ELG3121 Lab 1

Computers are often used to simulate random events to make calculations about the behavior of random experiments through so-called Monte Carlo experiments. To perform any such simulation of random phenomena, we need the ability of the computer to generate a random result. Usually, this takes the basic form of a subroutine that when called is supposed to return a real number that has an equal chance of being any value from the interval (0, 1). (In Matlab, the subroutine is named `"rand"`). Moreover, each time the subroutine is called, the value produced is described by a random number that is independent of the random number for any other call of the subroutine.

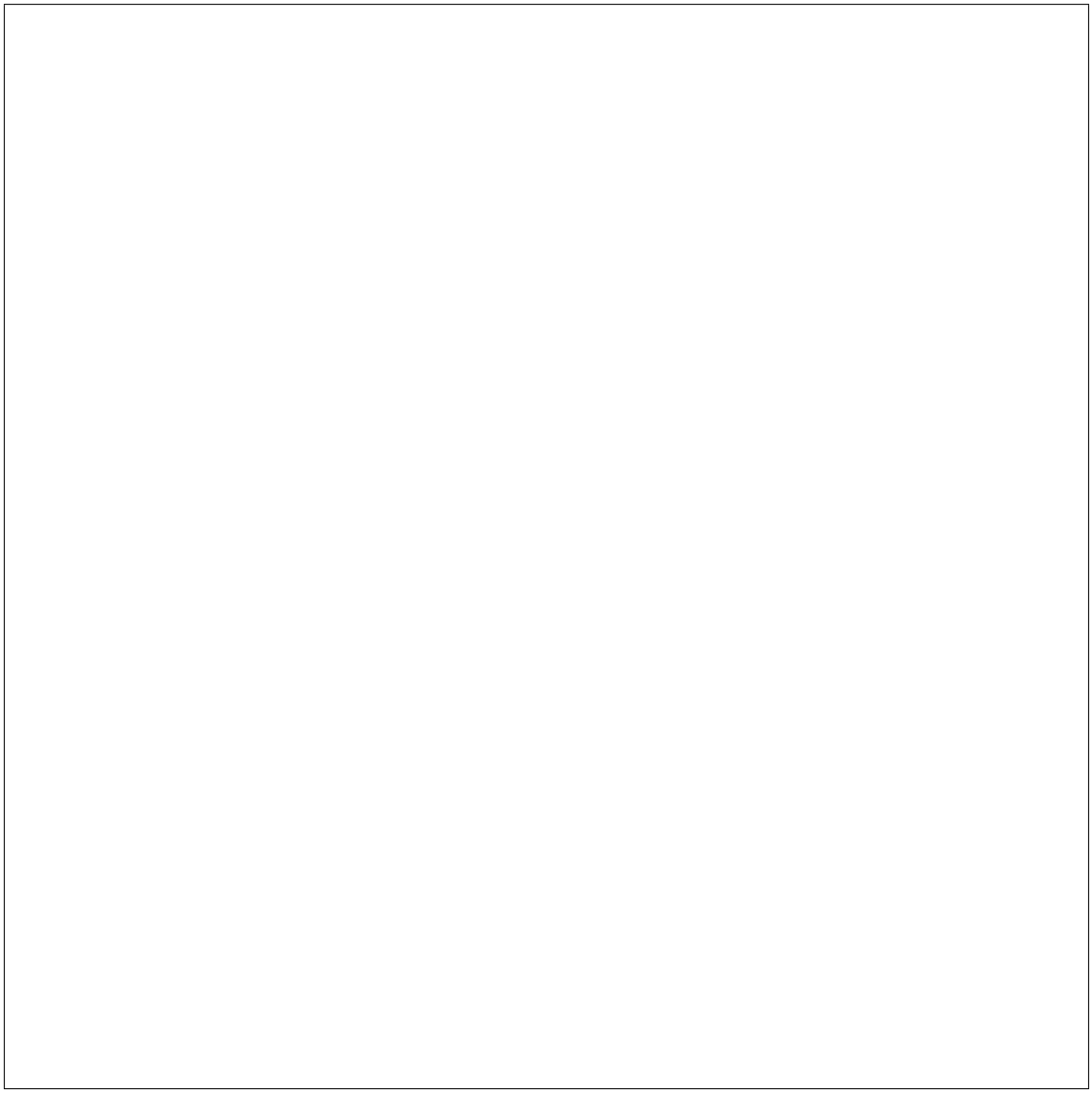
Experiment 1:

Toss a coin 50 times. Denote each outcome by a "0" or "1" corresponding to Head "H" or Tail "T" respectively. Record the outcome of your 50 trials.

a) What is the sample space for each trial?

b) Estimate the probability of getting "0" (i.e. Head "H") from the observed outcome?

c) Using Matlab, create a vector (e.g. x) which contains the results of the 50 trials. Plot the histogram of the outcomes using subroutine "`hist`". (see Appendix A)



d) Simulate this experiment using Matlab, by tossing the coin 100,000 times instead of 50. Calculate the relative frequency of getting "Head" (i.e. "0").
(Hint: To do that, you may call the subroutine "`rand`" 100,000 times using "for" loop, or using the subroutine "`rand( )`" to generate 100,000 random numbers. See Appendix A).

Experiment 2:

Consider now the experiment of tossing three coins 100,000 times. Generate a vector using Matlab which contains all the ordered 3-tuple outcomes.

a) What is the sample space for each trial?

b) Let A be defined as the event of obtaining two Heads in a single trial. List all elements in A.

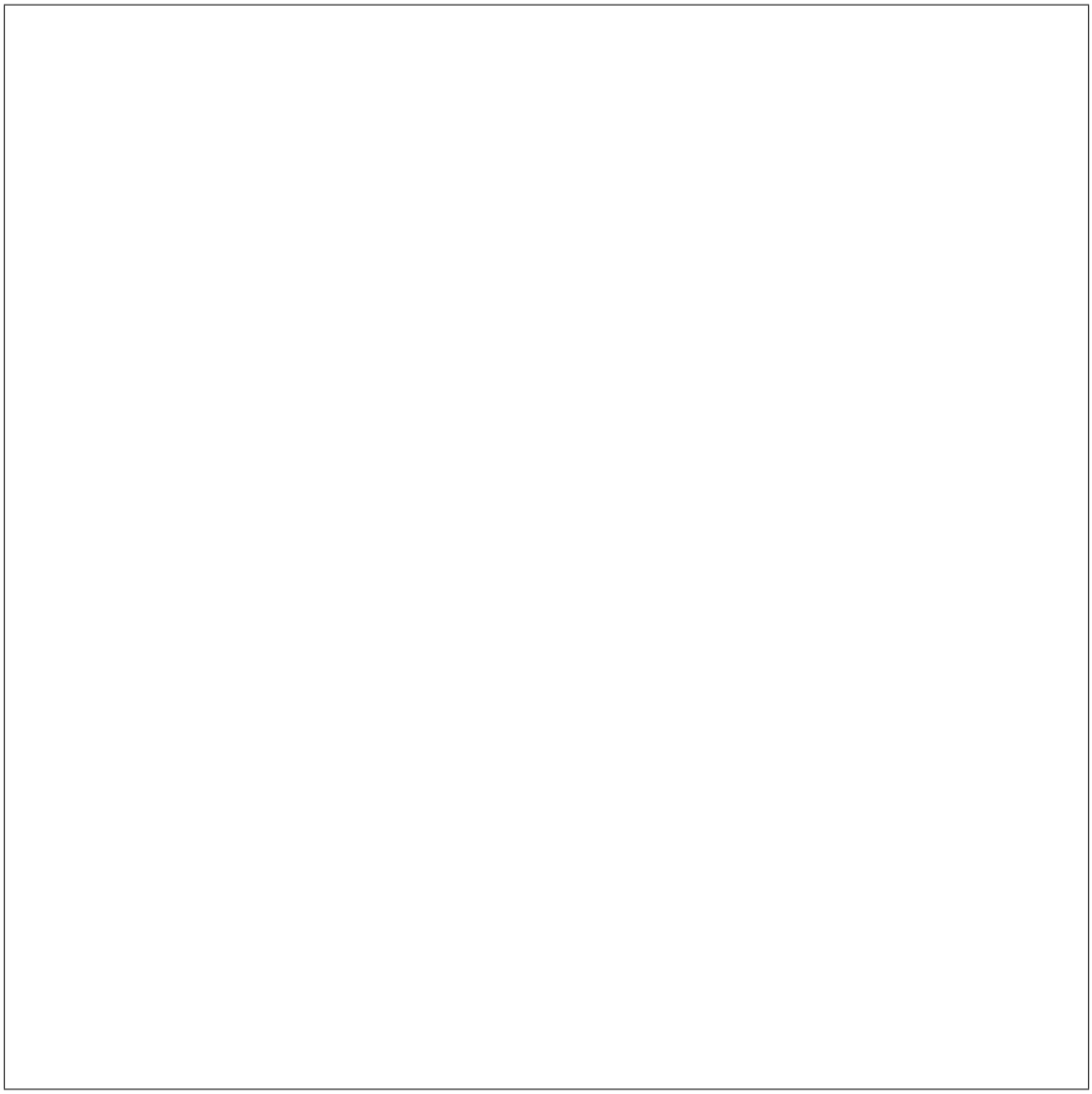
c) Calculate the relative frequency of occurrence of the event A using the vector generated for the observed outcome at the beginning of this experiment?

d) Assume each outcome in the sample space is equally likely. Find the probability of the event A? Compare it to the value obtained in part **c**.

Experiment 3:

In this experiment we want to generate a vector y of 1000 random numbers whose elements are drawn from the interval (0,1), where each element from the interval (0,1) is drawn with equal likelihood.

a) Plot the histogram of the 1000 numbers generated in vector y?



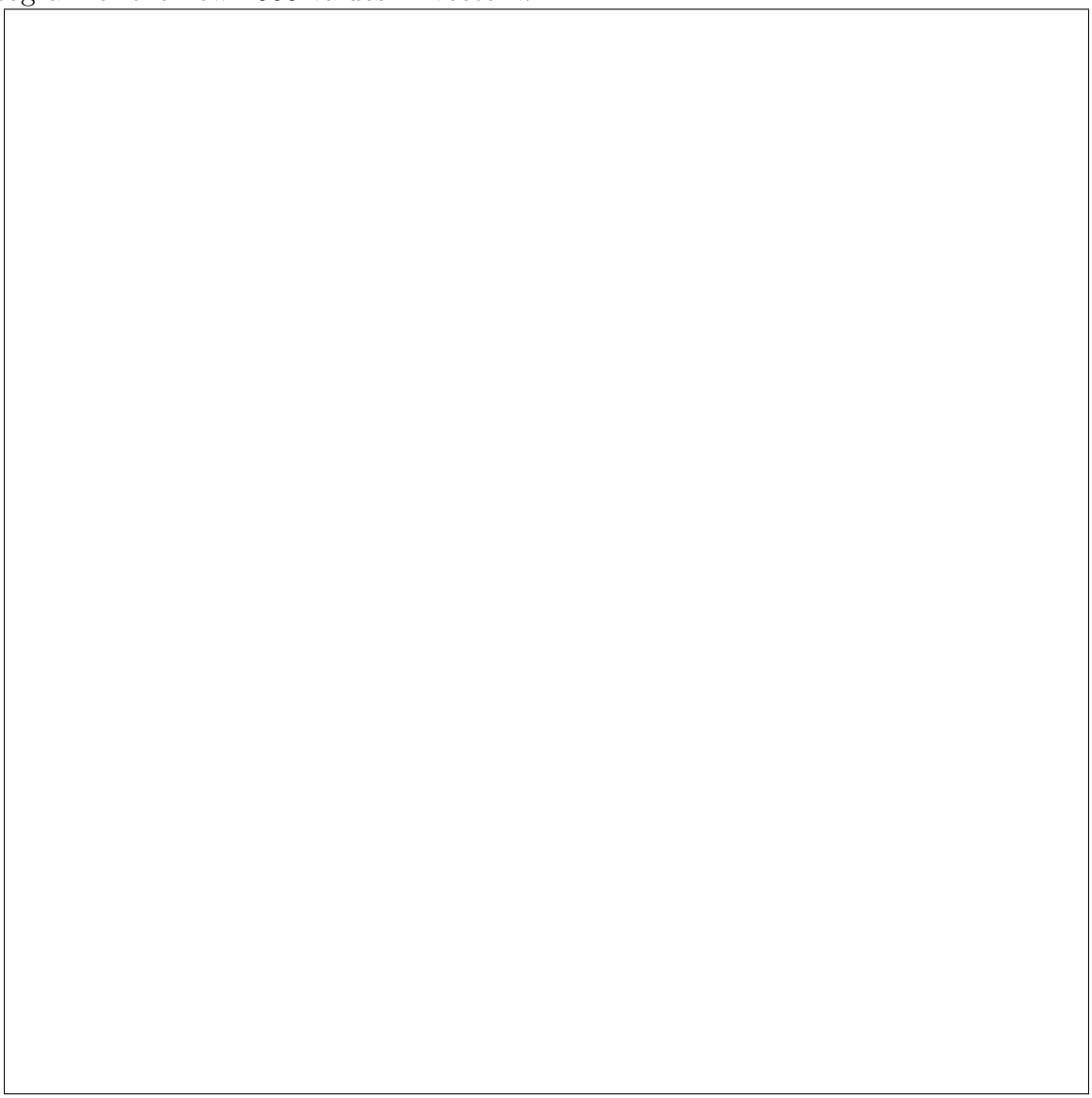
b) Compute the average of these 1000 values? (Hint: use the subroutine "`sum`" in Matlab).

c) Let B be defined as the event of obtaining a number between 0.2 and 0.3 (i.e. $B = [0.2, 0.3]$), find the relative frequency of B?

d) Now find the exact value of the probability of the event B (based on the uniform assumption)? Compare it to the value obtained in part **c**?

e) Suppose we wish to create a new vector z, by simply scaling each element of the original

vector y by 1.5 and taking the modulo 1 of the result (i.e. $z = 1.5y(\mod 1)$). Plot the histogram of the new 1000 values in vector z.



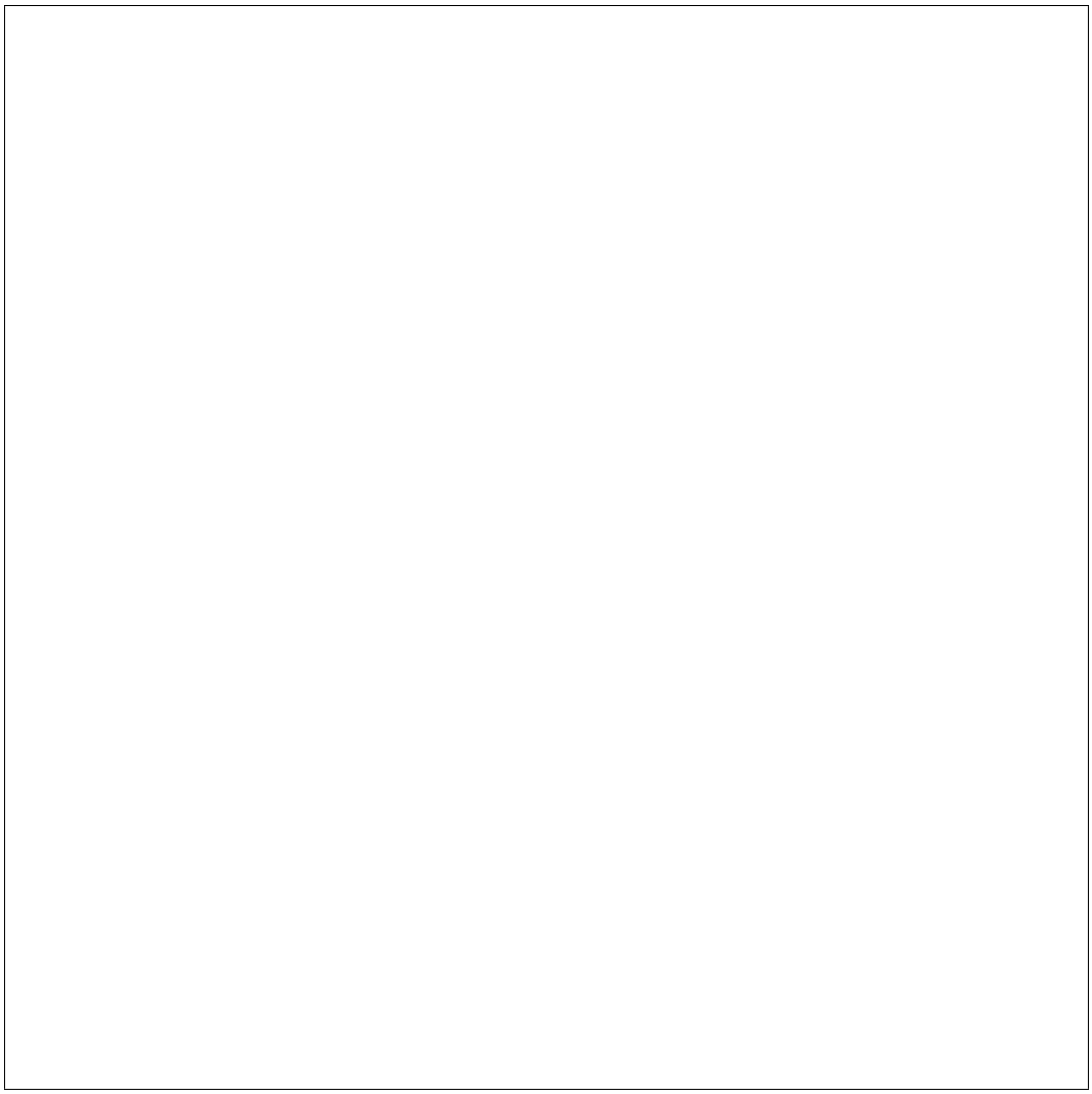
f) Find the average of the 1000 values contained in vector z.

Experiment 4:

Consider the experiment of picking any two numbers, say x_0 and y_0, at random from the interval $(0, 1)$ under the assumption of equal-likelihood assignment of probability on interval $(0, 1)$.

a) What is the sample space (note that the sample space is continuous)?

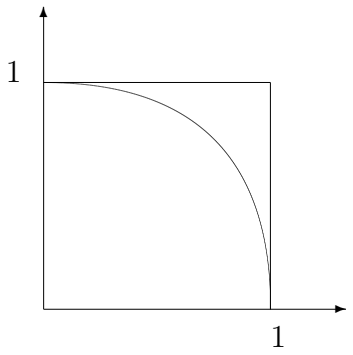
b) Using Matlab, generate two sequences, say x and y, of 1000 uniformly random numbers in the interval $(0, 1)$. Plot the 2-D scatter plot of the sequences x and y, which is a plot of the values of y versus the corresponding values of x, and show it to the TA. (Hint: Use subroutine "`scatter`". See Appendix A)



c) Use the plot in part **a** to find $P(x \leq 0.25)$, $P(y \leq 0.1)$ and $P(y - 2x \geq 0)$.

We will now find an approximate value of $\pi = 3.14159...$ using simulation in Matlab. We will do this in the following way:

Consider a square that has one corner at the origin of the X-Y plane and has sides of length 1 - it will obviously have an area of 1. Now consider inscribing a quarter of a circle inside of this with a radius of 1 – we know that its area is $\pi/4$ as shown in the figure below.



In Matlab we can generate thousands of random (X,Y) positions and determine whether each of them are inside of the circle. Each time, if the point is inside the circle, we will add one to a counter. After generating a large number of points, the ratio of the number of points inside the circle to the total number of points (N) generated will approach the ratio

of the area of the circle to the area of the square. Thus the value of π would simply be:

$$\pi = \lim_{N\to\infty} 4 \times f_N = \lim_{N\to\infty} 4 \times \frac{\text{Number of points inside quarter circle}}{N}$$

i.e.

$$\pi \approx 4 \times \frac{\text{Number of points inside quarter circle}}{\text{number of points inside square}}$$

d) Write a Matlab code to simulate and calculate the approximate value of π.

e) Plot the estimation of π as a function of the number of points N.

Appendix A

Introduction to Matlab

Matlab (short for MATrix LABoratory) is a language for technical computing. It provides a single platform for computation, visualization, programming and software development. All problems and solutions in Matlab are expressed in notation used in linear algebra and essentially involve operations using matrices and vectors.
As part of the undergraduate Electrical Engineering program, you will be using Matlab to solve problems in

- Circuits
- Communication systems
- Digital signal processing
- Control systems
- Probability and statistics

Online help can be accessed for all Matlab commands by issuing the "`help`" command.

```
>> help <type name of command here>
```

To get started, you can simply type

```
>> help
```

Vectors and Matrices

Variables in Matlab are just like variables in any other programming language (C, C++ etc.); only difference is that you do not have to define them by indicating the type etc. Also, variable names (case sensitive) can be used to refer to a single number (a scalar), a set of numbers (a vector) or an array of numbers (a matrix). Vectors are nothing but matrices having a single row (a row vector), or a single column (a column vector).

To create a row vector in Matlab, do:

```
>> r = [1 2 3 4]
r =
        1     2     3     4
```

A column vector can be created by

```
>> c = [1; 2; 3; 4]
c =
     1
     2
     3
     4
```

Vectors can also be created by incrementing a starting value with a constant quantity. For example,

```
>> r = [0:2:10]
r =
     0     2     4     6     8    10
```

creates a row vector, with the first element = 0; each element incremented by 2; until the final value of 10. You can access specific parts of a vector. For example, to get the third element in the vector r, you can do

```
>> r(3)
ans =
     4
```

Histogram Plot

Synopsis

```
hist
hist(y)
hist(y,nb)
hist(y,x)
[n,x] = hist(y,...)
```

Description: `hist` calculates or plots histograms.

`hist(y)` draws a 10-bin histogram for the data in vector y. The bins are equally spaced between the minimum and maximum values in y.

`hist(y,nb)` draws a histogram with nb bins.

`hist(y,x)`, if x is a vector, draws a histogram using the bins specified in x.

`[n,x] = hist(y)`, `[n,x] = hist(y,nb)`, and `[n,x] = hist(y,x)` do not draw graphs, but return vectors n and x containing the frequency counts and the bin locations such that `bar(x,n)` plots the histogram. This is useful in situations where more control is needed over the appearance of a graph, for example, to combine a histogram into a more elaborate plot.

Example: Generate bell-curve histograms from Gaussian data.

```
>> x = -2.9:0.1:2.9;
>> y = randn(10000,1);
>> hist(y,x)
```

Loops

The for loop (very similar to C expression) is a simple command for setting up a loop.

Example:

```
>> for i = 1:10;
>> a(i) = i*i;
>> end
>> a
a =
     1     4     9    16    25    36    49    64    81   100
```

All statements between the for and the end statements will be executed as per the command specifications. Example of a for loop where the increment is not 1 would be

```
>> for i = 1:3:20;
```

Type in

```
>> help for
```

for more details.

Help Function

`scatter`

Description: `scatter(X,Y)` displays colored circles at the locations specified by the vectors X and Y (which must be the same size).

rand

Uniformly distributed random numbers and arrays

Syntax

```
Y = rand(n)
Y = rand(m,n)
Y = rand([m n])
Y = rand(m,n,p,...)
Y = rand([m n p...])
Y = rand(size(A))
rand
```

Description: The `rand` function generates arrays of random numbers whose elements are uniformly distributed in the interval (0,1). `Y = rand(n)` returns an n-by-n matrix of random entries. An error message appears if n is not a scalar. `Y = rand(m,n)` or `Y = rand([m n])` returns an m-by-n matrix of random entries. `Y = rand(m,n,p,...)` or `Y = rand([m n p...])` generates random arrays. `Y = rand(size(A))` returns an array of random entries that is the same size as A. `rand`, by itself, returns a scalar whose value changes each time it's referenced.

plot

Linear 2-D plot

Syntax

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
plot(...,'PropertyName',PropertyValue,...)
h = plot(...)
```

Description: `plot(Y)` plots the columns of Y versus their index if Y is a real number. If Y is complex, `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`. In all other uses of plot, the imaginary component is ignored. `plot(X1,Y1,...)` plots all lines defined by Xn versus Yn pairs. If only Xn or Yn is a matrix, the vector is plotted versus the rows or columns of the matrix, depending on whether the vector's row or column dimension matches the matrix.

`sum`

Sum of array elements

Syntax

```
B = sum(A)
B = sum(A,dim)
```

Description: `B = sum(A)` returns sums along different dimensions of an array. If A is a vector, `sum(A)` returns the sum of the elements. If A is a matrix, `sum(A)` treats the columns of A as vectors, returning a row vector of the sums of each column. If A is a multidimensional array, `sum(A)` treats the values along the first non-singleton dimension as vectors, returning an array of row vectors.

Executable Files

Executable files in Matlab are generated by storing a list of Matlab commands in a file given the extension `.m`

These files are called M-files. To create an M-file, use the "New...M-file" Option under the "File Menu" in the "Command Window". Type in the following commands in the M-File Editor Window:

```
x = [0:0.1:10];
y = cos(x);
plot(x,y)
xlabel('x')
ylabel('y')
title('A plot of Cosine(x)')
```

Save the file using the "Save Option" under the "File Menu" in the M-File Editor Window. Call it, say, `cosineplot.m`

Now, to run this program in Matlab, move over to the Matlab "Command Window" and just type in

```
>> cosineplot
```

Remember, MATLAB IS EASY!!!!! GOOD LUCK :-)