MATLAB Session for CS4514

Adriana Hera

ahera @wpi.edu

Computing & Communications Center

- November 28, 2006-

Part of the notes are from Matlab documentation

MATLAB Session for CS4514

1. **Matlab Basics**
   - Starting Matlab
   - Matlab Help
   - Matlab Variables
   - Numbers
   - Operators
   - Functions
   - Matrices
   - Writing a script
   - Plotting Data
   - Importing Data
2. **Processing experimental data**
   - Importing Data in Matlab
   - Processing Data
   - Plotting Data

slides 32-end

codes: code1.m
code2.m

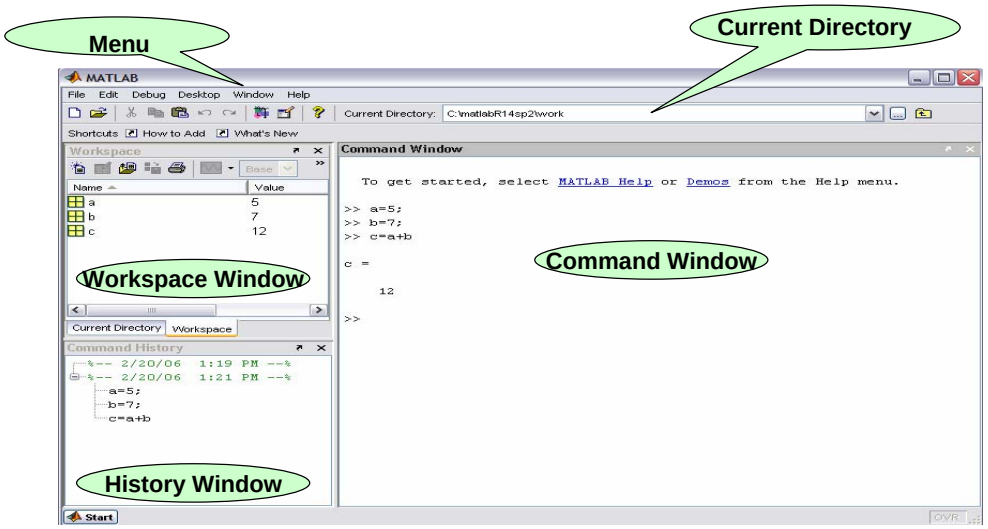
A. Files: typeperf1.csv, typeperf2.csv, etc
B. Files: logFile1.dat, logFile2.dat, etc

What is Matlab ?

- **MATLAB® is a high-performance language for technical computing.**
- It integrates **computation**, **visualization**, and **programming** in an *easy-to-use environment* where problems and solutions are expressed in familiar mathematical notation.
- **MATLAB** stands for **matrix laboratory.**
- MATLAB is an interactive system whose **basic data element** is an **matrix (array)** that does not require dimensioning.
- This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

Starting Matlab

- ***Windows:*** Start menu → Matlab → Matlab
- ***Unix:*** Terminal window→ type **`matlab`**



Matlab Help

1. **Using HELP menu → MATLAB Help**

   HELP → Using Help Browser

2. >> **helpdesk** **Opens the Help browser.**

3. **>> help** `commandname/toolboxname/functionname`

   Ex: >> help `sin`

4. **>> doc** `commandname/toolboxname/functionname`

   displays the detailed info in the Help browser.

   Ex: >> doc `sin`

Other commands:

5. >> **lookfor = helpdesk -> search**

I. Matlab Programming

- **Matlab Variables**
- **Operators**
- **Functions**
- **.....**

Matlab Variables

- A MATLAB variable is essentially a tag that you assign to a value in memory.
- MATLAB does not require any type declarations or dimension statements.
- When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage.
- If the variable already exists, MATLAB changes its contents.
- Variable names consist of a letter, followed by any number of letters, digits, or underscores.
- MATLAB uses only the first 31 characters of a variable name.
- MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters.
- MATLAB stores variables in a part of memory called workspace.
- To view what is stored in a variable type its name.

Types of Variables: MATLAB provides three basic types of variables:

Local Variables
Global Variables
Persistent Variables

Matlab Variables

Rules for variable names:

- Make Sure Variable Names Are Valid
- Don't Use Function Names for Variables
- Check for Reserved Keywords
- Avoid Using i and j for Variables

Syntax:

```
varialeName=Value;
```

Example:

```
>> a=5;
>> b=7;
>> c=a+b
>> method='linear'
```

How to remove a variable from workspace:

```
>> clear variableName
>> clear
```

- removes all variables from the workspace (!!!!)

ans = default variable, when the result is not assign to a variable

Exercise: 1. Define a1=8 and b2=8, c1=a1+b2
2. Other commands: variable= input('prompt') (>>help input)

```
>> a3=input('a3=')
```

Operators

Expressions use familiar **arithmetic operators** and precedence rules.

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division (described in "Matrices and Linear Algebra" in the
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

Matlab Functions

1. **Standard elementary mathematical functions**

```
>> help elfun
```

Trigonometric (`sin, cos`)

Exponential (`exp, log`)

Complex (`abs, angle`)

Rounding and remainder (`round`)

2. **Elementary matrices and matrix manipulation.**

```
>> help elmat
```

3. **Specialized math functions.**

```
>>  help specfun
```

pi	3.14159265...
i	Imaginary unit, $\sqrt{-1}$
j	Same as i
eps	Floating-point relative precision, $\varepsilon = 2^{-52}$
Inf	Infinity
NaN	Not-a-number

Statistics

`mean, std, min, max`

Given: a vector **`x`**

Syntax:

```
M = mean(x)          - average or mean value
S = std(x)           - returns the standard deviation
min_x = min(x)             - the smallest element in x
max_x=max(x)         - returns the largest element in x
```

Example: open the file myStatistics.m

myStatistics.m

myStatistics1.m

Advanced:

Statistics for multidimensional data sets
Statistics toolbox

- **Matrices**
- **Operators**
- **Functions**

.....

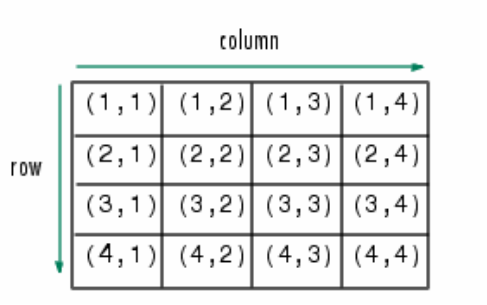
Matrix & basic matrix functions

Define a matrix:

1. Type the matrix
2. Use Specialized Matrix Functions

Matrix Manipulation

Matrix Functions



Matrix: Define a matrix

1. Type the matrix

- Separate the elements of a row with **blanks** or **commas**.
- Use a **semicolon**, **;** , to indicate the end of each row.
- Surround the entire list of elements with **square brackets**, [ ].

$$S = \begin{bmatrix} 3 & -10 & 0 \\ -10 & 0 & 30 \\ 0 & 30 & -27 \end{bmatrix}$$

```
1. >>  S=[3 -10 0; -10 0 30; 0 30 -27]
```

Basic matrix information: **size** (size of a matrix)

```
>> [m,n] = size(X)
```

Matrix: Accessing Matrix Elements

- **individual element**

```
>> A(2,2)
```

```
ans =
    20
```

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 11 & 20 & 30 \end{bmatrix}$$

- **column**

```
>>  A(:,2)
```

```
ans =
     2
    20
```

: (colon) → all elements

- **row**

```
>> A(2,:)
```

```
ans =
    11    20    30
```

- **group of elements**

```
>> A(2,1:3)
```

```
ans =
    11    20    30
```

first element : **step:** last element

Matrix: Operations

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division (described in "Mat
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

A+B
A-B
A*B
A/B
A\B
A^B
A'

.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Unconjugated array transpose

A.*B
A./B
A.\B
A.^B
A.'

Matrix: Operations

$$\mathbf{A}=\begin{bmatrix}1 & 2 & 3\\ 2 & 3 & 1\\ 2 & 2 & 2\end{bmatrix},\qquad \mathbf{B}=\begin{bmatrix}10 & 20 & 30\\ 11 & 21 & 31\\ 1 & 2 & 3\end{bmatrix},$$

```
>> A=[1 2 3; 2 3 1; 2 2 2];
>> B= [10 20 30; 11 21 31; 1 2 3];
```

```
>> A*B
```

```
ans =
    35    68   101
    54   105   156
    44    86   128
```

$$\mathbf{A}=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix},\qquad \mathbf{B}=\begin{bmatrix}b_{11} & b_{12} & b_{13}\\ b_{21} & b_{22} & b_{23}\\ b_{31} & b_{32} & b_{33}\end{bmatrix},$$

$$\mathbf{A}.*\mathbf{B}=\begin{bmatrix}a_{11}\cdot b_{11} & a_{12}\cdot b_{12} & a_{13}\cdot b_{13}\\ a_{21}\cdot b_{21} & a_{22}\cdot b_{22} & a_{23}\cdot b_{23}\\ a_{31}\cdot b_{31} & a_{32}\cdot b_{32} & a_{33}\cdot b_{33}\end{bmatrix}$$

```
>> A.*B
```

```
ans =
    10    40    90
    22    63    31
     2     4     6
```


IV. Matlab Programming

- How to write a program (M-files)
  - Script
  - Function
- How to plot data

M-files

- **Files that contain code in the MATLAB** language are called *M-files*.
- You create M-files using a text editor.
- Use a M-file as any other MATLAB function or command.
- A M-file is a plain text file.

Two kinds of **M-files**:

Scripts

do not accept input arguments or *return output arguments*

operate on data in the workspace.

Functions

can accept input arguments and *return output arguments*

internal variables are local to the function.

```
>> edit  fileName
>> edit exSwitch
```


Matlab - Plotting

`plot`

Sintax:

`plot(y);` **`plot(x,y);`** **`plot(x,y,s)`**

The plot function has different forms, depending on the input arguments.

If y is a vector, `plot(y)` produces a piecewise linear graph of *the elements of y* versus the *index of the elements of y*.

If you specify two vectors as arguments, `plot(x,y)` produces a *graph of y versus x*.

Matlab - Plotting

`plot(x,y, s);`

`s` allows to plot : colors, symbols, different lines

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		….			

`plot (x,y,'c+:')` plots a cyan dotted line with a plus at each data point;

Matlab - Plotting

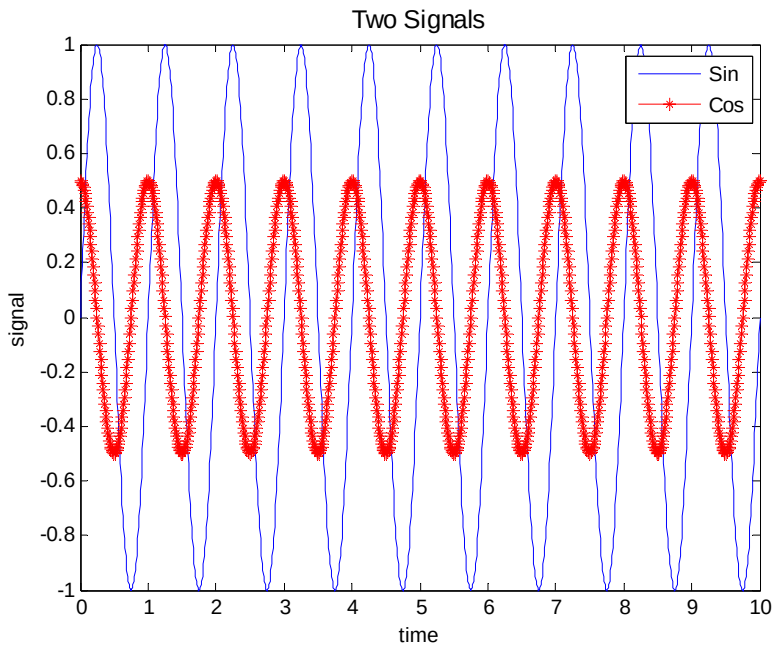
```
clear
t=0:0.01:10; % time seconds
signalSin=sin(2*pi*t); % signal1 - frequency =1 Hz
signalCos=0.5*cos(2*pi*t); % signal2 - frequency =1 Hz

figure
plot(t,signalSin);
```

plot2signals.m

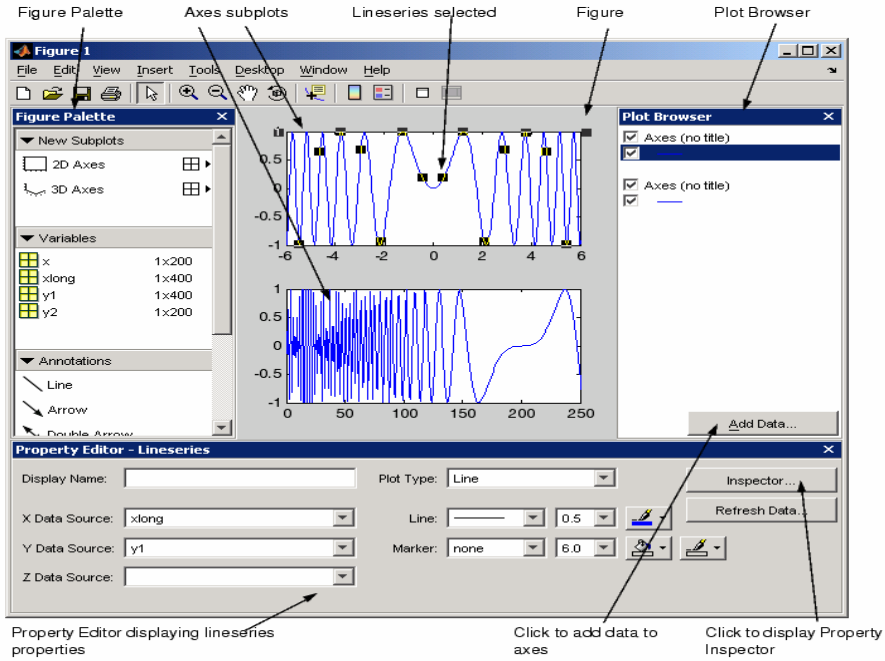
```
hold on
plot(t,signalCos, '-*r');

xlabel('time');  ylabel('signal');
legend('Sin', 'Cos');
title('Two Signals','FontSize',12)
```

Other commands: figure xlabel ylabel legend, title

Matlab - Plotting



Visualization - Interactive editing (optional)



Visualization (optional)

plot, plotyy, stem, subplot

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,PropertyName',PropertyValue,...)
```

plotyy (X1,Y1,X2,Y2) plots Y1 versus X1 with y-axis labeling on the left and plots Y2 versus X2 with y-axis labeling on the right.

stem (X,Y) plots the data sequence Y at the values specified in X.

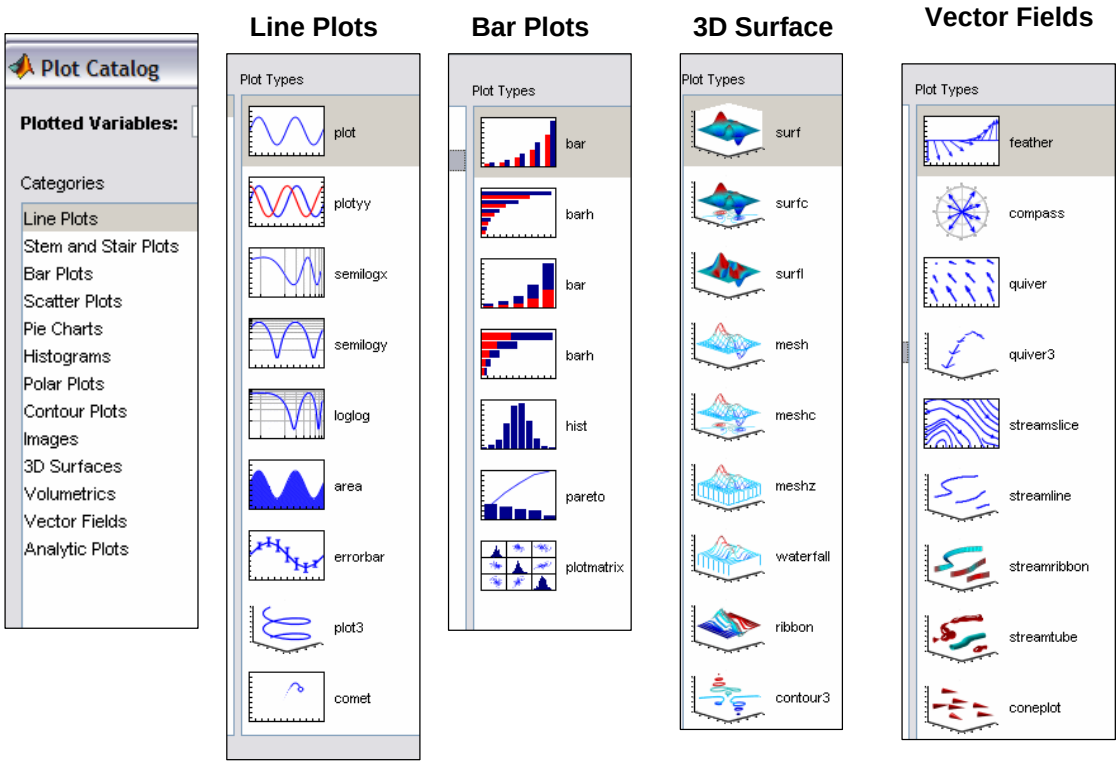
suplot (m,n,p), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for for the current plot

Visualization

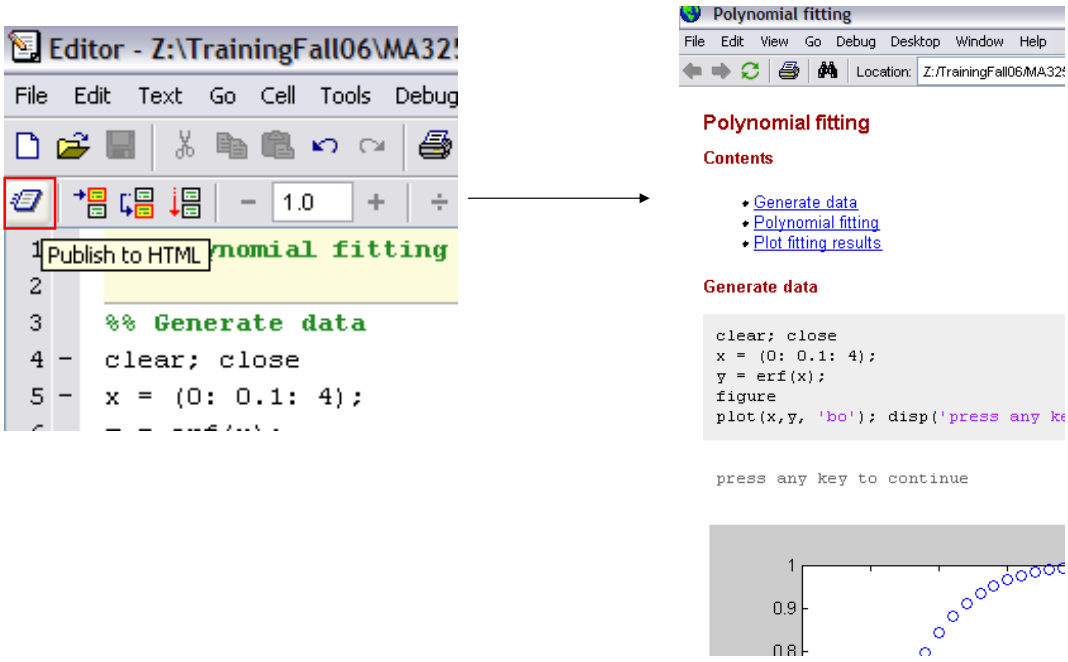
```
%%%%% plot commands %%%%%%%%%%%%%%%

t=1:1:20; x=sin(t/5);
y=0.01*x.*exp(-t);

figure
subplot(2,2,1)
plot(t,x,'--rs','LineWidth',2, MarkerEdgeColor','k',...
     'MarkerFaceColor','g','MarkerSize',6);
title('Plot 1');

subplot(2,2,2)
stem(t,x,'--rs','LineWidth',2);

title('Plot 2')
subplot(2,2,3);
bar(t,x); title('Plot 3')

subplot(2,2,4)
plotyy(t,x, t,y); title('plotyy')
```

see: testPlot.m

Visualization (optional)



Publishing a script to HTML



Importing and Exporting Data

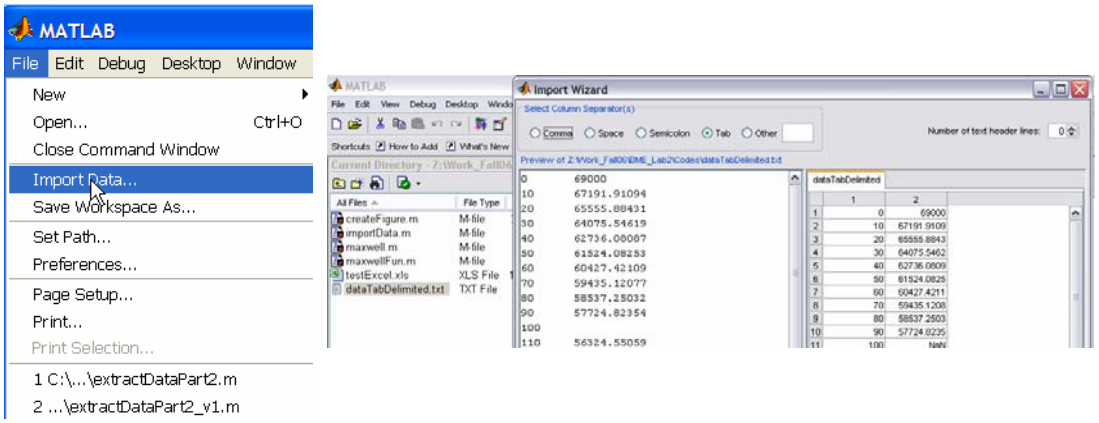
- **using the Import Wizard**
- **save , load**
- **dlmread , dlmwrite**
- **xlsread, xlswrite**
- **fopen, , fscanf, fprintf**
- **importdata**

Importing Data

1. Using the Import Wizard with Text Data

File → Import Data

or >> **uiimport**



2 Supported File Formats

Wizard: missing data: NaN (Not-a-Number.)

Importing and Exporting Data

2 Supported File Formats

File Format	File Content	Extension	Functions
MATLAB formatted	Saved MATLAB workspace	.mat	load, save
Text	Text	any	textscan
	Text	any	textread
	Delimited text	any	dlmread, dlmwrite
	Comma-separated numbers	.csv	csvread, csvwrite
Extended Markup Language	XML-formatted text	.xml	xmlread, xmlwrite
Audio	NeXT/SUN sound	.au	auread, auwrite
	Microsoft WAVE sound	.wav	wavread, wavwrite
Movie	Audio/video	.avi	aviread
Scientific data	Data in Common Data Format	.cdf	cdfread, cdfwrite
	Flexible Image Transport System data	.fits	fitsread
	Data in Hierarchical Data Format	.hdf	hdfread
Spreadsheet	Excel worksheet	.xls	xlsread, xlswrite
	Lotus 123 worksheet	.wk1	wk1read, wk1write
Graphics	TIFF image	.tiff	imread, imwrite
	PNG image	.png	same
	HDF image	.hdf	same
	BMP image	.bmp	same
	JPEG image	.jpeg	same

Processing Experimental Data for the Project

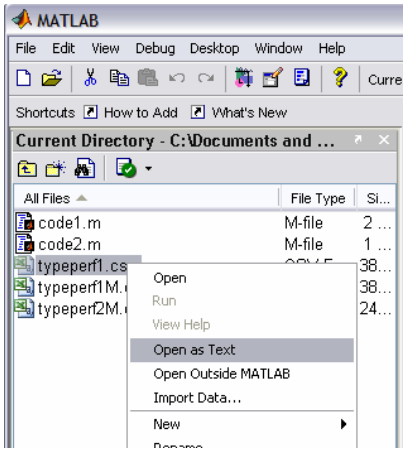
1. **Data Preparation**
2. **Import Data in Matlab**
3. **Process Data**
4. **Plot Data**

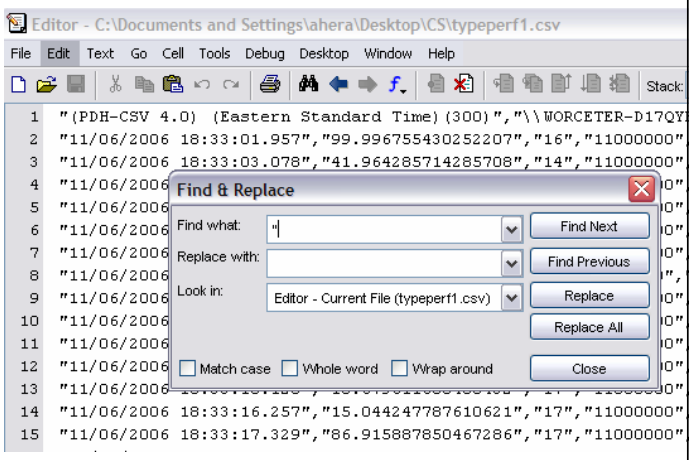
A. Files: typeperf1.csv, typeperf2.csv, etc ***code1.m***

B. Files: logFile1.dat, logFile2.dat, etc ***code2.m***

1. Data Preparation

1. Open as Text (typeperf.csv)
2. Editor window: Edit→ Find and Replace → *Remove double quotation marks*
3. File → Save as (*typeperfM.csv*)





2. Import Data in Matlab

```
A = importdata('typeperfM.csv');
% load data into a structure
```

→Structure A with two fields:

→A.data (numeric data)

→A.textdata (text data)

importdata

Load data from disk file

Syntax

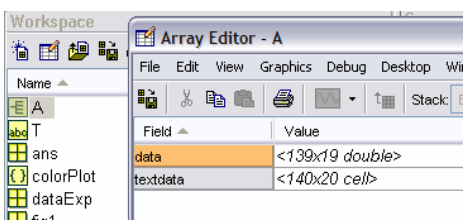
```
importdata('filename')
A = importdata('filename')
importdata('filename','delimiter')
```

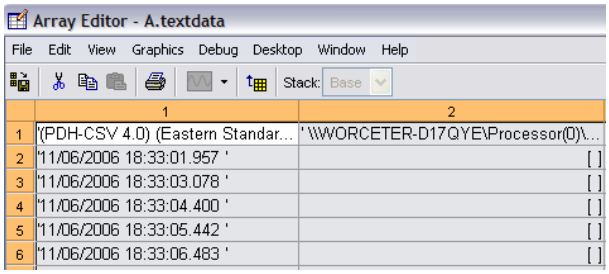
Description

`importdata('filename')` loads data from `filename`

`A = importdata('filename')` loads data from `file`:

`A = importdata('filename','delimiter')` loads





2. Import Data in Matlab

A = importdata('typeperfM.csv');

Col 1 Col 2 Col 3 Col 4

Editor - C:\Documents and Settings\ahera\Desktop\CS\typeperf1M.csv

File Edit Text Go Cell Tools Debug Desktop Window Help

```
1  (PDH-CSV 4.0) (Eastern Standard Time)(300) , \\WORCETER-D17QYE
2  11/06/2006 18:33:01.957 , 99.996755430252207 , 16 , 11000000 ,
3  11/06/2006 18:33:03.078 , 41.964285714285708 , 14 , 11000000 ,
4  11/06/2006 18:33:04.400 , 31.060606060606055 , 16 , 11000000 ,
5  11/06/2006 18:33:05.442 , 6.7307692307692291 , 16 , 11000000 ,
6  11/06/2006 18:33:06.483 , 23.076923076923073 , 15 , 11000000 ,
7  11/06/2006 18:33:08.025 , 57.792207792207797 , 16 , 11000000 ,
```

Array Editor - A.textdata

File Edit View Graphics Debug Desktop Window Help

Stack: Base

	1	2
1	'(PDH-CSV 4.0) (Eastern Standar...	'\\WORCETER...
2	'11/06/2006 18:33:01.957 '	[ ]
3	'11/06/2006 18:33:03.078 '	[ ]
4	'11/06/2006 18:33:04.400 '	[ ]
5	'11/06/2006 18:33:05.442 '	[ ]

Array Editor - A.data

File Edit View Graphics Debug Desktop Window Help

Stack: Base

	1	2	3	4
1	99.9968	16	11000000	1274.307
2	41.9643	14	11000000	5641.072
3	31.0606	16	11000000	6202.694
4	6.7308	16	11000000	683.561

→A.textdata (text data)

3. Process Data

datenum

Convert date and time to serial date number

Syntax

```
N = datenum(S)
N = datenum(S, P)
```

datestr

Convert date and time to string format

Syntax

```
S = datestr(V)
S = datestr(N)
S = datestr(D, F)
```

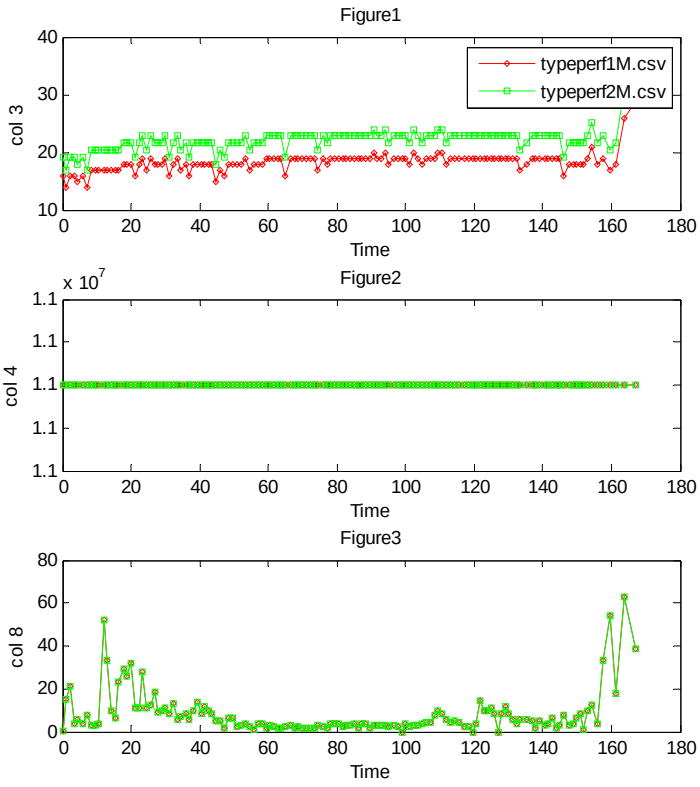
	1
1	'(PDH-CSV 4.0) (Eastern Standar... '
2	'11/06/2006 18:33:01.957 '
3	'11/06/2006 18:33:03.078 '
4	'11/06/2006 18:33:04.400 '
5	'11/06/2006 18:33:05.442 '

```
timeDate=A.textdata(2:end,1); % time data
timeUTC=datenum(timeDate); % transform time to UTC
relUTCTime=timeUTC-timeUTC(1); % define relative time
T=datestr(relUTCTime, 'MMSSFFF'); % time in minutes, se
```

T =

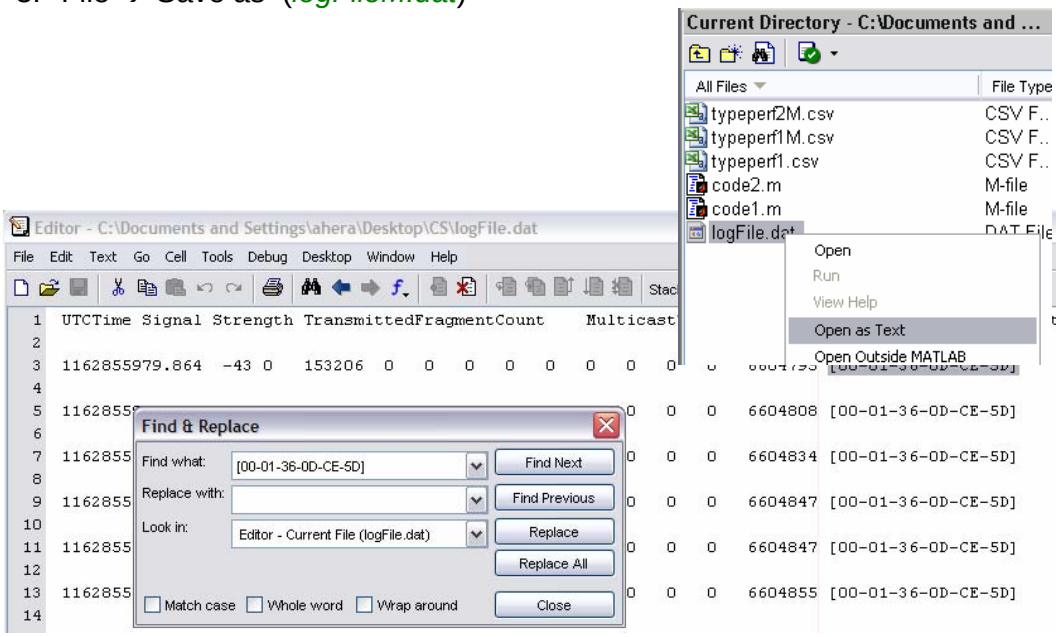
min sec msec
0000000
0001121
0002443
0003484
0004525
0006068

4. Plot Data



1. Data Preparation

1. Open as Text (logFile.dat)
2. Editor window: Edit→ Find and Replace → *Remove* [00-01-36-0D-CE-5D]
3. File → Save as (*logFileM.dat*)

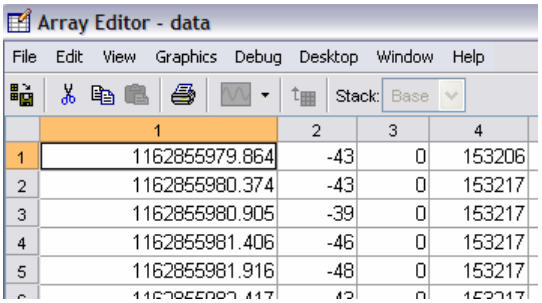


2. Import Data in Matlab

Row no

Col no

```
dataExp=dlmread('logFileM.dat', '', 1, 0);
```



dlmread

Read ASCII-delimited file of numeric data into matrix

Graphical Interface

As an alternative to `dlmread`, use the Import Wizard. To activate t

Syntax

```
M = dlmread('filename')
M = dlmread('filename', delimiter)
M = dlmread('filename', delimiter, R, C)
M = dlmread('filename', delimiter, range)
```

3. Process Data

```
timeUTC=dataExp(:,1);
relTime=timeUTC-timeUTC(1);
```


3. Plot Data & Extract Info

Processing Experimental Data for the Project

1. **Data Preparation**
2. **Import Data in Matlab**
3. **Process Data**
4. **Plot Data**

A. Files: typeperf1.csv, typeperf2.csv, etc — ***code1.m***

B. Files: logFile1.dat, logFile2.dat, etc — ***code2.m***