# Introduction to MATLAB

Violeta Ivanova, Ph.D.
Office for Educational Innovation & Technology

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

# Topics

- MATLAB Interface and Basics
- Calculus, Linear Algebra, ODEs
- Graphics and Visualization
- Basic Programming
- Programming Practice
- Statistics and Data Analysis

# Resources

- Class materials

  http://web.mit.edu/acmath/matlab/IAP2007

  - Previous sessions: InterfaceBasics, Graphics
  - This session: Programming <.zip, .tar>

- Mathematical Tools at MIT web site

  http://web.mit.edu/ist/topics/math

# MATLAB Help Browser

- MATLAB
  - + Data Types
  - + Basic Program Components
    - + Variables
    - + Operators
      - + Arithmetic Operators
      - + Relational Operators
      - + Logical Operators
    - + Program Control Statements
  - + M-File Programming
    - + M-File Scripts and Functions

# MATLAB Programming Basics

Variables and Operators

Program Control Statements

Script and Function M-Files

# Computer Languages

- Machine language (lowest level)

  1 0 1 0 1 0 1 1 1 0 0 0 1 1 0 1

- Higher level languages
  - Programming languages: Java, C++
  - Scripting languages: Perl
  - Markup languages: HTML, mathML
  - etc.

# What is MATLAB?

- Computational Software
- MATrix LABoratory
- Algorithm Development Environment

… with some built-in abilities of high-level programming and scripting language.

# 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.

# Function M-Files

## Example: odeLV.m

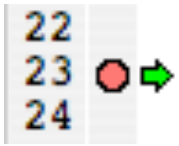
```
function DV = odeLV(t, V)
% H1 line: ODELV computes Mars lander's dV/dt.
% Help text: this text appears when you type
% "help odeLV" in the Control Window.

Gm = 3.688; K = 1.2; M = 150;
DV = sqrt( Gm - K/M * V.^2);
return
```

# A MATLAB Program

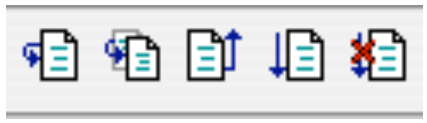
- Always has one script M-File
- Uses built-in functions as well as new functions defined in function M-files
- Created in MATLAB Editor / Debugger

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

  - Debugging mode




```
k >>
```



# 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.

# File I/O in M-Files

- Commands `load` and `save`

```
data = load('datain.txt', '-ascii')
save('dataout.txt', 'A', '-ascii')
```

- Open browser for input with `uigetfile`

```
[filename, pathname] = uigetfile( …
      {'*.txt', 'Get Text Files'}, …
       'Pick a file')
filename = [pathname filename]
data = load (filename, '-ascii');
```

# Basic Programming Exercises

- Exercise One: `loadpoints.m`, `plotdata.m`, `longscript.m`, `shortscript.m`
  - User defined functions
  - Function and script M-files
  - File input with `uigetfile`

*Follow instructions in m-files …*

# 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')
```

# 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: ' str2num(V)]
disp(message)
```

# More Data Types

- Keywords

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

  DO NOT USE AS VARIABLE NAMES!
- Special variables

  `nargin, pi, i, j, …`
- Other
  - 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

# 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 + x(n)*pi/6)
end
P = plot (x, y)
```

# 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
```

# Basic Programming Exercises

- Exercise Two: `velocityprogram.m`
  - User defined functions: `orbitalvelocity.m`
  - Function and script M-Files
  - Program flow control: `if` and `switch`
  - Control Window input and output

*Follow instructions in m-files …*

# More Useful Functions

- Workspace

```
>> clear
>> who, whos
```

- Search path

```
>> path
>> addpath
```

- File operations

```
>> ls, dir
>> cd
>> copyfile
>> pwd
>> mkdir
```

# More Programming

- MATLAB

  + External Interfaces

    + Importing and Exporting Data

      + Using MAT Files

    + Calling C and Fortran Programs

    + Creating C Language MEX-Files

    + Calling Java from MATLAB

      + Bringing Java classes and methods

- MATLAB Compiler