203484 selected topic in transportation
2552/2

Introduction to Matlab 2

Prepared by Weerakaset Suanpaga

Overview

Topic No#1

Matlab (Matrix Laboratory) 3/04/2008

- Writing a program: Environment, m files, path, editor, writing programs, subroutines
- Variable types – arrays, strings
- Basic operations: if/elseif/else, switch, for
- Input / Output
- Plotting
- Debugging
- Speed

Matlab: What is it + Why Use It?

- What is it?
  - Easy to use programming language for mathematically oriented programs
  - Great for numerical computing, bad for web design
- Why use it?
  - Matlab is often slow, not very elegant, not universally available, not free (although Octave is a freeware version), but …
  - Very easy to write programs (good development environment, simple data structures and syntax, nice graphics), very large libraries of numerical routines (including specialized toolboxes for differential equations, bioinformatics, finance, etc)
  - Very popular in University classes and industry

Writing Programs

- **Environment**
  - Command line input (quick way to test syntax, etc.)
  - All variables stay in memory for access (but this can be confusing if you don't clear them). Begin main program by clearing variables with
    - clear; clear global;
- **M-files:** Put each program, subroutine in its own file, *.m. All m-files begin with the lines
  - Main program: nothing, just type commands
  - Subroutines: function [outvar1,outvar2,…]=Func_Name(invar1,invar2,…)
  - Name the m-file for Func_Name with Func_Name.m
- **Path:** Put all m-files in one directory and add it to your path (file/set path). Also set "Current Directory" (in toolbar at top of Matlab window) to that directory.
- **Editor:** file/open or file/new allows you to use the Matlab editor to make m-files (I recommend this for its formatting features).

Variables

- The basic variable in Matlab is an array
- "a=7;" creates a 1x1 array (scalar)
- "a=[1,2,3;4,5,6];" creates a 2x3 array $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
- "a='array';" creates a 1x5 array of characters
- a(i,j) refers to element in row i of column j of matrix a
- There are many array operations, e.g., diag(a) will diagonalize a.
- Structures are very convenient to store related variables
  - s.name='s is a structure'; s.size=2;
- The ":" operator gives sets of variables
  - >> 1:3 is 1 2 3
  - >> 10:-1:5 is 10 9 8 7 6 5
  - >> a(1:3;j) is a(1,j) a(2,j) a(3,j)
- Many things are vectorized and can work with matrix input vectors x,y
  - plot(x,y);
  - y=2*x;
  - dot_prod = x*y';
  - Pointwise_prod = x.*y;

Basic Flow Control

if, elseif, else

```
if A > B
   s='A greater';
elseif A < B
   s='A less'
elseif A == B
   s='A,B equal'
else
   s='Unexpected situation'
end
```

switch

```
switch (int(A-B))
   case -1
      s='A less';
   case 0
      s='A,B equal'
   case 1
      s='A greater'
   otherwise
      s='Unexpected situation'
end
```

for

```
for i=1:5
   sum=sum+i;
end
```


Input and Output (I/O)

- **Write to screen**
  - disp(['Variable x has value: ' num2str(x)]);
  - x (with no ";" will print x)
- **Write to file**
  - Special Matlab routines (e.g., csvwrite('matrix.txt',M))
  - Open file: fid=fopen('file.txt',w);
  - Write to file: fprintf(fid,'%f\n',x); (c style formatting)
  - Close file: fclose(fid); (must do this to have it write)
- **Read from file**
  - Special Matlab routines (e.g., M = csvread('filename'))
  - C style: a = fscanf(fid,'%g %g',[m n]); returns mxn matrix a of data. % Vectorized function
  - Simple defining function for input values: In InitData.m type

    Function [indata]=InitData;

    indata.a=10; Indata.b=50;

Plotting

- **Make plot**
  - Very extensive graphics, 2D, 3D, contour, movies, etc.
  - Simple example to plot a 2 dependent variables and compare (with labels)

```
% Set up data
t=(1:.1:10);
cos_vals=cos(t); % Vectorized function
sin_vals=sin(t);
% Plot data
hold on; % Keeps plot from overwriting
title_text={['Cos and Sin Functions', 'date= ' date]};
title(title_text);
xlabel('Time');
ylabel('Functions');
plot(t,cos_vals,'b:');
plot(t,sin_vals,'r-');
hold off;
```


Debugging

- Great simple graphical debugger when using Matlab editor
- Can set breakpoints (where code stops execution)
- All variables are available in workspace at that time to examine
- Can step one line at a time, into routines, or continue until next breakpoint

Speed

- Easy html output profiling with profile
  - > profile on
  - > my_really_slow_code
  - > profile viewer
  - > profile off
- Timing given for each routine, easy to search through them by following links
- Matlab can be very slow
  - For loops are very slow (vectorizing can give x1000 speedup!!)
  - Vectorized operations are fast
  - Basic built in routines (e.g., matrix diagonalization) are fast (they are just C routines
  - Can call C,C++ functions for better speed

Functions (Toolboxes)

- Many built in functions that are easy to use
  - Transpose of M: M'
  - Eigenvalues of M: diag(M)
  - Symbolic equation solutions:

```
solve('a*x^2+b*x+c=0','x')

ans =

    1/2/a*(-b+(b^2-4*a*c)^(1/2))
    1/2/a*(-b-(b^2-4*a*c)^(1/2))
```

  - *Etc.*
- If you are doing any numerical task, from an integral to a permutation to a neural network, there is probably a built in function.

- **MATLAB Toolboxes**
  - Bioinformatics Toolbox
    Communications Toolbox
    Control System Toolbox
    Curve Fitting Toolbox
    Data Acquisition Toolbox
    Database Toolbox
    Datafeed Toolbox
    Excel Link
    Filter Design Toolbox
    Filter Design HDL Coder
    Financial Toolbox
    Financial Derivatives Toolbox
    Financial Time Series Toolbox
    Fixed-Income Toolbox
    Fixed-Point Toolbox
    Fuzzy Logic Toolbox
    GARCH Toolbox
    Genetic Algorithm and Direct Search Toolbox
    Image Acquisition Toolbox
    Image Processing Toolbox
    Instrument Control Toolbox
    Mapping Toolbox
    Model-Based Calibration Toolbox
    Model Predictive Control Toolbox
    Neural Network Toolbox
    Optimization Toolbox
    OPC Toolbox
    Partial Differential Equation (PDE) Toolbox
    RF Toolbox
    Robust Control Toolbox
    Signal Processing Toolbox
    Spline Toolbox
    Statistics Toolbox
    Symbolic Math Toolbox
    System Identification Toolbox
    Virtual Reality Toolbox
    Wavelet Toolbox
- Also Simulink

Miscellaneous

- Comment lines begin with "%"
- Almost all command lines end with ";"
- Help
  - ">> help FUN" gives help on function FUN.
  - ">> help help" gives info on help.
  - Huge amount online at www.mathwork.com and all over web.
- Everything can be tested on the command line
- Begin programs with "clear; clear global;" to erase variables.

Defining a function – the first line

```
function <outputVars> = <function name>(<inputVars>)
```

- Keyword - has to be exactly as appears
- 0, 1, or more vars. If more than 1, put in square brackets.
- required
- You choose the name - see text or ML help for valid name - mostly just start with a regular character and have no spaces.
- 0, 1, or more vars. If 0, then the enclosing parens are not needed.

Linkage Between Actual and Formal Parameters

calling program:

```
...
z = myFun3(a,b)
...
```

the function called:

```
function out33 = myFun3(x,y)
```

When the function myFun3 is "called"…

1. The formal input variables (x,y) take the values given in the calling line (a,b)
2. The function "runs"
3. The output variables in the function are given back to the calling program's variable.

Comments in Functions

- There are no uniformly agreed upon rules for inserting comments into functions
- It is always good programming practice to include comment lines indicating:
  - the purpose of the function
  - the inputs to the function
  - the outputs from the function
  - any assumptions

Examples

```
function <outputVars> = <function name>(<inputVars>)
```

For each of the following function definitions, how many input and output variables are there?

function x = myFun1
function z = myFun2(y)
function out33 = myFun3(x,y)
function [a,b] = myFun4(q,r)

Examples

Problem 4-B.15

You need to write a MATLAB function to compute ***L(x)*** given by the following equation (where $0 < x < 1$):

$$L = 100\left(\frac{x}{0.6}\right)^{0.625}\left(\frac{1-x}{0.4}\right)^{-1.625}$$

The function you write as a first draft is the following:

```
function L = computeL(x)
% This function calculates L(x) per the problem spec
% Input: x
% Output: L

  L = 100 * x/0.6^0.625 * 1-x/0.4^-1.625;
```

Find and correct the errors in your first draft.

Vector Operations

Topic No# 2

Vector Operations

- Vector Creation
- Accessing Vector Elements
- Row Vectors and Column Vectors, and the Transpose Operator
- Vector Built-in Functions, Operators, and Expressions

2-1 Vector Creation

- Vectors are defined in square brackets;
  ```
  temperaturesMonThu = [32 31 29 33];
  temperaturesFriSat = [35 33];
  ```
- You can concatenate a vector with a scalar;
  ```
  temperaturesFriSun = [35 33 27];
  ```
- or concatenate 2 vectors;
  ```
  weeklyTemperatures = [temperaturesMonThu, temperaturesFriSun];
  ```
- To find the size of a vector, we use length;
  ```
  numTemperatures = length(weeklyTemperatures);
  ```
- We could find the average temperature by typing;
  ```
  avgTemperature = mean(dailyTemperatures)
  ```
- or by using sum and length;
  ```
  totalTemperature = sum(dailyTemperatures)/length(dailyTemperatures);
  ```


Some Useful Vector Functions

- **brackets (e.g. [27 36 41]): Creates vectors.**
- **colon operator (e.g. [0:5:30]): Creates linearly spaced vectors.**
- ***linspace* (e.g. linspace(0,100,21)): Creates linearly spaced vectors.**
- ***length* (e.g. length([0:5:30])): Finds the length of a vector.**
- ***zeros* (e.g. zeros(1,5)): Creates vectors filled with zeroes.**
- ***ones* (e.g. ones(1,5): Creates vectors filled with ones.**
- ***sum* (e.g. sum([5 3 6 2])): Sums up the contents of a vector.**
- ***sort* (e.g. sort([5 3 6 2])): Sorts the contents of a vector.**
- ***mean* (e.g. mean([5 3 6 2])): Finds the average of contents.**

2.2 – Accessing Vector Elements – Examples

1. **Create a row vector x consisting of the numbers in the ordered set: {1 4 7 10} using the colon operator.**

   ```
   x = [1:3:10]
   ```

2. **Set a variable y to be the length of x.**

   ```
   y = length(x)
   ```

3. **Set variable y to be the 1st element of x.**

   ```
   y = x(1)
   ```

4. **Set variable y to be the 1st, 2nd, and 3rd elements of x.**

   ```
   y = x([1,2,3]) OR y = x(1:3)
   ```


Accessing Vector Elements – Example cont'd.

5. **Set variable y to be the 3rd through the last element of x - and do so such that your solution works no matter how long x is.**

   ```
   y = x(3:end)
   ```

6. **Set variable y to be the next-to-last and last element of x - and do so such that your solution works no matter how long x is.**

   ```
   y = x([end-1,end])
   ```

7. **Change the 2nd element of x to be 3.**

   ```
   x(2) = 3
   ```

8. **Change the 2nd element of x to be 102 and the 4th element of x be 205.**

   ```
   x([2.4]) = [102. 205]
   ```

Synopsis for Fetching and Setting Elements in Vectors

- Access to whole vector is similar to scalar access.
- Accessing element(s) in a vector is done by indexing into the vector.
- To delete element(s) in a vector, empty square brackets are used.
- To find the length of a vector V, use the length built-in function length(V).
- When setting elements of a vector, the number of elements being set must be equal to the number of elements in the vector on the right hand side of the assignment operation. The exception is that a scalar on the right-hand side can be used to set multiple vector elements.

2-3 Row Vectors and Column Vectors, and the Transpose Operator

- Row and column vectors are represented as single rows and columns of values, respectively.
- When creating a column vector with square brackets, you may use the semicolon operator:

```
temp = [35; 33; 27];
```

- or you may use the transpose operator;

```
temp = [35 33 27]';
```

- When creating an equally spaced column vector, you need to use the transpose operator;

```
springConstants = [10:10:100]';
springConstants = linspace(10,100,10)';
```


2-4 Vector Built-in Functions, Operators, and Expressions

Sample Problem – Vector Built-in Functions

Problem 5-A.30 (Section 5-4)

Define **x** as follows:

```
>> x = [3; 9; 4; 5; 2]
```

Create a variable **mySort**, which contains all values in **x** in sorted order.

Sample Problem – Vector Arithmetic Operators

Problem 5-A.31 (Section 5-4)

In MATLAB, define variables **x** and **y** as follows:

```
>> x = [3; 9; 4; 5; 2]
>> y = [2; 1; 4; 3; 2]
```

Create a new vector **z** in which each element is the sum of the corresponding elements in **x** and **y**.

Sample Problem – Vector Arithmetic Operators

Problem 5-A.33 (Section 5-4)

In MATLAB, define variables **x** and **y** as follows:

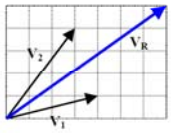
```
>> x = [3; 9; 4; 5; 2]
>> y = [2; 1; 4; 3; 2]
```

Create a new vector **z** in which each element is the product of the corresponding elements in **x** and **y**.

Sample Problem – Vector Arithmetic Operators

Problem 5-B.14

A two-dimensional physics vector can be represented in MATLAB by a collection of its x and y components. For example, V_1 and V_2 in the figure below could be represented as (4,1) and (3,4), respectively. The summation of two vectors is carried out by adding the components of vectors independently. In the example below, V_3, which is the sum of V_1 and V_2, is calculated as (4+3, 1+4). If vectors represent forces, a summation of vectors is said to be the Resultant Force.



Write a MATLAB function that accepts two arbitrary forces in a two-dimensional space and returns the resultant force.

Vector Relational Operators

- Think of them as comparing numbers…

  <, >, = =, >=, <=
- A relational operator can be used to compare the values of two variables

  a>b
- But… remember MATLAB is for matrices

  what are you testing?

Sample Problem – Vector Relational Operators

Problem 5-A.38 (Section 5-4)

In MATLAB, define variables **x** and **y** as follows:

```
>> x = [3; 9; 5; 5; 1]
>> y = [2; 1; 4; 6; 2]
```

Find the indices for all elements in **x** such that corresponding elements of **y** are twice as big.

Vector Logical Operators

- They operate on the ***results*** of relational operators
- How many elements in vector x are in range (6,10)?
- How many elements in x are
  - … greater than 6
    AND
  - … less than 10?
- We use logical operators…
  - AND (&), OR (|), NOT (~)
  - any, all

Sample Problem – Vector Logical Operators

Problem 5-A.42 (Section 5-4)

In MATLAB, define variables **x** and **y** as follows:

```
>> x = [3; 9; 5; 5; 1]
>> y = [2; 1; 4; 6; 2]
```

Set all the elements in **x** that have values between 4 and 10 to the value of the corresponding value in **y**.

2-D Plotting and Help in MATLAB

Topic No#3

2-D Plotting and Help in MATLAB

- Using EZPLOT to Plot Functions
- Using Vectors to Plot Numerical Data
- Overlay plots and subplots
- Other 2-D plot types in MATLAB
- Problem Sets for 2-D Plotting

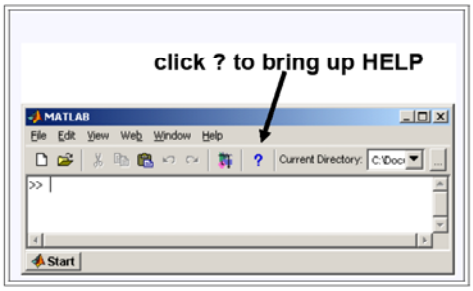
3-1 Using EZPLOT to Plot Functions

Getting Help

- You can't possibly learn everything there is to know about MATLAB,

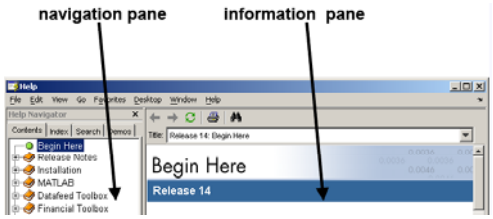
  … and you don't need to.
- It is crucial to develop the ability to augment your knowledge in MATLAB toward accomplishing a given task.

Getting Help cont'd



Getting Help cont'd

- Click the tab in the navigation pane labeled **Search**.
- Then type into the Search field the name **ezplot**.



Using EZPLOT to Plot Functions

- There are three forms of ***ezplot***:
  - f(x) e.g., $f(t) = 3e^{-2t}\cos(5t)$

    ezplot('3*exp(-2*t)*cos(5*t)')
  - f(t), g(t) e.g., $f(t) = 3t^2 + 2$; $g(t) = \sin(5t)$

    ezplot('3*t^2 + 2', 'sin(5*t)')
  - f(x,y) = 0 e.g., $f(x,y) = 3xy + y^2 + 55 = 0$

    ezplot('3*x*y + y^2 + 55',[-30,30,-20,20])

Sample Problem - EZPLOT

Problem 6-A.4 (Section 6-1)

Use **ezplot** to determine the roots graphically of the following equation in the interval $[0, 2\pi]$:

$$x\tan(x) = 9$$

Use the grid and zoom facilities of MATLAB for more accurate answers.

Label the roots in the plot using **text** or **gtext**. Save the figure and call it **Prob6_A_4.fig**.

Graphing with MATLAB

- Use ***ezplot*** to make a quick and dirty chart of functions.
- Optional arguments allow changing the default functional domain [-2π, 2π].
- Use ***xlabel***, ***ylabel***, and ***title*** built-in functions to refine labeling the plots made by ***ezplot***.
- When needed, use ***grid*** to activate a grid on a plot created.
- If you would like to keep the existing graph and generate a new one, use ***figure***.

3-2 Using Vectors to Plot Numerical Data

- Mostly from observed data - your goal is to understand the relationship between the variables of a system.

Speed (mi/hr)	20	30	40	50	60	70
Stopping Distance (ft)	46	75	128	201	292	385

- Determine the independent and dependent variables and plot:

```
speed = 20:10:70;
stopDis = [46,75,128,201,292,385];
plot(speed, stopDis, '-ro') % note the '-ro' switch
```

- Don't forget to properly label your graphs:

```
title('Stopping Distance versus Vehicle Speed', 'FontSize', 14)
xlabel('vehicle speed (mi/hr)', 'FontSize', 12)
ylabel('stopping distance (ft)', 'FontSize', 12)
grid on
```


Sample Problem – Plotting Numerical Data

Problem 6-A.10 (Section 6-2)

The data table below shows power dissipation for varying magnitudes of electric current in a circuit.

Current (amperes)	0	5	10	15	20	25
Power Dissipation (watts)	0	175	700	1575	2800	4375

Plot the data. Mark the data points with circles and connect them with a red solid line. Don't forget to label the x and y axes and to create a title for your graph.

Save the figure you generate as **Prob6_A_10.fig**.

Plotting Functions Numerically

- ***ezplot*** is a great tool for plotting functions, but it has several disadvantages:
  - it doesn't provide as much control as ***plot***, e.g. dotted lines.
  - you must fill in values for any constants, e.g.
- When you need more control, plot numerically with ***plot***:

$$V_{cylinder} = \frac{\pi d^2}{4} h$$

```
d = 4;
h = linspace(1,10); % Step 1 - create vector for independent variable
V = pi*d^2/4*h; % Step 2 – compute vector for dependent variable
plot(h,V,'-r') % Step 3 - plot and label
xlabel('height (m)', 'FontSize', 12)
ylabel('Volume (m^3)', 'FontSize', 12)
title('Volume of a cylinder versus its height','FontSize', 14)
grid on
```


Sample Problem – Plotting Functions Numerically

A function G(x,y,z) of three independent variables is defined as:

$$G(x, y, z) = \frac{ze^{-0.3/x}}{\ln(y)\sqrt{x}}$$

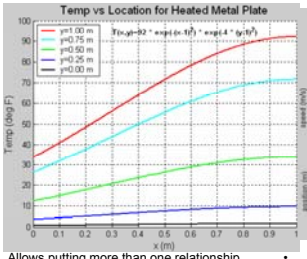
Write a functi utputs but creates a plot of G(x,y,z), s

y = 5, z = 3

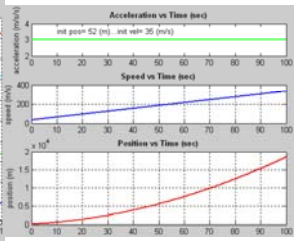
Synopsis for ezplot and plot

- The first argument to plot should be the vector of values for the independent variable (going on the x-axis); the second argument should be the vector of values for the dependent variable (going on the y-axis).
- An optional third argument plot is the line spec which specifies the type of line used (solid, dotted, etc.), the color of the line used, and the type of data marker (if any).
- For plotting numerical data from experimentation or observation, use data markers.
- For plotting numerical data that are computed from a mathematical relationship, data markers must not be used.

3-3 Overlay Plots and Subplots



- Allows putting more than one relationship directly into the same plotting window.
- Two key functions: ***hold*** and ***legend***



- For multiple dependent variables whose data are not of the same type, e.g. acceleration, speed and distance
- Key function to learn: ***subplot***

Sample Problems – Overlay Plots and Subplots

Problem 6-A.18 (Section 6-3)

The following sample data represents the number of drivers that were ticketed on a given freeway in Michigan for traveling over 75 mph and for traveling under 55 mph.

Create two column vectors: **`over75mph`** and **`under55mph`** that each have twelve elements as given in the data table above.

	Jan	*Feb*	*Mar*	*Apr*	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
Over 75 mph	38	29	43	51	67	84	79	95	73	55	46	19
Under 55 mph	12	11	9	3	4	2	0	1	3	7	11	14

Create a function **`Prob6_A_18(over75mph, under55mph)`**.

Your function should generate an overlay plot of high and low speeders versus months (use month numbers). Mark the data points for **`over75mph`** with triangles pointing up and **`under55mph`** with triangles pointing down. Use different colors and widths of line for connecting the data points. Label your graph appropriately and generate a legend.

Sample Problems – Overlay Plots and Subplots

Problem 6-A.23 (Section 6-3)

Engineers often use the *small angle approximation* to simplify their calculations. An example application of small angle approximation is the calculation of angular displacement for small motions of a pendulum.

Here is one version of the small angle approximation:

$$\sin(x) \approx x$$

This is used when **x** is small.

Write a function, **`Prob6_A_23`**, to demonstrate graphically the validity of this approximation for angles from 0 to 20 degrees.

In a single figure, show the three separate plots:

a) An overlay plot of $y1 = \sin(x)$ and $y2 = x$ in the interval $[0, 40]$ degrees. Put a legend on your plot.

b) A plot of the absolute error, i.e., $y = abs(\sin(x) - x)$

c) A plot of the relative error, i.e., $y = abs\left(\frac{\sin(x) - x}{\sin(x)}\right)$

Synopsis for Overlay Plots and Subplots

- Overlay plots are used to show a family of parameterized results
- ***hold on*** is the key MATLAB command needed to turn on overlays
- Subplots are used to display plots of different independent variables usually from one experimental data set or from one set of equations for a single physical system.
- ***subplot*** is the key MATLAB command needed to identify the target for a created plot.

Web Help

- The total Mathworks doc:
  - http://www.mathworks.com/access/helpdesk/help/helpdesk.html
  - For matlab specifically: http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
- Useful tutorials
  - Getting started: http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.html
  - A good intro tutorial: Maybe work through this one! http://www.mathworks.com/academia/student_center/tutorials/launchpad.html
  - Links to many more tutorials: http://www.mathworks.com/matlabcentral/link_exchange/MATLAB/Tutorials/