Introduction to matlab for EE128 students

Takis P. Konstantopoulos

1 Accessing matlab

There are currently two versions of matlab: The 87 version and the 90 version. To access any of them, use any of the workstations of the *ara* cluster in room 119. Say you are using *hydrus*. You have to rlogin to *buddy* and run the program there. Note that you are using X-Windows.

Do the following steps:
1. Type *xhost buddy*.
2. Type *rlogin buddy*. You will be asked for login name and password.
3. When in buddy, type *setenv DISPLAY hydrus:0.0*. This will specify the correct display environment so that you can make plots.
4. Create a directory called matlab by typing *mkdir matlab*.
5. It is best to run matlab from the above directory, so type *cd matlab*.
6a. To use the 90 version type
/usr/local/matlab3.5.
6b. To use the 87 version type
source /c/matlab3.26/matlab def,
and then *matlab*.
7. Once you are inside matlab, you have to specify the terminal type by typing *terminal*. You will then be presented with a list of options. Select *xterm* by typing the corresponding number.

I am giving you the procedures for both versions as, for the time being, there seem to be some problems with the new version, namely buddy cannot be recognized as an xhost by the ara machines. This is a system error that will hopefully be fixed soon.
Any questions should be addressed to root. Just send mail to *root@buddy*.

2 Description of basic operations

Matlab is basically software for numerical operations using matrices. I suggest that you run *demo* and examine carefully what is being done there. You'll be able to pick up quickly the basic commands. Besides the basic commands of matlab we will use the commands of the control system toolbox. You will be provided with a list for both sets of commands. You can get information on the use of a specific command by typing *help command*. E.g. *help* *roots* will tell you how to find polynomial roots. Similarly, *help +* will tell you which things you can add. Also *help intro* will give you a brief introduction. Below you will find some examples of various commands:

2.1 Matrices

Enter a matrix A row-wise by typing $A = [1\ 2;\ 3\ 4]$.
Add matrices by $A + B$.
Multiply by $A * B$.
Compute A^{-1} by *inv(A)*.
Eigenvalues of A: *eig(A)*.
Eigenvalues and eigenvectors of A: *[a,b]=eig(A)*.
Exponential of A: *exp(A)*.
Transpose of A: *A'*.

2.2 Functions

Special functions do exist: *cos, sin, log, exp, sqrt*, etc... Just try... If you are not sure about the use, type *help* and then the function name. Note that you can use numbers as well as matrices as arguments for any of the existing functions.

2.3 Polynomials

A polynomial is handled as a row vector. E.g., $d(s) = s^3 + 6s - 3.1$ is entered as $d =$ [1 0 6 -3.1]. The roots of $d(s) = 0$ are found by

$$roots(d)$$

To specify a transfer function $H(s) = b(s)/a(s)$, just enter a and b separately.

2.4 Plots

First specify range and stepsize of the independent variable, say t by

$$t = a : s : b;$$

This means that t ranges in the interval from a to b incrementing by steps of size s. Do not forget the semicolon at the end; else you'll get the whole list of t's.

To plot a function $f(t)$ type

$$plot(t, f(t))$$

For instance, $plot(t, t.^2)$ plots the function $f(t) = t^2$. Do not forget the dot (.) after the t in the second argument of $plot$. The reason is that t is thought of as a row. So to raise it to the power 2, you must specify that you want to do this component-wise. This is specified by the dot.

To plot the curve $x(t), y(t)$ (defined in parametric form), type

$$plot(x(t), y(t))$$

E.g., $plot(sin(3 * t), cos(5 * t))$ will produce an interesting Lissajous figure.

To plot a curve $\phi(t), \rho(t)$ in polar coordinates (t is a parameter), type

$$polar(\phi(t), \rho(t))$$

E.g., $polar(t, sqrt(t))$ will produce a "contracting" spiral.

To plot a function $z = f(x, y)$ of 2 arguments, first specify the range by

$$[x, y] = meshdom(a_1 : s_1 : b_1, a_2 : s_2 : b_2);$$

and then type

$$mesh(f(x, y))$$

Of course, and this is a general comment, you can have an intermediate step

$$z = f(x, y);$$

(if you forget the ; you'll get a screen full of numbers) before typing

$$mesh(z)$$

This may be useful if you want to re-use the expression z.

2.5 Impulse response

The impulse response of the transfer function $H(s) = b(s)/a(s)$ is computed over a time interval, say $[0, 10]$, as follows:
1. First enter the polynomials a and b.
2. Then enter $t = 0 : 0.01 : 10$; (I selected 0.01 as stepsize–it doesn't have to be so.)
3. Then compute the impulse response by $y = impulse(b, a, t)$.
4. Finally, you can plot the result by typing $plot(t, y)$.
4a. Or, $plot(t, y), title($'impulse response$')$, if you want to have a caption.

2.6 Step response

The response to a step input of size 1 is found bby following the previous rules and by substituting *impulse* by *step*.

2.7 Frequency response

It is customary to use logarithmic scale. Say w is the independent variable (frequency). Type

$$w = logspace(\alpha, \beta);$$

to generate 50 points between 10^α and 10^β equally spaced in a logarithmic scale. To generate a number n of points, not necessarily equal to 50, type

$$w = logspace(\alpha, \beta, n);$$

Compute the frequency response by

$$[mag, phase] = bode(b, a, w);$$

Remember that $H = b/a$ is our transfer function. Plot the magnitude on a log-log scale by

$$loglog(w, mag)$$

Plot the phase on a semi-log scale by

$$semilogx(w, phase)$$

The latter creates a log scale on the x axis and a linear scale on the y axis. The converse is achieved by the command *semilogy*.

2.8 Root locus

Say that $G(s) = b(s)/a(s)$ is the open loop transfer function. We want to plot the roots of the equation $1 + KG(s) = 0$ for various values of the gain K. First specify K as usual:

$$K = 0 : 0.1 : 2;$$

Then type

$$r = rlocus(b, a, K);$$

The closed loop poles are now contained in r. You can plot the root locus by typing

$$plot(r,'.')$$

You thus create a sequence of dots on the complex plane. If you wish stars, substitute the dot by a star.