# MATLAB Tutorials

16.01 & 16.02 Unified Engineering - Fall 2006

Violeta Ivanova, Ph.D.
Educational Technology Consultant
MIT Academic Computing

violeta@mit.edu
http://web.mit.edu/violeta/www

Information Services and Technology

# What is MATLAB?

- Computational Software

  From The MathWorks: www.mathworks.com

- MATrix LABoratory

- Algorithm Development Environment

  … with built-in capabilities of a high-level programming and scripting language

# This Tutorial

- Class materials

  web.mit.edu/acmath/matlab/unified/
- Topics
  - Interface & Basics
  - Matrix Mathematics
  - Programming

# Other References

- Mathematical Tools at MIT

  web.mit.edu/ist/topics/math
  - MATLAB Mastery I (beginners' tutorial)
  - Introduction to MATLAB (IAP series)
- Other Course 16 MATLAB Tutorials

  web.mit.edu/acmath/matlab/course16/

# MATLAB @ MIT

- On Athena
  - 250 floating licenses (free)
- For student-owned computers
  - 300 floating licenses (free)

    http://matlab.mit.edu

# MATLAB Interface & Basics

Desktop Interface

Variables, Vectors, Matrices

Built-In Functions

IST Information Services and Technology

MIT

# Starting MATLAB

- On Athena

```
athena% add matlab
athena% matlab &
>> desktop
```

- On laptops

  Desktop interface starts by default.

# MATLAB Desktop Interface

*You must be running MATLAB now …*

- Default desktop
  - Command Window
  - Current Directory Window
  - Command History Window
  - Menu Toolbar

# Help in MATLAB

- Command line help

  `>> help <command>`

  e.g. `help polyval`

  `>> lookfor <keyword>`

  e.g. `lookfor polynomial`

- Help Browser
  - Help->Help MATLAB

# MATLAB Help Browser

- MATLAB
  - + Getting Started
    - + Desktop Tools and Development Environment
  - + Mathematics
    - + Matrices and Linear Algebra
  - + Data Analysis
  - + Programming
    - + Data Types
    - + M-File Programming
  - + Graphics
- *Other Toolboxes*

# Variables

- Begin with an alphabetic character: `a`
- Case sensitive: `a, A`
- No data typing: `a=5; a='ok'; a=1.3`
- Default output variable: `ans`
- Built-in constants: **`pi i j Inf`**
- **`clear`** removes variables
- **`who`** lists variables
- Special characters

  **`[] () {} ; % : = . … @`**

# Vectors

- Row vector

```
>> R1 = [1 6 3 8 5]
>> R2 = [1 : 5]
>> R3 = [-pi : pi/3 : pi]
```

- Column vector

```
>> C1 = [1; 2; 3; 4; 5]
>> C2 = R2'
```

# Matrices

- Creating a matrix

```
>> A = [1 2.5 5 0; 1 1.3 pi 4]
>> A = [R1; R2]
```

- Accessing elements

```
>> A(1,1)
>> A(1:2, 2:4)
>> A(:,2)
```

# Polynomials

- Evaluating polynomials

$$y = p_1 x^n + p_2 x^{n-1} \ldots + p_n x + p_{n+1}$$

```
>> p = [p1 p2 … ]
>> t = [-3 : 0.1 : 3]
>> z = polyval(p, t)
```

- Integrating polynomials

```
>> P = polyint[p]
>> area = polyval(P, a) - polyval(P, b)
```

# Curve Fitting

- Fit a polynomial through points ($x_i$, $y_i$)

```
>> X = [x1 x2 … xn];
>> Y = [y1 y2 … yn];
```

- Built-in function `polyfit`

```
>> C1 = polyfit(X, Y, 1);
>> C2 = polyfit(X, Y, 2);
>> Cm = polyfit(X, Y, m);
```

# Graphics

- 2D linear plots

```
>> plot (X, Y, 'ro')
>> plot (X, Y, 'Color', [0.5 0 0], …
         'Marker', 'o', …
         'LineStyle', 'none')
```

- Colors: b r g y m c k w
- Markers: o * . + x d
- Line styles: - -- -. :

# Multiple Graphs on One Plot

- Built-in function `hold`

```
>> g1 = plot(X, Y, 'ro')
>> hold on
>> g2 = plot(x, y1, 'b-')
>> hold on
>> g2 = plot(x, y2, 'g--')
>> hold off
```

# Subplots on One Figure

- Built-in function `subplot`

```
>> s1 = subplot(1, 3, 1)
>> g1 = plot(x, y1)
>> s2 = subplot(1, 3, 2)
>> g2 = plot(x, y2)
>> s3 = subplot(1, 3, 3)
>> g3 = plot(x, ym)
```

# Customizing Graphs

- Annotating graphs

```
>> plot (X, Y, 'ro')
>> legend ('Points')
>> title ('Coordinates')
>> xlabel ('X')
>> ylabel ('Y')
```

- Plot Edit mode: icon  in Figure editor
- Property Editor: `View->Property Editor`
- Saving figures: `File->Save As`

# MATLAB Matrix Mathematics

Built-In Functions

Matrix Operations

Linear Equations

Information Services and Technology

# Linear Algebra Functions

- Matrices & vectors

```
>> [n, m]= size(A)
>> n = length(X)
>> det(A)
>> M1 = ones(n, m)
>> M0 = zeros(n, m)
>> En = eye(n)
>> N1 = diag(En)
>> [evecs, evals] = eig(A)
>> rank(A); trace(A)
```

# Matrix Operations

- Operators **+** and **–**

```
>> X = [x1 x2 x3];
>> Y = [y1 y2 y3];
>> A = X + Y
   A =
      x1+y1   x2+y2   x3+y3
```

- Operators *****, **/**, and **^**

```
>> Ainv = A^-1
```

Matrix math is default!

# Element-Wise Operations

- Operators **.***, **./**, and **.^**

```
>> Z = [z1 z2 z3]'
>> B = [Z.^2   Z   Z.^0]
B =
    z1^2   z1   1
    z2^2   z2   1
    z3^2   z3   1
```

# Solving Linear Equations

- Example:

  Find the quadratic that passes through three points $(x_1,y_1)$, $(x_2,y_2)$, and $(x_3,y_3)$, i.e. find $(c_1, c_2, c_3)$ such that:

$$y_1 = c_1 x_1^2 + c_2 x_1 + c_3$$

$$y_2 = c_1 x_2^2 + c_2 x_2 + c_3$$

$$y_3 = c_1 x_3^2 + c_2 x_3 + c_3$$

# Solving Linear Equations (continued)

- Matrix form: Y = AC
- Solution:
  - Define matrix A

```
>> A = [X.^2   X ones(3,1)];
```

  - Solve matrix equation

```
>> C = A \ Y
```

# Matrix Math Exercise

- Exercise 1: `MatrixMath.m`
  - Matrices and vectors
  - Element-wise and matrix operations
  - Systems of linear equations
  - Eigenvalues and eigenvectors
  - Polynomials
  - Curve fitting
  - 2D plotting
- *Follow instructions in exercise handout …*

# MATLAB Programming

Data Types & Operators
Program Control Statements
Script & Function M-Files

IST Information Services and Technology

MIT

# Data Types

- Numeric

```
>> x = 5; y = 5.34; z = 0.23e+3
```

  - Default: double-precision floating point
  - Can be converted to integers, etc.
  - Numeric manipulation

```
>> y = 5.3456;
>> x = round(y);
>> format long
```

  - Complex numbers

```
>> x = 5 + 6i
```

# Data Types (continued)

- Characters and strings

```
>> a = '5'
>> b = 'Hello'
```

  - String conversions

```
>> x = 5; a = num2str(x)
>> a = '5';     x = str2num(a)
```

  - String manipulations

```
>> isempty(b)
>> strcmp(b, 'hi there')
>> abc = lower('ABC')
```

# Data Types (continued)

- Keywords

  `if, switch, for, end, global, for, …`

  DO NOT USE AS VARIABLE NAMES!
- Special variables

  `nargin, pi, i, j, …`
- Structures

```
person.name = 'Jane'; person.age = 20
```
- Cell Arrays

```
person = {'Jane' 'female'; 20  1996}
```

# Operators

- Arithmetic: `x+y;  A*B;  X.*Y;` etc.
- Logical
  - Element-wise AND: `a & b`
  - Element-wise OR: `a | b`
  - "Short cuts": `||` and `&&`
- Relational

  `a == 5;  a >= b;  b ~= 6;`
- Operator precedence

  `()  {}  []` -> Arithmetic -> Relational -> Logical

# M-File Programming

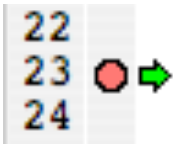
- Script M-Files
  - Automate a series of steps.
  - Share workspace with other scripts and the command line interface.
- Function M-Files
  - Extend the MATLAB language.
  - Can accept input arguments and return output arguments.
  - Store variables in internal workspace.

# A MATLAB Program

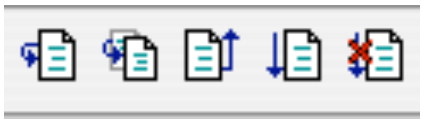
- Always has one script M-File
- Uses built-in functions as well as new functions defined in function M-files
- Created in MATLAB Editor / Debugger

```
>> edit program.m
```

  - Debugging mode




```
k >>
```



# Function M-Files

Example: orbitalvelocity.m

```
function V = orbitalvelocity(R, g, H)
% H1 line: ORBITALVELOCITY computes V.
% Help text: this text appears when
% you type "help orbitalvelocity".

% Comment: function body is below
V = sqrt( g * R^2 / (R + H);
return
```

# Variable Types

- Local (default)
  - Every function has its own local variables.
  - Scripts share local variables with functions they call and with the base workspace.
- Global

```
global speedoflight  windspeed
```

  - Functions, scripts, and the base workspace share global variables.
- Persistent

```
persistent R, C
```

  - Can be declared and used only in functions.

# Program Flow Control: `for`

```
x = [1 : 0.01 : 10]; a = 60; b = 30;
N = length(x);
y = zeros(1, N);
for n = 1 : N
   y(n) = a - b*cos(pi/3 + x(n)*pi/6)
end
P = plot (x, y, 'ro')
```

# Program Flow Control: `if`

```
if strcmp(planet, 'Earth')
        R = 6376; g = 9.814;
elseif strcmp(planet, 'Mars')
        R = 3396; g = 3.688;
else
        R  = input('Enter R: ');
        g  = input('Enter g: ');
end
```

# Program Flow Control: `switch`

```
switch units
   case 'metric'
        R = 6376; g = 9.814;
   case 'English'
        R = 3963; g = 32.2;
   otherwise
        error('Unknown units.')
end
```

# Command Window I/O

- Get input from Command Window

```
num = input('What altitude: ')
str = input('Which planet: ', 's')
```

- Display output in Command Window
  - Strings

```
disp('Velocity is 500.')
error('Unknown units.')
```

  - If there are numbers to display:

```
message = ['Velocity is: ' str2num(V)]
disp(message)
```

# File Input / Output

- Import Wizard for data import

  `File->Import Data …`

- File input with `load`

```
B = load('datain.txt')
```

- File output with `save`

```
save('dataout', 'A', '-ascii')
```

# Programming Exercise

- Exercise 2: `velocityprogram.m`
  - User defined functions: `orbitalvelocity.m`
  - Funciton and script M-Files
  - Program flow control: `if` and `switch` statements
  - Control Window input and output
- *Follow instructions in exercise handout …*

# Resources

- web.mit.edu/ist/topics/math
- web.mit.edu/acmath/matlab/unified
- web.mit.edu/acmath/matlab/course16

## Questions?