# Introduction to MATLAB on the Region IV Computing Facilities

## 1 What is MATLAB?

MATLAB is a high-performance interactive software package for scientific and enginnering numeric computation. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment *without traditional programming.*

The name MATLAB stands for *matrix laboratory.* MATLAB was originally written to provide easy access to the matrix software developed by the LINPACK and EISPACK projects.

MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. Furthermore, problem solutions are expressed in MATLAB almost exactly as they are written mathematically.

MATLAB has evolved over the years with inputs from many users. MATLAB TOOLBOXES are specialized collections of MATLAB files designed for solving particular classes of functions. Currently available toolboxes include:

- signal processing
- neural networks
- $\mu$ analysis
- image processing
- control system
- system identification
- splines
- statistics
- optimization
- robust control
- symbolic mathematics

## 2 Accessing MATLAB at ER4

Sit down at a workstation. Log on.

```
login:  smithj  <press return>
password:  1234js  <press return>
```

Use the mouse button to view the Root Menu. Drag the cursor to "Design Tools" then to "Matlab." Release the mouse button. You are now running the MATLAB interactive software package. The prompt is a double "greater than" sign. You may use emacs line editor commands and the arrow keys while typing in MATLAB.

# 3 An Introductory Demonstration

Execute the following command to view a quick introduction to MATLAB.

```
>> intro
```

(Use your mouse to position windows on the screen for easy viewing.)

# 4 Making and Printing Graphs

| MATLAB commands introduced in this section: |
| --- |
| `plot, subplot, grid, title, xlabel, ylabel, stem, print, clg` |

| UNIX commands introduced in this section: |
| --- |
| `ghostview, lpr` |

## 4.1 Example signal

MATLAB represents objects as vectors and matrices. Therefore, a signal is represented by a vector of samples.
Generate 500 samples of a sine wave of frequency 60 Hz and sampled at 8 kHz.

```
>> n=[0:499]/8000;
>> x=sin(2*pi*60*n);
```

The time indices and signal amplitudes are stored in the row vectors `n` and `x`, respectively. Note that MATLAB returns the result of command unless output is suppressed by ending the command with a semicolon.

## 4.2 Plot commands

Creating graphs in MATLAB is easy.

```
>> plot(x)
```

You can specify two vectors of equal length in the plot command to specify both the horizontal and vertical axes

```
>> plot(n,x)
```

Note how the horizontal axes is now in units of seconds, rather than sample number. Let's title the graph, label the axes, and place a grid on the plot.

```
>> plot(n,x)
>> title('Title of plot')
>> xlabel('seconds')
>> ylabel('amplitude')
>> grid
```

By specifying elements of the vectors, you may plot a selected portion of the signal.

```
>> plot(n(1:50),x(1:50))
```

**Subplots**

We can place several graphs on a single page using the `subplot` command. The arguments in `subplot` specify the number of rows and columns, then the position of the plot.

```
>> subplot(2,1,1),plot(x,n)
>> title('plot a')
>> subplot(2,1,2),plot(x(1:50))
>> title('plot b')
```

**And much more!**

The MATLAB graphics tools provide options for line types, overlaid curves, data point symbols, line colors, etc. Use the help command to explore these commands:

```
plot, semilogx, semilogy, stem, hold, mesh, contour
```

For a colorful demo of MATLAB graphics, type `expo`. Note that the GUI (graphical user interface) tools to build the demo are part of MATLAB, and can be used to make menu-driven, point-and-click program interfaces for course work, teaching, and research projects.

## 4.3 Printing

To print your figure we will first create a file called "temp.ps" containing the plot, then view the file on the screen, the send the file to the printer. In this fashion, you can save files of figures you generate using MATLAB, and can view those files on your workstation screen, and can incorporate your MATLAB plots into documents generated using Latex, Word, Claris, etc.

```
>> print temp -f1
```

The "-f1" option selects the graph window labeled "Figure No. 1."
In your workstation window, type

```
smithj>  ghostview temp.ps
```

If a printout is desired, then the figure or the .ps file can be sent to the printer from MATLAB or the workstation window

| | |
|---|---|
| in MATLAB window | `>> print -f1` |
| in workstation window | `smithj> lpr temp.ps` |

To clear the figure in MATLAB and to return to plotting only one graph in a figure, type `clg` for "clear graph."

### 4.4 Importing data for graphing

Data generated by other programs can be ported into MATLAB for easy graphics or further computation. See Appendix 1 for an example program (courtesy Jeff Spooner).
Data generated by MATLAB can be stored, transferred and reloaded into MATLAB using files with the `.mat` suffix. For example, load and plot the canine electrocardiogram stored in the vector "ecg" in a file named canine.mat.

```
>>  load /user2/faculty/potter/canine
>>  plot(ecg)
```

You can use the `save` command to save data files while working in MATLAB (see `help save`).

## 5 Getting Help

On-line documentation in MATLAB is available using the `help` command. To have the screen scroll through long help files, use `more on`. To exit a scrolling help file, hit q. To disable the scrolling, use `more off`.

```
>>  more on
>>  help help
>>  help fft
>>  more off
```

# 6 M-files

MATLAB puts many commands at your disposal. Additionally, you can create your own commands or programs. You may wish to write a MATLAB program whenever you anticipate executing a sequence of statements several times or again at a later session. To create your own MATLAB program, use your favorite text editor and save the file with extension `.m` in the directory where you will run MATLAB (see `help chdir`). You many execute your m-file by typing the filenames (without the .m extension) at the command prompt (`>>`). There are two kinds of m-files: script files and functions.

## 6.1 Script Files

Executing a script file is exactly like typing the commands it contains at the command prompt. This is useful in executing a sequence of commands while composing and debugging a m-file.

## 6.2 Function Files

Functions have designated input and output variables. Any other variable used within a function are local variables, which do not remain after the function terminates. Many of the functions supplied in MATLAB are actually m-file functions. For example

```
>>  type sinc
```

**Avoid loops**
Since MATLAB is an interpreted language, do loops and for loops are very inefficient. Loops can often be avoided by using vectors and the following commands

```
tt sum   prod   .*   .^   :   toeplitz
```

For example, the sinusoid signal in Section 4.1 was generated without loops.

# 7 Controls Toolbox

The control toolbox contains a collection of MATLAB file designed for solving controls and linear systems problems. For reference, Appendix 2 contains a categorized listing of available commands. This list is available on-line

```
>>  more on; help control
```

An interesting demonstration routine is also available

```
>>  ctrldemo
>>  fourier
```

# 8 Signal Processing Toolbox

The signal processing toolbox contains a collection of MATLAB file designed for designing discrete and continuous time filters, filtering, statistical signal processing, spectrum analysis, and linear systems problems. For reference, Appendix 3 contains a categorized listing of available commands. This list is available on-line

```
>>  more on; help signal
```

Interesting demonstration routines are also available

```
>>  filtdemo
>>  moddemo
```

# 9 Finishing

To exit MATLAB

```
>> quit
```

Having quit MATLAB, delete any files you no longer need using the `rm` command, then use the Root Menu to logout.

# Appendix 1

### Using MATLAB to plot data generated by other programs

For the C language:

```
/* This file may be compiled using the format:
     gcc filename.c -o exefile -lm

   Running this program creates a file named "outvar.dat".  The
   data for a cosine and sine function is calculated and saved
   as three columns of numbers.

   To plot the data within MATLAB, you may use the following
   from a MATLAB window:

>> load outvar.dat
>> x = outvar(:,1);
>> y = outvar(:,2);
>> z = outvar(:,3);
>> plot(x,y)
>> hold on
>> plot(x,z)
>> hold off
*/

#include <stdio.h>        /* standard Input/Output */
#include <stdlib.h>       /* standard library      */
#include <math.h>         /* math library          */

main()
{
  double x,y,z;           /* variables used */
  FILE *fp;               /* file pointer   */

/* Check to see if there are any file errors.        */
/* This will either create a new file OUTVAR.DAT, or */
/* write over an existing version of the file.       */
  if ( (fp = fopen("outvar.dat", "w")) == NULL)
    {
      printf("Cant open OUTVAR.DAT \n");
      exit(1);
    }
```

```
/* This loop is used to save y=cos(x).*/
  for(x=0; x<=10; x += 0.1)
    {
      y = cos(x);
      z = sin(x);
      fprintf(fp, "%f %f %f",x,y,z);   /* save the x and y values */
      fprintf(fp, "\n");               /* create a line break     */
    }
  fclose(fp);                          /* close the file */
}
```

Comment:
For Other High Level Languages: If you want to use Fortran, Pascal, Basic, or some other language and MATLAB to generate plots you proceed in a similar manner to how you do for C. Note that any space delimited ASCII data file can be read by MATLAB (i.e. columns of data separated by any number of "blank" spaces). Hence, if you use any high level language simply output your data into an ASCII file and use the MATLAB commands given above.

# Appendix 2

# Appendix 3

# Appendix 4