Matlab Tutorial - II

Based on Matlab tutorial by IPLab®SUT

What we did last week
- Basic math
- Environment
- Matrix and array operations
- Logical operations
- Strings, cells and structures

What we’ll do today
- Scripts
- Functions
- Control flow
- Debugging
- I/O
- Efficiency tips

Programming/Debugging

Script and Function Files
- **Script Files**
  - Work as though you typed commands into MATLAB prompt
  - Variable are stored in MATLAB workspace
- **Function Files**
  - Let you make your own MATLAB Functions
  - All variables within a function are local
  - All information must be passed to functions as parameters
  - Subfunctions are supported

Basic Parts of a Function M-File
Flow Control Statements

*if* Statement
```
if ([attendance > 0.90] & (grade_average > 60))
    pass = 1;
end
```

*while* Loops
```
eps = 1;
while ([eps*1] > 1)
    eps = eps/2;
end
eps = eps*2
```

Flow Control Statements for Loop
```
x = cos(x); % Preallocate matrix
for = 1:i
    s(i, x) = 1/(x-1);
end
switch Statement
method = 'Wilkinson';
switch (method)
    case 'Wilkinson'
        disp('Method is Wilkinson')
    case 'Golub'
        disp('Method is Golub')
end
end
end
end
end
```

M-file Programming Features

- SubFunctions
- Varying number of input/output arguments
- *Local* and *Global* Variables
- Obtaining User Input
- Prompting for Keyboard Input
- Pausing During Execution
- Errors and Warnings
- Displaying error and warning Messages
- Shell Escape Functions ([! Operator])
- Optimizing MATLAB Code
- Variable-length inputs
- Preallocating Arrays

Function M-file
```
function x = ourenn(k, x)
    n = rank(x);
    if (nargin > 1)
        tol = max(x) * eps;
    end
    x = sum(x > tol);
end
```
```
function [max, stderr] = ourenn(k, x)
    if k > 1
        w = w/2;
    end
    max = sum(x)/w;
    stderr = sqrt(sum(x.^2)/w - mean(x)^2);
end
```

Editing and Debugging M-Files

- What is an M-File?
- The Editor/Debugger
- Search Path
- Debugging M-Files
  - Types of Errors (Syntax Error and Runtime Error)
  - Using *keyboard* and ';' statement
  - Setting Breakpoints
  - Stepping Through
    - Continue, Go Until Cursor, Step, Step In, Step Out
  - Examining Values
    - Selecting the workspace
    - Viewing *Datatips* in the Editor/Debugger
    - Evaluating a Selection

Debugging
Input/Output

Importing and Exporting Data

- Using the Import Wizard
- Using **Save** and **Load** command

```
save frame 1
save frame x y z
save frame x y z
load frame x
load frame x y z
load frame x y z
load frame -mat
```

Input/Output for Text File

- Read formatted data, reusing the format string N times.
  ```
  >> [A1, A2] = textread('filename', 'format', N)
  ```

- Import and Exporting Numeric Data with General ASCII delimited files
  ```
  >> X = csvread('filename', delimiter, range)
  ```

Graphics Fundamentals

Graphics

- Basic Plotting
  ```
  plot(t, x); title('title'); xlabel('x label'); ylabel('y label'); grid
  ```
- Editing Plots
  ```
  Property Editor
  ```
- Mesh and Surface Plots
  ```
  meshgrid, mesh, surf, colorbar, patch, hidden
  ```
- Handle Graphics

2-D Plotting

Syntax:

```
plot(x1, y1, 'color', x2, y2, 'color', ...)
```
Sample Plot

Subplots
Syntax: subplot(rows, cols, index)

Tips and tricks

Surface Plot Example

3-D Surface Plotting

Specialized Plotting Routines

bar-bar3-hist-area-pie3-rose
Efficiency

- Whenever possible, use Matlab inbuilt functions
  - `sum`, `.*`, `repmat`
- Pre-allocate space
- Vectorize code
- Profile your code
  - `profile on; <your_code>; profile report`

See [research.microsoft.com/~minka/software/matlab.html](http://research.microsoft.com/~minka/software/matlab.html) for more tricks