EXPERIMENT # 2

OVERVIEW:

1. Plotting elementary functions
2. Product, division and power of vectors
3. Matrices- 2D arrays

1. Plotting Elementary Functions

Suppose we wish to plot a graph of y = sin3πx for $0 \leq x \leq 1$. We do this by sampling the function at a sufficiently large number of points and then joining up the points (x, y) by straight lines. Suppose we take N+1 points equally spaced a distance h apart:

>> N =10; h =1/N; x =0:h:1;

defines the set of points x = 0, h, 2h, . . ., 9h, 1.The corresponding y values are computed by

>> y =sin(3*pi*x);

and finally, we can plot the points with

>> plot(x,y)

The result is shown in Figure 1, where it is clear that the value of N is too small.

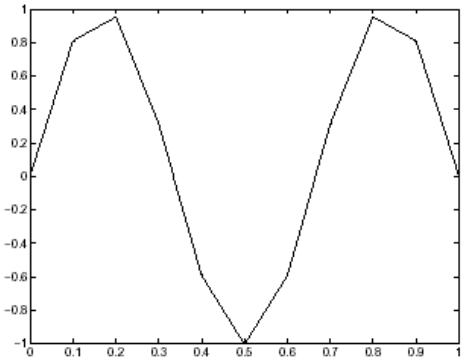


Figure 1: Graph of $y = \sin 3\pi x$ for $0 \leq x \leq 1$ using $h = 0.1$.

On changing the value of N to 100:

>> N =100; h =1/N; x =0:h:1;

>> y =sin(3*pi*x); plot(x,y)

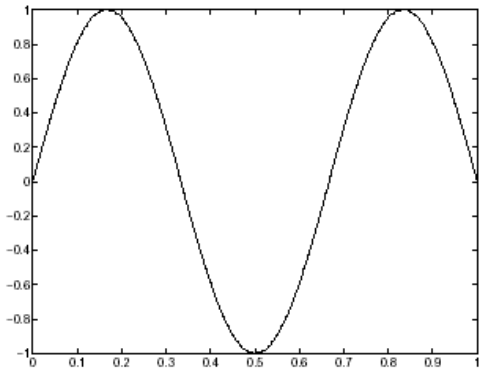


Figure 2: Graph of $y = \sin 3\pi x$ for $0 \leq x \leq 1$ using $h = 0.01$.

1.1 Plotting—Titles & Labels

To put a title and label the axes, we use
>> title('Graph of y =sin(3pi x)')
>> xlabel('x axis')
>> ylabel('y-axis')
The strings enclosed in single quotes, can be anything of our choosing.

1.2 Grids

A dotted grid may be added by
>> grid
This can be removed using either grid again, or grid off.

1.3 Line Styles & Colors

The default is to plot solid lines. A solid white line is produced by
>> plot(x, y, 'w-')
The third argument is a string whose first character specifies the color (optional) and the second the line style. The options for colors and styles are:

Colours		Line Styles	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green	-	solid
b	blue	*	star
w	white	:	dotted
k	black	-.	dashdot
		--	dashed

1.4 Multi–plots

Several graphs can be drawn on the same figure as:

>> plot(x,y,'w-',x,cos(2*pi*x),'g--')

A descriptive legend may be included with

>> legend('Sin curve','Cos curve')

which will give a list of line–styles, as they appeared in the plot command, followed by a brief description.

Matlab fits the legend in a suitable position, so as not to conceal the graphs whenever possible. For further information do help plot etc.

The result of the commands

>> plot(x,y,'w-',x,cos(2*pi*x),'g--')
>> legend('Sin curve','Cos curve')
>> title('Multi-plot ')
>> xlabel('x axis'), ylabel('y axis')
>>grid

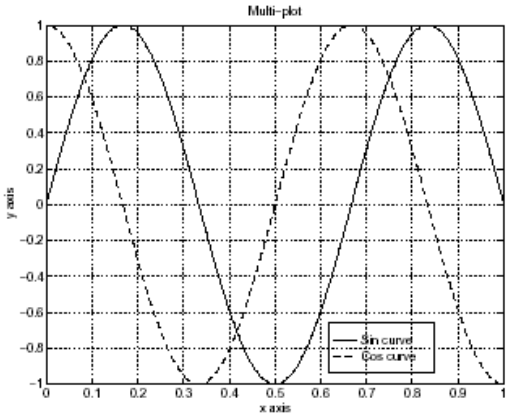


Figure 3: Graph of $y = \sin 3\pi x$ and $y = \cos 3\pi x$ for $0 \leq x \leq 1$ using $h = 0.01$.

1.5 Hold

A call to plot clears the graphics window before plotting the current graph. This is not convenient if we wish to add further graphics to the figure at some later stage. To stop the window being cleared:

>> plot(x,y,'w-'), hold
>> plot(x,y,'gx'), hold off

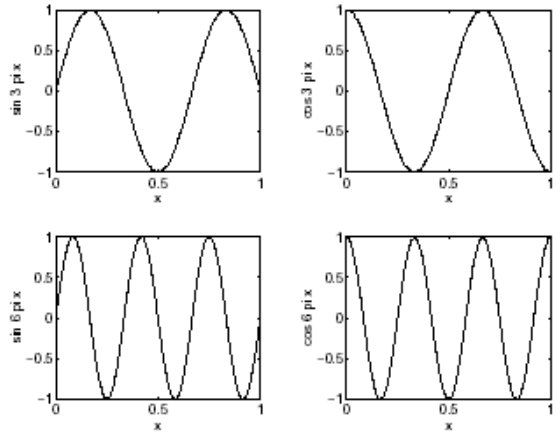
"hold on" holds the current picture; "hold off" releases it (but does not clear the window, which can be done with clf). "hold" on its own toggles the hold state.

1.6 Subplot

The graphics window may be split into an m × n array of smaller windows into which we may plot one or more graphs. The windows are counted 1to mn row–wise, starting from the top left. Both hold and grid work on the current subplot.

>> subplot(221), plot(x,y)
>> xlabel('x'),ylabel('sin 3 pi x')
>> subplot(222), plot(x,cos(3*pi*x))
>> xlabel('x'),ylabel('cos 3 pi x')
>> subplot(223), plot(x,sin(6*pi*x))
>> xlabel('x'),ylabel('sin 6 pi x')
>> subplot(224), plot(x,cos(6*pi*x))
>> xlabel('x'),ylabel('cos 6 pi x')

subplot(221) (or subplot(2,2,1)) specifies that the window should be split into a 2 × 2 array and we select the first sub-window.



1.7 Zooming

We often need to "zoom in" on some portion of a plot in order to see more detail. This is easily achieved using the command

>> zoom

Left-Click on plot = Zoom in
Right-Click on plot = Zoom out
Selecting a specific portion of the plot using the mouse will zoom in to the selected portion of the plot.

1.8 Controlling Axes

Once a plot has been created in the graphics window you may wish to change the range of x and y values shown on the picture.

>> clf, N =100; h =1/N; x =0:h:1;
>> y =sin(3*pi*x); plot(x,y)
>> axis([-0.5 1.5 -1.2 1.2]), grid

The axis command has four parameters, the first two are the minimum and maximum values of x to use on the axis and the last two are the minimum and maximum values of y.
Note the square brackets. The result of these commands is shown in Figure 4. For more info, check out help axis.

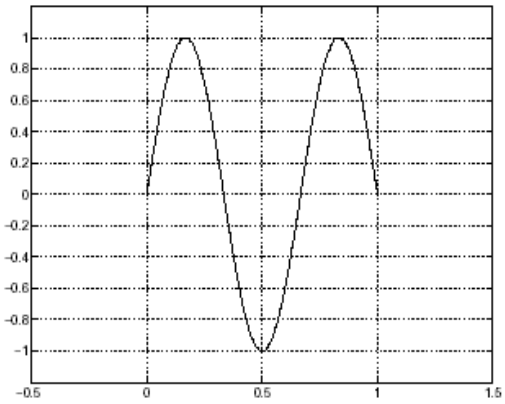


Figure 4: The effect of changing the axes of a plot.

2. Products, Division & Powers of Vectors

2.1 Scalar Product (*)

We shall describe two ways in which a meaning may be attributed to the product of two vectors. In both cases the vectors concerned must have the same length.
The first product is the standard scalar product. Suppose that u and v are two vectors of length n, u being a **row** vector and v a **column** vector:

$$\underline{u} = [u_1, u_2, \ldots, u_n], \qquad \underline{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

The scalar product is defined by multiplying the corresponding elements together and adding the results to give a single number (scalar).

$$\underline{u}\,\underline{v} = \sum_{i=1}^{n} u_i v_i.$$

For example, if $\underline{u} = [10, \quad 11, 12]$, and $\underline{v} = \begin{bmatrix} 20 \\ 21 \\ 22 \end{bmatrix}$

then $n = 3$ and

$$\underline{u}\,\underline{v} = 10 \times 20 + (\quad 11) \times (\quad 21) + 12 \times (\quad 22) = 167.$$

We can perform this product in Matlab by
>> u =[10, -11, 12], v =[20; -21; -22]
>> prod =u*v % row times column vector
Suppose we also define a row vector w and a column vector z by
>> w = [2, 1, 3], z = [7; 6; 5]
and we wish to form the scalar products of u with w and v with z.
>> u*w

??? Error using ==> *
Inner matrix dimensions must agree.
An error results because w is not a column vector.
Recall that transposing (with **'**) turns column vectors into row vectors and vice versa. So, to form the scalar product of two row vectors or two column vectors,

>> u*w' **% u & w are row vectors**

>> u*u' **% u is a row vector**

>> v'*z **% v & z are column vectors**

2.2 Dot Product (.*)

The second way of forming the product of two vectors of the same length is known as the Dot product. It is not often used in Mathematics but frequently used in Signals & Systems. It involves vectors of the same type. If u and v are two vectors of the same type (both row vectors or both column vectors), the mathematical definition of this product, is the **vector** having the components:

$$\underline{u} \cdot \underline{v} = [u_1 v_1, u_2 v_2, \ldots, u_n v_n].$$

The result is a vector of the same length and type as u and v. Thus, we simply multiply the corresponding elements of two vectors. In Matlab, the product is computed with the operator **.*** and, using the vectors u, v, w, z defined previously,
>> **u.*w**

>> **u.*v'**

>> **v.*z, u'.*v**

Example 2.1 *Tabulate the function y = x sinπx for x = 0, 0.25, . . ., 1.*
It is easier to deal with column vectors so we first define a vector of x-values: (see Transposing)
>> **x =(0:0.25:1)';**
To evaluate y we have to multiply each element of the vector x by the corresponding element of the vector sin πx:
x × sin πx = x sin πx
0 × 0 = 0
0.2500 × 0.7071 = 0.1768
0.5000 × 1.0000 = 0.5000
0.7500 × 0.7071 = 0.5303
1.0000 × 0.0000 = 0.0000
To carry this out in Matlab:
>> **y =x.*sin(pi*x)**

Note: **a)** the use of pi, **b)** x and sin(pi*x) are both column vectors (the sin function is applied to each element of the vector). Thus, the dot product of these is also a column vector.

2.3 Dot Division of Arrays (./)

There is no mathematical definition for the division of one vector by another. However, in Matlab, the operator **./** is defined to give element by element division—it is therefore only defined for vectors of the same size and type.

>> **a =1:5, b =6:10, a./b**

>> **a./a**

>> **c =-2:2, a./c**

The previous calculation required division by 0 --- notice the **Inf**, denoting infinity, in the answer.

>> **a.*b -24, ans./c**

2.4 Dot Power of Arrays (.^)

To square each of the elements of a vector we could, for example, do u.*u. However, a neater way is to use the .^ operator:

>> **u.^2**
>> **u.*u**
>> **u.^4**
>> **v.^2**
>> **u.*w.^(-2)**

Recall that powers (**.^** in this case) are done first, before any other arithmetic operation.

3. Matrices --- 2-DArrays

Row and Column vectors are special cases of **matrices**. An m×n matrix is a rectangular array of numbers having m rows and n columns. It is usual in a mathematical setting to include the matrix in either round or square brackets—we shall use square ones. For example, when m = 2, n = 3 we have a 2 × 3 matrix. To enter such an matrix into Matlab we type it in row by row using the same syntax as for vectors:

>> A =[5 7 9;1 -3 -7]

Rows can be separated by a new line rather than a semicolon.

>> B = [-1 2 5
9 0 5]

>> C = [0, 1; 3, -2; 4, 2]
>> D =[1:5; 6:10; 11:2:20]
So A and B are 2 × 3 matrices, C is 3 × 2 and D is
3 × 5.
In this context, a row vector is a 1 × n matrix and a column vector a m ×1 matrix.

3.1 Size of a matrix
We can get the size (dimensions) of a matrix with the command size
>> **size(A), size(x)**

>> **size(ans)**

So A is 2×3 and x is 3 ×1 (a column vector). The last command size(ans) shows that the value returned by size is itself a 1 × 2 matrix (a row vector). We can save the results for use in subsequent calculations.

>> **[r c] =size(A'), S =size(A')**

3.2 Transpose of a matrix

Transposing a vector changes it from a row to a column vector and vice versa. The extension of this idea to matrices is that transposing interchanges rows with the corresponding columns:
The 1st row becomes the 1st column, and so on.
>> **D, D'**

>> **size(D), size(D')**

3.3 Special Matrices

Matlab provides a number of useful built–in matrices of any desired size. ones(m,n) gives an m × n matrix of 1's,
>> P =ones(2,3)

zeros(m,n) gives an m × n matrix of 0's,
>> Z =zeros(2,3), zeros(size(P'))

The second command illustrates how we can construct a matrix based on the size of an existing one. **Try ones(size(D))**

An n×n matrix that has the same number of rows and columns is called a **square** matrix.
A matrix is said to be **symmetric** if it is equal to its transpose (i.e. it is unchanged by transposition):
>> S =[2 -1 0; -1 2 -1; 0 -1 2],

>> St =S'

>> S-St

3.4 The Identity Matrix

The n × n **identity** matrix is a matrix of zeros except for having ones along its leading diagonal (top left to bottom right). This is called eye(n) in Matlab (since mathematically it is usually denoted by I).
>> I =eye(3), x = [8; -4; 1], I*x

Notice that multiplying the 3 × 1 vector x by the 3 × 3 identity I has no effect (it is like multiplying a number by 1).

3.5 Diagonal Matrices

A diagonal matrix is similar to the identity matrix except that its diagonal entries are not necessarily equal to 1.
D =
3 0 0
0 4 0
0 0 2
is a 3 × 3 diagonal matrix. To construct this in Matlab, we could either type it in directly
>> D = [-3 0 0; 0 4 0; 0 0 2]

But this becomes impractical when the dimension is large (e.g. a 100 × 100 diagonal matrix). We then use the **diag** function. We first define a vector d, say, containing the values of the diagonal entries (in order) then diag(d) gives the required matrix.

>> d =[-3 4 2], D =diag(d)

On the other hand, if A is any matrix, the command diag(A) extracts its diagonal entries:
>> F = [0 1 8 7; 3 -2 -4 2; 4 2 1 1]
>> diag(F)

Notice that the matrix does not have to be square.

3.6 Building Matrices

It is often convenient to build large matrices from smaller ones:

>> C=[0 1; 3 -2; 4 2]; x=[8;-4;1];
>> G =[C x]

>> A, B, H =[A; B]

so we have added an extra column (x) to C in order to form G and have stacked A and B on top of each other to form H.

>> J =[1:4; 5:8; 9:12; 20 0 5 4]

>> K =[diag(1:4) J; J' zeros(4,4)]

The command spy(K) will produce a graphical display of the location of the nonzero entries in K (it will also give a value for nz—the number of nonzero entries):

>> spy(K), grid

3.7 Extracting Bits of Matrices

We may extract sections from a matrix in much the same way as for a vector. Each element of a matrix is indexed according to which row and column it belongs to. The entry in the ith row and jth column is denoted mathematically by Ai,j and, in Matlab, by A(i,j). So
>> J
>> J(1,1)

>> J(2,3)

>> J(4,3)

>> J(4,5)

>> J(4,1) =J(1,1) + 6
>> J(1,1) =J(1,1) - 3*J(1,2)

In the following examples we extract
i) the 3rd column,
ii) the 2nd and 3rd columns,
iii) the 4th row,
and
iv) the "central" 2 ×2 matrix
>> J(:,3) % 3rd column

>> J(:,2:3) % columns 2 to 3

>> J(4,:) % 4th row

>> J(2:3,2:3) % rows 2 to 3 & cols 2 to 3

Thus, **:** on its own refers to the entire column or row depending on whether it is the first or the second index.

3.8 Dot product of matrices (.*)

The dot product works as for vectors: corresponding elements are multiplied together—so the matrices involved must have the same size.
>> A, B
A =
5 7 9
1 -3 -7
B =
-1 2 5
9 0 5

>> A.*B

>> A.*C

>> A.*C'

3.9 Matrix–Matrix Products

To form the product of an m × n matrix A and a n×p matrix B, written as AB, we visualize the first matrix (A) as being composed of m row vectors of length n stacked on top of each other while the second (B) is visualized as being made up of p column vectors of length n:
(m× n) times (n ×p) = (m × p).
Check that you understand what is meant by working out the following examples by hand and comparing with the Matlab answers.
>> A =[5 7 9; 1 -3 -7]
>> B =[0, 1; 3, -2; 4, 2]
>> C =A*B

>> D =B*A

>> E =B'*A'

We see that E = C' suggesting that (A*B)' = B'*A'

Exercise

Draw the following:

1. x(t) = 2 sin (2πt - π/2) for 0 ≤ t ≤ 4
2. *Draw graphs of the functions*
   y = cos(x)
   y = x
   for 0 ≤ x ≤ 2 in the same window. Use the zoom facility to determine the point of intersection of the two curves (and, hence, the root of x = cos(x)) to two significant figures.

3. Draw graphs of the functions for x =0:0.1:10 and label your graph properly.

i. $y = \sin(x)/x$

ii. $u = (1/(x-1)^2)+x$

iii. $v = (x^2+1)/(x^2-4)$

iv. $z = ((10-x)^{1/3}-1)/(4 - x^2)^{1/2}$