MATLAB for complete novices

Roland Memisevic

January 18, 2007
Why MATLAB?

- Easy to learn.
- Many useful features ('toolboxes').
- Standard!
  - Used all over by engineers, scientists, etc.
  - Useful to know even if you don’t want to use it... (see e.g. Octave, Python’s pylab, etc.)
- Excellent documentation.
  - Can use MATLAB to learn some math itself!
Caveats

- Has some disadvantages, too:
  - Not a very 'modern' programming language.
  - Can be awkward.
  - Not good for large software projects.
  - Proprietary.
Starting MATLAB...

- At the command prompt type:
  ```
  matlab
  ```
- Or, if you don’t like windows:
  ```
  matlab -nodesktop
  ```
- Get help any time with
  ```
  help 'function-name'
  ```
- To exit:
  ```
  exit
  ```
Working with MATLAB

- The MATLAB-prompt behaves in many ways like a standard UNIX-prompt.
  - Navigate with cursor, TAB-completion, etc.
- MATLAB can be (and usually is) used interactively!
- MATLAB is verbose: Shows results immediately
- You can suppress this by ending the line with ';'
Operations

- To use MATLAB, enter stuff at the command prompt:
  \[ 5 \times 7 \]

- Some simple operations:
  \[ +, -, *, ^, /, ==, <, >, \sim = \]

- To use variables, assign to them:
  \[ x = 5 \]
  \[ y = 234 \]
  \[ x \times y \]

- Some simple functions:
  \[ \text{sin, cos, exp, log, sqrt, \ldots} \]
Matrices

- To enter matrices use [ , ]
- Separate columns with ',' or ' '
- Separate lines with ';
- $A = [1 \ 2 \ 3; \ 4 \ 5 \ 6]$ yields
  
  $A = [1 \ 2 \ 3$
  
  $4 \ 3 \ 6]$

- Important shortcut is ': : ' Works like this:
  
  $1 : 0.5 : 3 \ \text{gives you} \ \ [1.0 \ 1.5 \ 2.0 \ 2.5 \ 3.0]$

- Access matrix elements with ( ). For example
  
  $A(2,2) = 3$

- Note: Indexes start at one!
Working with vectors and matrices

- Most functions mentioned before are performed *element-wise*. Two exceptions are
  * and ^

  To make these element-wise, use 'dot-notation':
  .* and .^

- You can summarize vectors (and matrices) with
  min, max, mean, sum, ...

  For example: \( \text{min}([3, 2, 4, 5, 6]) = 2 \)
Matrix algebra

- Standard matrix algebra rules apply. E.g.
  \[ [1,2]'*[1,2] = ? \]
  \[ [1,2]*[1,2]' = ? \]

- To transpose use ’
Working with vectors and matrices

- Special functions for quickly building big matrices:
  - zeros, ones, rand, randn, eye

- Work like this:
  - To get a $3 \times 3$-matrix filled with zeros, type
    ```matlab
    zeros(3)
    ```
  - To get a $3 \times 1$-matrix filled with zeros, type
    ```matlab
    zeros(3,1)
    ```
  - Etc.
  - The other functions similarly.
Scripts and functions

- Can write *scripts* by stacking commands in a file ending in `.m`
- Similarly, define *functions* by starting the file with
  ```matlab
  function [y] = myfunction(x)
  
  The value of y will be the return value. The name of the file will be the function name.
  ```
- Comments start with `%`
- Example:
  ```matlab
  function [y] = timestwo(x)
  y = 2*x  % multiply by two...
  ```
for, while, if ...

- **for-loops**
  - ```
    for i = 1 : 0.5 : 5
       exp(i)
    end
  ```

- **while-loops**
  - ```
    i = 1.1
    while i<=2
       i = i^2
    end
  ```

- **conditionals**
  - ```
    i = sin(2.1374)
    if i < 0.5
       i = i^2
    end
  ```
Plotting

- To plot use 'plot'.
- For example
  \[ x = 1 : 0.5 : 10; \]
  \[ y = \sin(x); \]
  \[ \text{plot}(x,y) \]

- You can use an additional string argument. One example:
  \[ \text{plot}(x,y, 'r--') \]

- Use 'help plot' for more on this.
- Overlay plots using 'hold on/off'
More plotting

- Change labeling with 'xlabel', 'ylabel', 'title'.
- Generate subplots with 'subplot'.
- Display matrices with 'imagesc'.
- E.g.

  ```
  A = rand(10)
  subplot(1,2,1)
  imagesc(A)
  B = (1:0.1:10)'*(1:0.1:10)
  subplot(1,2,2)
  imagesc(B)
  ```

- Other: 'plot3', 'scatter', 'bar', 'hist', ...
Slicing and logical indexing

- Can refer to *slices* of matrices using ‘:’
- Example: Let \( a = \text{eye}(3); \)
  \[
a(1,:) = [1, 0, 0] \text{ and } a(2,:) = [0, 1, 0], \text{ etc.}
\]
- You can use *logical matrices* to access elements of other matrices.
- \( \text{==}, <, >, \) etc. actually return logical matrices (they work component-wise).
- So, if \( a = [1, 2, 3] \) you have:
  - \( a(a > 1) = [2, 3] \)