Matlab Tutorial
(material available at
http://www.cs.clemson.edu/~jmarty/courses/matlabTutorial)

Dr. Jim Martin
Associate Professor
School of Computing
Clemson University
jim.martin@cs.clemson.edu
http://www.cs.clemson.edu/~jmarty

Networking Lab’s Website:  http://www.cs.clemson.edu/~jmarty/netlab/
Matlab Tutorial: Overview

• Objective is to give you some hands on experience with using Matlab for data analysis
Matlab Overview

What Is MATLAB?

• MATLAB (MATrix LABoratory)
• high-performance language for technical computing
• computation, visualization, and programming in an easy-to-use environment

Typical uses include:

• Math and computation
• Algorithm development
• Modelling, simulation, and prototyping
• Data analysis, exploration, and visualization
• Scientific and engineering graphics
• Application development, including Graphical User Interface building
What is Matlab?

• Matlab is basically a high level language which has many specialized toolboxes for making things easier for us

• How high?

```
  Matlab
  └── High Level Languages such as C, Pascal etc.
    └── Assembly
```
Matlab Intro

A good choice for vision program development because:

- Easy to do very rapid prototyping
- Quick to learn, and good documentation
- A good library of image processing functions
- Excellent display capabilities
- Widely used for teaching and research in universities and industry
- Another language to impress your boss with!
Matlab Intro

Has some drawbacks:

- Slow for some kinds of processes
- Not geared to the web
- Not designed for large-scale system development
MATLAB consists of:

- **The MATLAB language**
  - a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

- **The MATLAB working environment**
  - the set of tools and facilities that you work with as the MATLAB user or programmer, including tools for developing, managing, debugging, and profiling

- **Handle Graphics**
  - the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics.

- ...(cont’d)
Matlab Intro

- ... 

- **The MATLAB function library.**
  - a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms as well as special image processing related functions

- **The MATLAB Application Program Interface (API)**
  - a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.
Matlab Intro

Some facts for a first impression

• Everything in MATLAB is a matrix!

• MATLAB is an interpreted language, no compilation needed (but possible)

• MATLAB does not need any variable declarations, no dimension statements, has no packaging, no storage allocation, no pointers

• Programs can be run step by step, with full access to all variables, functions etc.
Matlab Screen

- **Command Window**
  - type commands

- **Current Directory**
  - View folders and m-files

- **Workspace**
  - View program variables
  - Double click on a variable to see it in the Array Editor

- **Command History**
  - view past commands
  - save a whole session using diary
Variables

• No need for types. i.e.,

```c
int a;
double b;
float c;
```

• All variables are created with double precision unless specified and they are matrices.

Example:
```c
>>x=5;
>>x1=2;
```

• After these statements, the variables are 1x1 matrices with double precision
Array, Matrix

- **a vector** \( x = [1 \ 2 \ 5 \ 1] \)
  
  \[
  x = \\
  \begin{bmatrix}
  1 & 2 & 5 & 1 \\
  \end{bmatrix}
  \]

- **a matrix** \( x = [1 \ 2 \ 3; \ 5 \ 1 \ 4; \ 3 \ 2 \ -1] \)
  
  \[
  x = \\
  \begin{bmatrix}
  1 & 2 & 3 \\
  5 & 1 & 4 \\
  3 & 2 & -1 \\
  \end{bmatrix}
  \]
Long Array, Matrix

• \( t = 1:10 \)

\[
t = \\
1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10
\]

• \( k = 2:-0.5:-1 \)

\[
k = \\
2 \quad 1.5 \quad 1 \quad 0.5 \quad 0 \quad -0.5 \quad -1
\]

• \( B = [1:4; \ 5:8] \)

\[
x = \\
1 \quad 2 \quad 3 \quad 4 \\
5 \quad 6 \quad 7 \quad 8
\]
Generating Vectors from functions

- **zeros(M,N)**  MxN matrix of zeros
  
  \[ x = \text{zeros}(1,3) \]
  
  \[ x = \\
  0 \hspace{1cm} 0 \hspace{1cm} 0 \]

- **ones(M,N)**  MxN matrix of ones
  
  \[ x = \text{ones}(1,3) \]
  
  \[ x = \\
  1 \hspace{1cm} 1 \hspace{1cm} 1 \]

- **rand(M,N)**  MxN matrix of uniformly distributed random numbers on (0,1)
  
  \[ x = \text{rand}(1,3) \]
  
  \[ x = \\
  0.9501 \hspace{1cm} 0.2311 \hspace{1cm} 0.6068 \]
Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be positive integer

Given:

A =

\[
\begin{bmatrix}
3 & 5 & 3 \\
6 & 8 & 2 \\
2 & 7 & 3 \\
\end{bmatrix}
\]

- \( A(-2), A(0) \)
  
  Error: ??? Subscript indices must either be real positive integers or logicals.

- \( A(4,2) \)
  
  Error: ??? Index exceeds matrix dimensions.
Concatenation of Matrices

- $x = [1 \ 2]$, $y = [4 \ 5]$, $z = [0 \ 0]$

\[
A = \begin{bmatrix} x & y \end{bmatrix}
\]

\[
1 \ \ 2 \ \ 4 \ \ 5
\]

\[
B = \begin{bmatrix} x ; y \end{bmatrix}
\]

\[
1 \ \ 2
4 \ \ 5
\]

\[
C = \begin{bmatrix} x \ y ; z \end{bmatrix}
\]

Error:

??? Error using ==> vertcat CAT arguments dimensions are not consistent.
Operators (arithmetic)

+ addition
- subtraction
* multiplication
/ division
^ power
` complex conjugate transpose
Exercise

• Create a 4x4 matrix that contains random numbers
• Look up help for the random number generator (help rand)
Exercise

• Create a 4x4 matrix that contains random numbers
• Look up help for the random number generator (help rand)

• \( x=[\text{rand}(1,4);\text{rand}(1,4);\text{rand}(1,4);\text{rand}(1,4)] \)
Example Plot

- $X = \text{rand}(1,100)$; creates 100 RNs
- $Y = (2^x) + 2$
- Plot($x, y$)
Random Numbers

How can you tell if a random number generator is good?

Properties of a U(0,1) RN Generator:
- Uniformly distributed in the interval (0,1)
- The RNs should be independent...i.e., uncorrelated
- Many RNs should be generated before the cycle repeats (ideally it exhibits full period)
- Reproducible and allow multiple streams
- Consumes minimal cpu and memory resources

• Technique: plot sequential random numbers from the generator....
  - X=rand(1,1000);
  - Y=rand(1,1000);
  - Plot(x,y) ??

  - Try a scatter plot instead...(help scatter)
Random Numbers

Technique: plot sequential random numbers from the generator….

- X=rand(1,1000);
- Y=rand(1,1000);
- Plot(x,y) ??

- Try a scatter plot instead…(help scatter)
• A time series is a sequence of observations taken sequentially in time.
  • Typically the observations are dependent
• Time series analysis is concerned with techniques for the analysis of this dependence.
• A time series $Z_1 \ldots Z_n$ of $n$ successive observations can be regarded as a sample realization from an infinite population of such time series that could have been generated by a stochastic process.
• A model that describes the probability structure of a sequence of observations is called a stochastic process.
• A special class of stochastic processes called stationary processes assumes the process is in a particular state of statistical equilibrium.
  • i.e., the statistical properties are unaffected by time.
Sample Mean in Matlab

For observations $x_1, x_2 \ldots, x_n$ of a time series, the sample mean is

$$\bar{x} = \frac{1}{n} \sum_{t=1}^{n} x_t$$

$$>> \text{load dataset1.dat}$$
$$>> \text{mySize=size(dataset1)}$$

mySize =

$$478 \quad 2$$

If we issue ‘mean dataset1’, should see

ans =

$$98.8750$$
• Look at a Matlab program that reads a data file (call it ping.dat, contains ping RTT samples in milliseconds):
  • 50.0
  • 366
  • 51.3
  • 51.9
• First, let’s compute the sample mean without using the mean function. Second, let’s plot the PDF
function mySampleMean()

load dataset1.dat
sampleSize = size(dataset1);
numberSamples = sampleSize(1);
totalThroughputCount = 0;
averageThroughput = 0.0;
for i = 1 : numberSamples
    totalThroughputCount = totalThroughputCount + dataset1(i,2);
end
averageThroughput = totalThroughputCount / numberSamples;

fprintf(1,'mySampleMean: NumberSamples: %d, sample Throughput: %12.0f \n',numberSamples,averageThroughput);

return;
• Using the same dataset, input the following to plot the PDF of the timeseries:

```matlab
load dataset1.dat
myData = dataset1(:,2);
myDataSize = size(dataset1)
[N,X] = hist(myData,100);
N = N/myDataSize(1);
bar(X,N);
axis([0 max(myData) 0 0.25]);
hold on
ylabel('PDF')
xlabel('Distribution Values')
title('distribution PDF')
grid
```
Plotting the Distribution (pdf)

• See plotPDF.m

load ping.dat
myData=ping(:,2);
myDataSize=size(dataset1)
[N,X] = hist(myData,100);
N=N/myDataSize(1);
bar(X,N);

hold on
ylabel('probability density')
xlabel(Values')
title('PDF')
grid