MATLAB Tutorial

MATLAB Basics
&
Signal Processing Toolbox
Part 1: Introduction

- Toolboxes & Simulink
- Commands & functions
- Help system
- Variables & operators
- Graphics
- Symbolic Math Toolbox

Part 2: Signal Processing Toolbox

- Representing Signals
- Basic Waveform Generation
- Convolution
- Impulse Response
- Frequency Response
- Discrete Fourier Transform
- Filters
MATLAB Tutorial
Part 1

MATLAB Basics
What is MATLAB?

Matlab = Matrix Laboratory

A software environment for interactive numerical computations

*Examples:*
  - Matrix computations and linear algebra
  - Solving nonlinear equations
  - Numerical solution of differential equations
  - Mathematical optimization
  - Statistics and data analysis
  - Signal processing
  - Modelling of dynamical systems
  - Solving partial differential equations
  - Simulation of engineering systems
MATLAB Toolboxes

MATLAB has a number of add-on software modules, called toolboxes, that perform more specialized computations.

Signal & Image Processing
- Signal Processing
- Image Processing
- Communications
- System Identification
- Wavelet
- Filter Design

Control Design
- Control System
- Fuzzy Logic
- Robust Control
- μ-Analysis and Synthesis
- LMI Control
- Model Predictive Control

More than 60 toolboxes!
Simulink - a package for modeling dynamic systems
Simulink (cont’d)

Analyzing results:

TX Data

Unequalized signal

RX power spectrum (dB)

SNR (dB)

Bit rate (Mb/s)

Equalized signal

Equalized power spectrum (dB)

BER (per packet)
MATLAB Workspace

The MATLAB environment is command oriented

To get started, select MATLAB Help or Demos from the Help menu.

>>
Some Useful MATLAB commands

- `what` List all m-files in current directory
- `dir` List all files in current directory
- `ls` Same as `dir`
- `type test` Display `test.m` in command window
- `delete test` Delete `test.m`
- `cd a:` Change directory to `a:`
- `chdir a:` Same as `cd`
- `pwd` Show current directory
- `which test` Display current directory path to `test.m`
Construction

• Core functionality: compiled C-routines

• Most functionality is given as m-files, grouped into toolboxes
  – m-files contain source code, can be copied and altered
  – m-files are platform independent (PC, Unix/Linux, MAC)

• Simulation of dynamical systems is performed in Simulink
Math

- MATLAB can do simple math just as a calculator.

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>SYMBOL</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition, $a + b$</td>
<td>+</td>
<td>3 + 22</td>
</tr>
<tr>
<td>Subtraction, $a - b$</td>
<td>-</td>
<td>90 - 44</td>
</tr>
<tr>
<td>Multiplication, $a \times b$</td>
<td>*</td>
<td>3.14*4.20</td>
</tr>
<tr>
<td>Division, $a \div b$</td>
<td>/ or \</td>
<td>56/8 = 8\56</td>
</tr>
<tr>
<td>Exponentiation, $a^b$</td>
<td>^</td>
<td>2^16</td>
</tr>
</tbody>
</table>
Interactive Calculations

Matlab is interactive, no need to declare variables

>> 2+3*4/2
>> a=5e-3; b=1; a+b

Most elementary functions and constants are already defined

>> cos(pi)
>> abs(1+i)
>> sin(pi)
MATLAB has many built-in functions. Some math functions are:

acos, acosh, acot, acsc, acsch, asec, asech, asin, asinh, atan, atan2, atanh, cos, cosh, cot, coth, csc, csch, sec, sech, sin, sinh, tan, tanh, exp, log, log10, log2, pow2, sqrt, nextpow2, abs, angle, conj, imag, real, unwrap, isreal, cplxpair, fix, floor, ceil, round, mod, rem, sign, cart2sph, cart2pol, pol2cart, sph2cart, factor, isprime, primes, gcd, lcm, rat, rats, perms, nchoosek, airy, besselj, bessely, besselh, besseli, besselk, beta, betainc, betaln, ellipj, ellipke, erf, erfc, erfcx, erfinv, expint, gamma, gammainc, gammaln, legendre, cross, dot
The Help System

Search for appropriate function
>> lookfor keyword

Rapid help with syntax and function definition
>> help function

An advanced hyperlinked help system is launched by
>> helpdesk

Complete manuals (html & pdf)
The help system example

Step 1: >> lookfor convolution
CONV Convolution and polynomial multiplication.
CONV2 Two dimensional convolution.
CONVN N-dimensional convolution.
DECONV Deconvolution and polynomial division.
CONVENC Convolutionally encode binary data.
DISTSPEC Compute the distance spectrum of a convolutional code.
...

Step 2: >> help conv
CONV Convolution and polynomial multiplication.
    C = CONV(A, B) convolves vectors A and B. The resulting
    vector is length LENGTH(A)+LENGTH(B)-1.
    If A and B are vectors of polynomial coefficients, convolving
    them is equivalent to multiplying the two polynomials.

    Class support for inputs A,B:
        float: double, single

    See also deconv, conv2, convn, filter and, in the Signal
    Processing Toolbox, xcorr, convmtx.

    ...

MATLAB
MATLAB Variable Names

- Variable names ARE case sensitive
- Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer)
- Variable names must start with a letter followed by letters, digits, and underscores.

All variables are shown with

```
>> who
>> whos
```

Variables can be stored on file

```
>> save filename
>> clear
>> load filename
```
MATLAB Special Variables

ans      Default variable name for results
pi       Value of π
inf      ∞
NaN      Not a number e.g. 0/0
i and j  i = j = \sqrt{-1}
eps      Smallest incremental number
realmin  The smallest usable positive real number
realmax  The largest usable positive real number
MATLAB Math & Assignment Operators

Power
^ or .^ a^b or a.^b

Multiplication
* or .* a*b or a.*b

Division
/ or ./ a/b or a./b
or \ or .\ b\a or b.\a

NOTE:
56/8 = 8\56

− (unary) + (unary)

Addition
+ a + b

Subtraction
− a − b

Assignment
= a = b (assign b to a)
MATLAB Matrices

• MATLAB treats all variables as matrices. For our purposes a matrix can be thought of as an array, in fact, that is how it is stored.

• Vectors are special forms of matrices and contain only one row OR one column.

• Scalars are matrices with only one row AND one column
Vectors and Matrices

Vectors (arrays) are defined as

\[
\begin{align*}
>> v &= [1, 2, 4, 5] \\
>> w &= [1; 2; 4; 5]
\end{align*}
\]

Matrices (2D arrays) defined similarly

\[
\begin{align*}
>> A &= [1,2,3;4,-5,6;5,-6,7]
\end{align*}
\]
Find polynomial roots:

\[ 1.2x^3 + 0.5x^2 + 4x + 10 = 0 \]

>> x=[1.2,0.5,4,10]

x =

1.200   0.500   4.00   10.00

>> roots(x)

ans =

0.59014943179299 + 2.20679713205154i
0.59014943179299 - 2.20679713205154i
-1.59696553025265
Visualization of vector data is available

```matlab
>> x=-pi:0.1:pi; y=sin(x);
>> plot(x,y)
>> xlabel('x'); ylabel('y=sin(x)');
```

```
plot(x,y)
```

```
stem(x,y)
```
Matlab Selection Structures

An if-elseif-else structure in MATLAB.

```matlab
if expression1 % is true
    % execute these commands
elseif expression2 % is true
    % execute these commands
else % the default
    % execute these commands
end
```

MATLAB Repetition Structures

A for loop in MATLAB:
```
for x = array
    for x = 1: 0.5 : 10
        % execute these commands
    end
end
```

A while loop in MATLAB:
```
while x <= 10
    % execute these commands
end
```
"for" loop example

```matlab
>> x = -1:.05:1;
>> for n = 1:8
    subplot(4,2,n);
    plot(x,sin(n*pi*x));
end
```
m-file example

**Task:** The area, \( A \), of a triangle with sides of length \( a \), \( b \) and \( c \) is given by

\[
A = \sqrt{s(s - a)(s - b)(s - c)},
\]

where \( s = (a + b + c)/2 \).

**File area.m:**

```matlab
function [A] = area(a,b,c)
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
```

**Usage example:** To evaluate the area of a triangle with side of length 10, 15, 20:

```matlab
>> Area = area(10,15,20)
Area =
  72.6184
```
Integration example

Find the integral: \[ \int_{0}^{10} \left( \frac{1}{2} \sqrt{x} + x \sin(x) \right) dx \]

example with \texttt{trapz} function:

\[
\begin{align*}
\texttt{x} &= 0:0.5:10; \quad \texttt{y} = 0.5 * \texttt{sqrt(x)} + \texttt{x} .* \texttt{sin(x)}; \\
\texttt{integral1} &= \texttt{trapz(x,y)} \\
\texttt{integral1} &= 18.1655
\end{align*}
\]
Symbolic Math Toolbox

The Symbolic Math Toolbox uses "symbolic objects" produced by the "sym" function.

\[ \int x^3 \, dx \, \quad \text{and} \quad \frac{d}{dx} \left( x^3 \right) \]

Example:

\[ \frac{d}{dx} \left( x^3 \right) \quad \text{and} \quad \int x^3 \, dx \]

>> x = sym('x'); % produces a symbolic variable named x
>> f = x^3; % defines a function

\[
\begin{align*}
\text{>> x} & = \text{sym('x');} \\
\text{>> diff(x^3)} & \text{ ans =} \\
& 3*x^2 \\
\text{>> int(x^3)} & \text{ ans =} \\
& 1/4*x^4
\end{align*}
\]
Once a symbolic variable is defined, you can use it to build functions. EZPLOT makes it easy to plot symbolic expressions.

\[
\text{>> } x = \text{sym('x');}
\text{>> } f = 1/(5+4\cos(x))
\text{>> } \text{ezplot(f)}
\]
Symbolic Math Toolbox

Plot the following functions:

\[
\text{Gaussian}
\]

\[
>> x = \text{sym('x')};
\]

\[
\text{Gaussian}
\]

\[
>> \text{ezplot(exp(-pi*x^2))}
\]

\[
sinc(x) = si(\pi x) = \sin(\pi x)/(\pi x)
\]

\[
>> \text{ezplot(sinc(x))}
\]
MATLAB Tutorial
Part 2
Signal Processing Toolbox
What Is the Signal Processing Toolbox?

The Signal Processing Toolbox is a collection of tools or functions expressed mostly in M-files, that implement a variety of signal processing tasks.

Command line functions for:

- Analog and digital filter analysis
- Digital filter implementation
- FIR and IIR digital filter design
- Analog filter design
- Statistical signal processing and spectral analysis
- Waveform generation

Interactive tools (GUIs) for:

- Filter design and analysis
- Window design and analysis
- Signal plotting and analysis
- Spectral analysis
- Filtering signals
Representing signals

MATLAB represents signals as *vectors*:

```
>> x=[1,2,3,5,3,2,1]
```

```
x =
```

```
    1   2   3   5   3   2   1
```

```
>> stem(x)
```
Waveform Generation

Consider generating data with a 1000 Hz sample frequency.

An appropriate time vector:

```matlab
>> t = 0:0.001:1;  % a 1001-element row vector that represents
                  % time running from zero to one second
                  % in steps of one millisecond.
```

A sample signal $y$ consisting of two sinusoids, one at 50Hz and one at 120 Hz with twice the amplitude:

```matlab
>> y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
>> plot(t(1:50),y(1:50));
```
Waveform Generation

Basic Signals:

Unit impulse:
>> t = 0:0.01:1;
>> y = [zeros(1,50),1,zeros(1,50)];
>> plot(t,y);

Unit step:
>> y = [zeros(1,50),ones(1,51)];
>> plot(t,y);

Rectangle:
>> t=-1:0.001:1;
>> y=rectpuls(t);
>> plot (t,y);

Triangle:
>> t=-1:0.001:1;
>> y=tripuls(t);
>> plot (t,y);
Waveform Generation

Common Sequences:

Sawtooth:
>>> fs = 10000;
>>> t = 0:1/fs:1.5;
>>> x = sawtooth(2*pi*50*t);
>>> plot(t,x), axis([0 0.2 -1 1]);

Square wave:
>>> t=0:20;
>>> y=square(t);
>>> plot(t,y)

Sinc function:
>>> t = -5:0.1:5;
>>> y = sinc(t);
>>> plot(t,y)
Convolution

A digital filter’s output $y(k)$ is related to its input $x(k)$ by convolution with its impulse response $h(k)$.

$$y(k) = h(k) * x(k) = \sum_{l=-\infty}^{\infty} h(k-l)x(l)$$

```matlab
>> t1=-1:0.001:1;
>> tri=tripuls(t1,2);
>> plot(t1,tri);

>> c=conv(tri,tri);
>> t2=-2:0.001:2;
>> plot(t2,c);
```
Convolution (Example)

Let the rectangular pulse \( x(n) = \gamma(0.1n-5) \) be an input to an LTI system with impulse response \( h(n)=0.9^n \gamma(n) \). Determine the output \( y(n) \).

\[
\text{>> } x=\text{rectpuls}(n,10); \\
\text{>> } x=\text{circshift}(x,[0 \ 5]); \\
\text{>> } \text{stem}(n,x)
\]

\[
\text{>> } \text{step}=[\text{zeros}(1,5),\text{ones}(1,51)]; \\
\text{>> } h=0.9.^n.*\text{step}; \\
\text{>> } \text{stem}(n,h)
\]

\[
\text{>> } y=\text{conv}(h,x); \\
\text{>> } \text{stem}(y)
\]
Filters

Z-transform $Y(z)$ of a digital filter’s output $y(n)$ is related to the $z$-transform $X(z)$ of the input by:

$$Y(z) = H(z)X(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(n + 1)z^{-n}}{a(1) + a(2)z^{-1} + \cdots + a(m + 1)z^{-m}} X(z)$$

The system can also be specified by a linear difference equation:

$$y(k) = b_1x(k) + b_2x(k - 1) + \cdots + b_{n+1}x(k - n) - a_2y(k - 1) - \cdots - a_{m+1}y(k - n)$$

MATLAB function `filter` - filter data with a recursive (IIR) or nonrecursive (FIR) filter
Filter (Example 1)

Given the following difference equation of a filter:

\[ y(n) - y(n-1) + 0.9y(n-2) = x(n) \]

Calculate and plot the impulse response \( h(n) \) and unit step response \( s(n) \) at \( n = -20,\ldots,100 \).

```matlab
>> a=[1,-1,0.9]; b=[1];
>> n=[-20:120];

>> x=zeros(1,20),1,zeros(1,120)];
>> h=filter(b,a,x);
>> stem(n,h); title('impulse response');

>> x=zeros(1,20),ones(1,121]);
>> s=filter(b,a,x);
>> stem(n,s); title('step response');
```
Filter (Example 2)

Create a 10-point averaging lowpass FIR filter:

$$y[n] = \frac{1}{10} x[n] + \frac{1}{10} x[n-1] + \ldots + \frac{1}{10} x[n-9]$$

As an input consider a 1-second duration signal sampled at 100 Hz, composed of two sinusoidal components at 3 Hz and 40 Hz.

```matlab
>> fs = 100;
>> t = 0:1/fs:1;
>> x = sin(2*pi*t*3) + .25*sin(2*pi*t*40);
>> b = ones(1,10)/10; % 10 point averaging filter
>> y = filter(b,1,x);
>> plot(t,x,'b',t,y,'r')
```
Discrete-Time Fourier Series

DTFS is a frequency-domain representation for periodic discrete-time sequences.

For a signal \( x[n] \) with fundamental period \( N \), the DTFS equations are given by:

\[
x[n] = \sum_{k=0}^{N-1} a_k e^{jk(2\pi/N)n}
\]

\[
a_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n]e^{-jk(2\pi/N)n}
\]

fft – is an efficient implementation in MATLAB to calculate \( a_k \).
Discrete-Time Fourier Series (Example)

Find DTFS for periodic discrete-time signal $x[n]$ with period $N=30$

```matlab
>> x=[1,1,zeros(1,28)];
>> N=30; n=0:N-1;
>> a=(1/N)*fft(x);

>> real_part=real(a);
>> stem(n,real_part);
>> xlabel('k'); ylabel('real(a)');

>> imag_part=imag(a);
>> stem(n,imag_part);
>> xlabel('k'); ylabel('imag(a)');
```
Find the frequency response of a 10-point averaging lowpass FIR filter and plot its magnitude and phase

\[ y[n] = \frac{1}{10} x[n] + \frac{1}{10} x[n-1] + \cdots + \frac{1}{10} x[n - 9] \]

```matlab
>> b = ones(1,10)/10; a=1;
>> [H omega]=freqz(b,a,100,'whole');
>> magH=abs(H);
>> plot(omega, magH); grid;

>> angH=angle(H);
>> plot(omega, angH/pi); grid;
```
Example

Find the spectrum of the following signal:

\[ f = 0.25 + 2\sin(2\pi 5k) + \sin(2\pi 12.5k) + 1.5\sin(2\pi 20k) + 0.5\sin(2\pi 35k) \]

```matlab
>> N=256; % number of samples
>> T=1/128; % sampling frequency=128Hz
>> k=0:N-1; time=k*T;
>> f=0.25+2*sin(2*pi*5*k*T)+1*sin(2*pi*12.5*k*T)+...  
     +1.5*sin(2*pi*20*k*T)+0.5*sin(2*pi*35*k*T);
>> plot(time,f); title('Signal sampled at 128Hz');
>> F=fft(f);
>> magF=abs([F(1)/N,F(2:N/2)/(N/2)]);
>> hertz=k(1:N/2)*(1/(N*T));
>> stem(hertz,magF), title('Frequency components');
```
Example

Find the frequency components of a signal buried in noise. Consider data sampled at 1000 Hz. Form a signal consisting of 50 Hz and 120 Hz sinusoids and corrupt the signal with random noise.

```matlab
>> t = 0:0.001:0.6;
>> x = sin(2*pi*50*t) + sin(2*pi*120*t);
>> y = x + 2*randn(1,length(t));
>> plot(y(1:50));
```
Example (cont’d)

It is difficult to identify the frequency components by studying the original signal.

The discrete Fourier transform of the noisy signal using a 512-point fast Fourier transform (FFT):

\[ Y = \text{fft}(y,512); \]

The power spectral density, a measurement of the energy at various frequencies, is

\[ Pyy = Y.*\text{conj}(Y) / 512; \]

\[ f = 1000*(0:255)/512; \]
\[ \text{plot}(f,Pyy(1:256)) \]
Links

One-hour recorded online Webinars
http://www.mathworks.com/company/events/archived_webinars.html

All matlab manuals

Matlab Tutorials
http://www.math.ufl.edu/help/matlab-tutorial/
http://www.math.unh.edu/~mathadm/tutorial/software/matlab/