Digital Image Processing

CS  4650/7650
ECE 4655/7655

Lecture 3. Matlab Tutorial

Tony X. Han

Electrical and Computer Engineering
University of Missouri
Content

- Introduction to Matlab
- Working with Matlab
- Image Processing using Matlab
- Class Project
Introduction

- MatLab: **Matrix Laboratory**
- Numerical Computations with matrices
  - *Every number can be represented as matrix*
- Why Matlab?
  - User Friendly (GUI)
  - Easy to work with
  - Powerful tools for complex mathematics
 Snapshot of Matlab

Workspace

Command History

Command Window

>>
Matrices in Matlab

- To enter a matrix

\[
\begin{bmatrix}
3 & 1 \\
6 & 4
\end{bmatrix}
\]

```matlab
>> A = [3 1 ; 6 4]
```

```matlab
>> A = [3, 1 ; 6, 4]
```

```matlab
>> B = [3, 5 ; 0, 2]
```

*Matlab demo*

*Mat_tut.m*
Basic Mathematical Operations

Addition:

>> C = A + B

Subtraction:

>> D = A – B

Multiplication:

>> E = A * B (Matrix multiplication)
>> E = A .* B (Element wise multiplication)

Division:

Left Division and Right Division

>> F = A . / B (Element wise division)
>> F = A / B (A * (inverse of B))
>> F = A . \ B (Element wise division)
>> F = A \ B ((inverse of A) * B)
Generating basic matrices

Matrix with ZEROS:
>> Z = ZEROS (r, c)

Matrix with ONES:
>> O = ONES (r, c)

IDENTITY Matrix:
>> I = EYE (r, c)

r → Rows

c → Columns

zeros, ones, eye → Matlab functions
Working with complex numbers

- real and imaginary
  - `real` % real part of complex number
  - `imag` % imaginary part of complex number

- magnitude and phase
  - `abs` % magnitude of complex number
  - `angle` % phase of complex number

Matlab demo
Mat_tut.m
Making the best from Matlab

- Need help?
  - HELP <function name>
  - LOOKFOR <key word>

- Case sensitive

- M files (.m)
  - To write and save matlab commands
  - Save time and easy to debug
  - Use of semicolon (;)
  - Comments (%), Block Comments (%{, %})

- Online tutorial and documentation
  - [www.mathworks.com](http://www.mathworks.com)
Image processing with Matlab

To read and display images

- \( \text{im} = \text{imread("filename.fmt")} \)
  - \( \text{im} \) is \( r \times c \) if gray scale
  - \( \text{im} \) is \( r \times c \times 3 \) if color image (RGB)

- \( \text{imshow}(\text{im}) \) % displays image

- \( \text{imwrite}(\text{im}, "filename.fmt") \) % writes image

Size of image

- \([r \ c] = \text{size(}\text{im})\)

Matlab demo
Mat_tut.m
Plotting / displaying

- **PLOT(x,y)**
  - Plots \( y \) versus \( x \).
  - Linear plot
  - XLABEL('label')
  - YLABEL('label')
  - TITLE('title')

- **imshow(x)**
  - Displays image
  - `>>impixelinfo` %creates a Pixel Information tool in the current figure.
  - `>>imdistline` %generates a draggable Distance tool.
  - `>>imdisplayrange` %creates a display range tool in the current figure.

- **3D plots:**
  - **MESH**
    - 3D mesh surface (Ex. filters)
  - **MESHGRID**
    - Useful in 3D plots
  - **SURF**
    - 3D colored surface (Ex. filters)
Random Number generators
  - `rand`, `randn`

Image Processing Functions
  - `fft2`, `ifft2`, `fftshift`, `ifftshift`, `filter2`, `imfilter` etc...

Writing Functions
Random Number Generators

rand / randn:

>>rand(m,n);
  ☐ Generates uniformly distributed random number of size $m \times n$
  ☐ Between (0.0 to 1.0)

>>randn(m,n);
  ☐ Generates normally distributed random number of size $m \times n$
  ☐ *Mean* = 0, *Variance* = 1, and *SD* = 1
Image Transforms

Fourier Transforms:

fft2 / ifft2

- **FFT2**: Performs 2-D discrete Fourier Transform
- **IFFT2**: Performs Inverse 2-D discrete Fourier Transform

Matlab demo: test_fft.m
Frequency Shifts:

fftshift / ifftshift

- **FFTSHIFT**: Shifts zero-frequency component to center of spectrum
- **IFFTSHIFT**: Inverses the fftshift

Examples during Matlab demo

Discrete Cosine Transform:

dct2 / idct2

- **DCT2**: Performs 2-D discrete Cosine Transform
- **IDCT2**: Performs Inverse 2-D discrete Cosine Transform
Image Filters

filter2 / imfilter:

```matlab
>> filter2(h, X, 'shape');
```

- Performs 2-D digital filtering on X with 2-D filter h
- SHAPE: 'same', 'valid', 'full'

```matlab
>> imfilter(X, h, 'Option1', 'Option2'...);
```

- Performs 2-D digital filtering on X with 2-D filter h
- OPTIONS: Boundary Options ; Output Size Options ; Correlation / Convolution

fspecial

- To generate 2-D special filters (h)
- TYPES: Average, Gaussian, etc...

Examples during Matlab demo

Original Image

Average Filtered
Functions in Matlab

\[ \text{function return values = function\_name (arguments)} \]

- Begins with key word \textit{function}
- First line of the file
- Save it as “function\_name.m”
- No semicolon (;)

\begin{itemize}
  \item Example:
  \begin{verbatim}
  function showGrayIm (im)
  \end{verbatim}
  \begin{verbatim}
  \ \ % show 8bits/pixel gray scale image
  \ \ % Zhou Wang, Sept. 2006
  \ \ colormap(gray(256));
  \ \ image(im);
  \ \ axis('image', 'off');
  \end{verbatim}
\end{itemize}

- \texttt{gray(256)}: Generates a 256 x 3 matrix containing \textit{grayscale} color map
- \texttt{colormap(gray(256))}: Transforms / changes the \textit{current} image color map to new color map generated by \texttt{gray(256)}
Write Fast Matlab code

- Find the hardcore of your code using the profiling tools.

- Matlab is NOT slow if you write it right!!

Matlab Profiler
Matlab Profiler (cont.)

![Profiler Window](image)

**Profile Summary**

Generated 11-Sep-2012 17:05:53 using cpu time.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Calls</th>
<th>Total Time</th>
<th>Self Time*</th>
<th>Total Time Plot (dark band = self time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_lap</td>
<td>1</td>
<td>1.604 s</td>
<td>0.607 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>newplot</td>
<td>6</td>
<td>0.314 s</td>
<td>0.012 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imshow</td>
<td>1</td>
<td>0.300 s</td>
<td>0.021 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imagesc</td>
<td>3</td>
<td>0.266 s</td>
<td>0.019 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imread</td>
<td>1</td>
<td>0.231 s</td>
<td>0.013 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>graphics/private/clo</td>
<td>7</td>
<td>0.216 s</td>
<td>0.051 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imagesci/private/imftype</td>
<td>1</td>
<td>0.205 s</td>
<td>0.002 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imagesci/private/istif</td>
<td>1</td>
<td>0.201 s</td>
<td>0.201 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>newplot&gt;ObserveFigureNextPlot</td>
<td>6</td>
<td>0.169 s</td>
<td>0.003 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>clf</td>
<td>1</td>
<td>0.166 s</td>
<td>0.039 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>imuertools/private/initSize</td>
<td>1</td>
<td>0.137 s</td>
<td>0.016 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>newplot&gt;ObserveAxesNextPlot</td>
<td>6</td>
<td>0.131 s</td>
<td>0.001 s</td>
<td>[Bar Graph]</td>
</tr>
<tr>
<td>cla</td>
<td>6</td>
<td>0.130 s</td>
<td>0.004 s</td>
<td>[Bar Graph]</td>
</tr>
</tbody>
</table>
Some tips for writing fast code

- Vectorizing Loops

   Matlab demo: VectorizingLoops.m
Code Analyzer Report

%Basic Mathematical Operations
Code Analyzer Report (cont.)

VectorizingLoops 6 messages
6: The value assigned to variable 'y' might be unused.
6: The variable 'y' appears to change size on every loop iteration (within a script).
Consider preallocating for speed.

mat_tut 23 messages
3.1 Terminate statement with semicolon to suppress output (within a script).
4.1 Terminate statement with semicolon to suppress output (within a script).
9.1 Terminate statement with semicolon to suppress output (within a script).
10: Terminate statement with semicolon to suppress output (within a script).
11: The value assigned to variable 'Y' might be unused.
11: Terminate statement with semicolon to suppress output (within a script).
12: Terminate statement with semicolon to suppress output (within a script).
13: The value assigned to variable 'Y' might be unused.
13: Terminate statement with semicolon to suppress output (within a script).
14: The value assigned to variable 'Y' might be unused.
14: Terminate statement with semicolon to suppress output (within a script).
15: Terminate statement with semicolon to suppress output (within a script).
20: Terminate statement with semicolon to suppress output (within a script).
21: Terminate statement with semicolon to suppress output (within a script).
22: Terminate statement with semicolon to suppress output (within a script).
27: Terminate statement with semicolon to suppress output (within a script).
27: Replace complex i and j by li for speed and improved robustness.
27: Replace complex i and j by li for speed and improved robustness.
27: Replace complex i and j by li for speed and improved robustness.
27: Replace complex i and j by li for speed and improved robustness.
3.1 Terminate statement with semicolon to suppress output (within a script).
3.1 Terminate statement with semicolon to suppress output (within a script).
3.1 Terminate statement with semicolon to suppress output (within a script).
3.1 Terminate statement with semicolon to suppress output (within a script).

Test About Memory 1 message
1: The variable 'x' appears to change size on every loop iteration (within a script).
Consider preallocating for speed.

Test fft 2 messages
56: Replace complex i and j by li for speed and improved robustness.
127: It appears that flipud(fliplr(x)) or fliplr(flipud(x)) can be replaced by a faster rot90(x).

Test_lap 1 message
9: Terminate statement with semicolon to suppress output (within a script).