KICK OFF MATLAB

Marco Aiolfi Antonio Falanga

Working and plotting Matlab variables

1. Matrix and Array operations

The following operations are available: addition (**+**), subtraction (**-**), multiplication (*****), power (**^**), transpose (’), left division (**n**), right division (/).These arithmetic operators are interpreted in a matrix sense. If the matrices sizes are incompatible for the requested operation, you get an error message.
Array operations simply act identically on each element of an array. We have already seen some array operations, namely + and -. But *,’,ˆ,and / have particular matrix interpretations. To get an element wise behavior, precede the operator with a dot.
Some examples:
Using always the previous matrices we can get:

```
>> A*B'

ans =

   14
   32
   50
```

or

```
>> A.*[B;B;B]

ans =

    1    4    9
    4   10   18
    7   16   27
```

The following lines show the difference between the matrix product and the element-wise product:

```
>> a=magic(2)

a =
1 3
4 2

>> a*a
```

```
ans =
13 9
12 16

>> a.*a
ans =
1 9
16 4
```

2. Scalar functions

Some functions operate essentially on scalars, and operate element wise when applied to a matrix: *sin, cos, tan, asin, acos, atan, exp, log* (natural log), *abs, sqrt, sign, round, floor, ceil.* For example:

```
>> b=randn(4)

b =
1.0668 0.2944 -0.6918 -1.4410
0.0593 -1.3362 0.8580 0.5711
-0.0956 0.7143 1.2540 -0.3999
-0.8323 1.6236 -1.5937 0.6900

>> sign(b)

ans =
1 1 -1 -1
1 -1 1 1
-1 1 1 -1
-1 1 -1 1

>> abs(b)

ans =
1.0668 0.2944 0.6918 1.4410
0.0593 1.3362 0.8580 0.5711
0.0956 0.7143 1.2540 0.3999
0.8323 1.6236 1.5937 0.6900

>> round(b)

ans =
1 0 -1 -1
0 -1 1 1
0 1 1 0
-1 2 -2 1

>>exp(b)

ans =
2.9060 1.3423 0.5007 0.2367
```

1.0611 0.2628 2.3584 1.7703
0.9088 2.0428 3.5043 0.6704
0.4350 5.0711 0.2032 1.9937

3. Vector functions

The following functions operate essentially on a vector basis and for a MxN matrix they produce a row vector containing the results of their application to each column. These functions are: *max, min, sort, sum, prod, mean, median, std, any, all.*
The function *any* is true if any element of a vector is non zero;. The function *all* is true if all elements of a vector are nonzero.

```
>> c=randn(3,2)

c =
0.2573 -0.8051
-1.0565 0.5287
1.4151 0.2193

>> max(c)

ans =
1.4151 0.5287
```

We can use these vector functions along a specified dimension of a matrix.
In the first example we use the max function across rows, while in the second example is applied across columns.

```
>> max(c,[],1)

ans =
1.4151 0.5287

>> max(c,[],2)

ans =
0.2573
0.5287
1.4151

>> sort(c)

ans =
-1.0565 -0.8051
0.2573 0.2193
1.4151 0.5287

>> d=diag([1,2,3])

d =
1 0 0
0 2 0
```

```
0 0 3

>> any(d)

ans =
1 1 1

>> all(d)

ans =
0 0 0
```

4. Matrix functions

Matrix functions are functions operating on matrix variable. These are: *inv* (inverse), *chol* (choleski decomposition), *eig* (eigenvalues and eigenvectors), *det* (determinant), *rank* (rank), *cond* (condition number), *size* (size), *flipdim* (flip matrix along specified dimension)

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

ans =
4 9 2
3 5 7
8 1 6

>> flipdim(a,2)

ans =
6 1 8
7 5 3
2 9 4

c =
0.2573 -0.8051
-1.0565 0.5287
6
1.4151 0.2193

>> size(c)

ans =
3 2

>> [m,n]=size(c)

m = 3 n= 2
```

5. Graphics

Graphical display is one of Matlab®'s greatest strengths—and most complicated subjects. The basics are quite simple, but you also get complete control over practically every aspect of each graph, and with that power comes complexity. Graphical objects are classified by type. Each figure has its own window. Inside a figure is one or more axes. You can make an existing figure or axes "current" by clicking on it. Inside the axes are drawn data-bearing objects like lines and surfaces. While there are functions called line and surface, you will very rarely use those. Instead you use friendlier functions that create these object types.

Matlab® can produce 2D, 3D plots and surface plots. With the command *figure* you open a window separate from the command window to put your plot. If y is a vector, *plot(y)* produces a linear graph of the elements of y versus the index of the elements of y. If y is a matrix *plot(y)* produces a graph of the elements of each column of y against the index of the elements of y. If you specify two vectors as arguments, *plot(x,y)* produces a graph of y versus x. You can plot multiple lines (like *y, z, w*) in the same plot by using *plot(x, y, z)*.
You can display multiple plots in the same figure window and print them on the same piece of paper with the subplot function. *subplot(m, n, i)* breaks the figure window into an m-by-n matrix of small subplots and selects the ith subplot for the current plot. The plots are numbered along the top row of the figure window, then the second row, and so forth.
You can edit all the elements of the plot (axis scale, linewidth, linecolor, etc) or insert a title or a legend in your figure. For further details see the help.
To export the current or most recently active figure, type *print – fileformat filename* where *fileformat* is a graphics format supported by Matlab® and filename is the name you want to give to the exported file.

The process of constructing a basic graph to meet your presentation graphics requirements is outlined in the following table. The table shows seven typical steps and some example code for each. If you are performing analysis only, you may want to view various graphs just to explore your data. In this case, steps 1 and 3 may be all you need. If you are creating presentation graphics, you may want to fine-tune your graph by positioning it on the page, setting line styles and colors, adding annotations, and making other such improvements.

Step Typical Code

- Prepare your data x = 0:0.2:12;
  y1 = bessel(1,x);
  y2 = bessel(2,x);
  y3 = bessel(3,x);
- Select a window and position a plot region within the window
  figure(1)
  subplot(2,2,1)
- Call elementary plotting function
  h = plot(x,y1,x,y2,x,y3);
- Select line and marker characteristics
  set(h,'LineWidth',2,{'LineStyle'},{'';':';'-.'})
  set(h,{'Color'},{'r';'g';'b'})

- Set axis limits, tick marks, and grid lines

  axis([0 12 –0.5 1])

  grid on
- Annotate the graph with axis labels, legend, and text

  xlabel('Time')

  ylabel('Amplitude')

  legend(h,'First','Second','Third')

  title('Bessel Functions')

  [y,ix] = min(y1);

  text(x(ix),y,'First Min \rightarrow',... 'HorizontalAlignment','right')
- Export graph

  print –depsc -tiff -r200 myplot