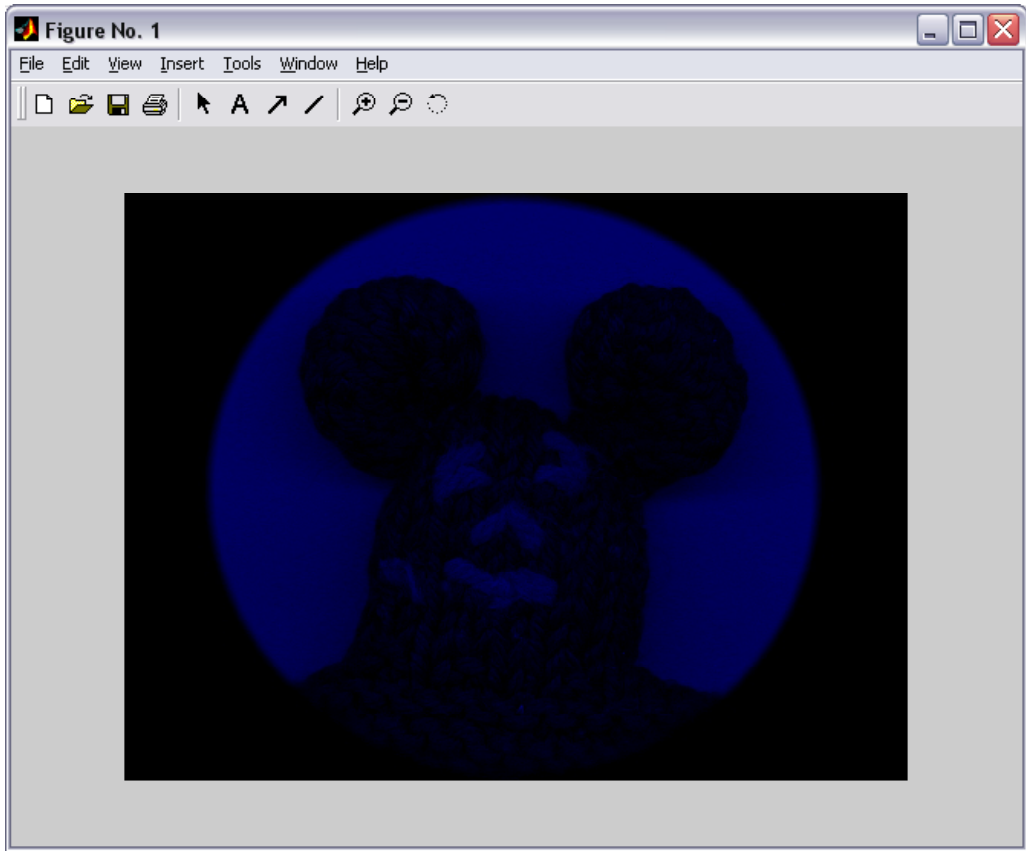
Image



Show GREEN plane:

```
a(:,:,[1 3]) = 0;
imshow(a);
```

Image

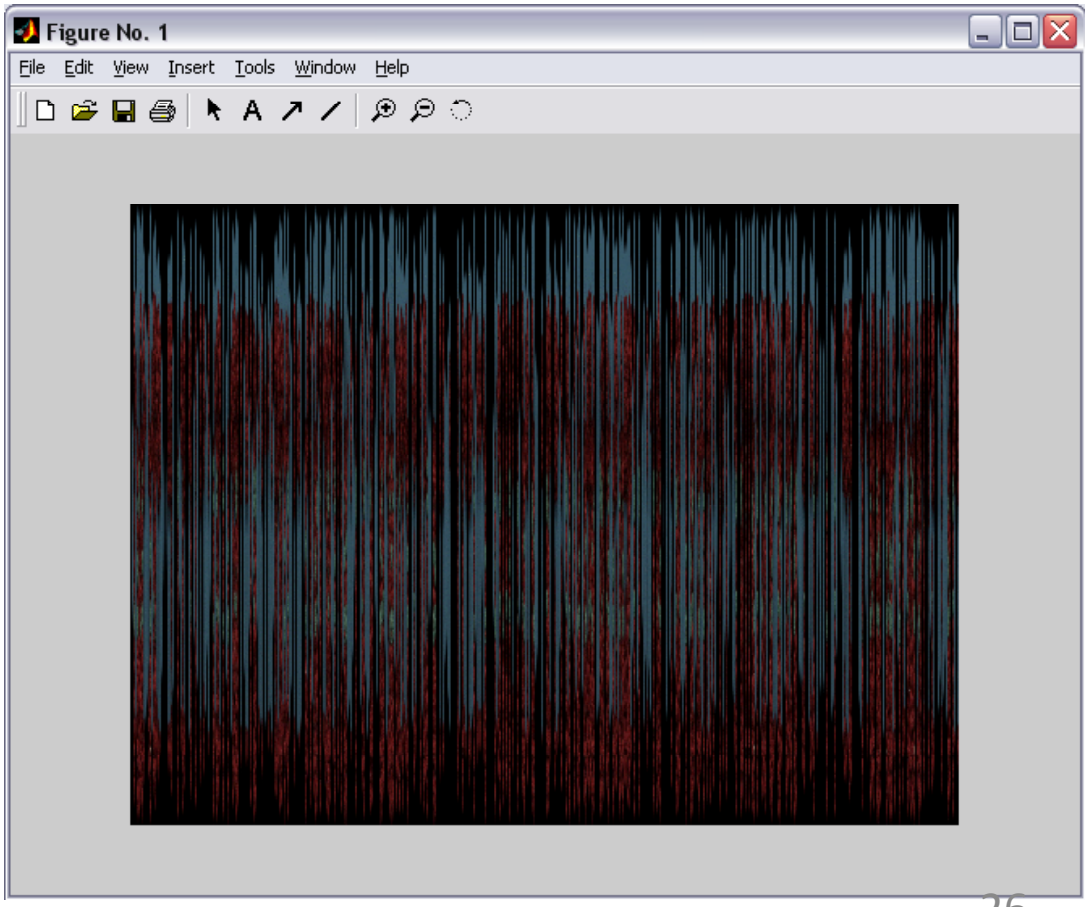


Show BLUE plane:

```
a(:,:,1:2) = 0;
imshow(a);
```

Advanced: Shuffling columns

```
rn = rand(1,512);
[rn1,i] = sort(rn);
b = a(:,i,:);
imshow(b);
```



Reading / Writing Images

Images can be imported as a matrix of pixel values

```
>> im=imread('myPic.jpg');
>> imshow(im);
```

- Matlab supports almost all image formats

➢ jpeg, tiff, gif, bmp, png, …

➢ see help imread for details (e.g., pixel format and types)

- To write an image, give:

➢ rgb matrix (0 to 1 doubles, or 0 to 255 uint8)

```
>> imwrite(rand(300,300,3),'t1.jpg');
```

➢ indices and colormap

```
>> imwrite(ceil(rand(200)*256),jet(256),'t2.jpg');
```

➢ see help **imwrite** for more options

Statistics

- Whenever analyzing data, you have to compute statistics

```
>> scores = 100*rand(1,100); % random data
```

- Built-in functions ➢ mean, median, mode
- To group data into a histogram

```
>> hist(scores,5:10:95);
```

➢ makes a histogram with bins centered at 5, 15, 25…95

```
>>  hist(scores,20);
```

➢ makes a histogram with 20 bins

```
>> N=histc(scores,0:10:100);
```

➢ returns the number of occurrences between the specified bin *edges* 0 to <10, 10 to <20…90 to <100. you can plot these manually:

```
>> bar(0:10:100,N,'r')
```

Random Numbers

- Many probabilistic processes rely on random numbers
- MATLAB contains the common distributions built in

```
>>  rand
```

➢ draws from the uniform distribution from 0 to 1

```
>> randn
```

➢ draws from the standard normal distribution (Gaussian)

```
>> random
```

➢ can give random numbers from many more distributions

➢ see **help random**

- You can also seed the random number generators

```
>> rand('state',0); rand(1); rand(1); rand('state',0);
rand(1);                    % same random number
```

Changing Mean and Variance

We can alter the given distributions

```
>> y=rand(1,100)*10+5;
```

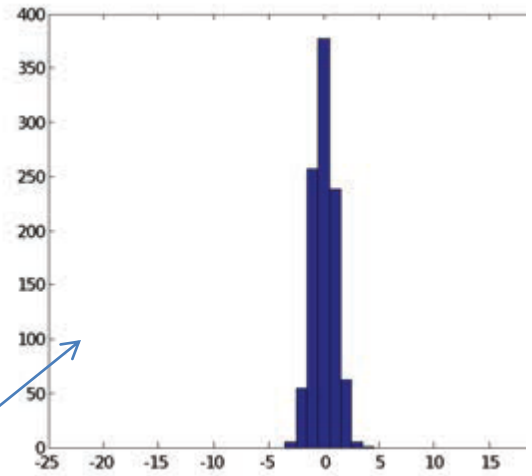
➢ gives 100 uniformly distributed numbers between 5 and 15

```
>> y=floor(rand(1,100)*10+6);
```

➢ gives 100 uniformly distributed integers between 6 and 15.

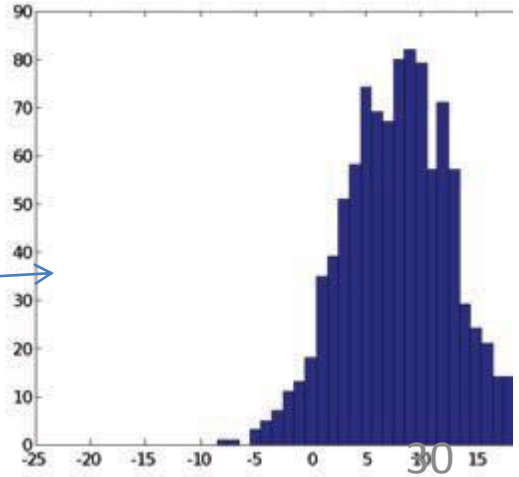
floor or ceil is better to use here than round

➢ you can also use randi([6,15],1,100)




```
>> y=randn(1,1000)
>> y2=y*5+8
```

➢ increases std to 5 and makes the mean 8



Application

We will simulate Brownian motion in 1 dimension.

- Call the script ‘brwn’
- Make a 10,001 element vector of zeros
- Write a loop to keep track of the particle’s position at each time
- Assume middle of the vector is position 0.
- To get the new position, pick a random number, and if it’s <0.5, go left; if it’s >0.5, go right.
- Keep count of how many times each position is visited.
- Plot a 50 bin histogram of the positions.