A Practical Guide to Matlab

Study Session 9/24/2009

Linjie Luo

Outline

- Matlab program structure
- Read, write & show images
- All about matrix
- Flow control
- Useful functions
- Debugging
- Where to get help
- Q & A

Matlab Program Structure

- Command line (no source files)
- Two types of M-File
  - Scripts:
    - Global variable access
    - Start of the control flow
  - Functions:
    - Local variable access
    - Start with “function” keyword with arguments & return values
- Functions in working folder are automatically recognized. No #include
- NOTE: function name MUST be its M-File name!

Matlab Program Structure

main.m

```
'Compute factorial of n'

n = input('Input n:');

factorial(n)
```

factorial.m

```
function [ retval ] = factorial( n )
if n == 1
   retval = 1;
else
   retval = n * factorial(n-1);
end
end
```

output

```
Compute factorial of n
Input n:10
ans =
     3628800
```

Read Image & Preprocessing

- Function for reading image
  - img = imread('filename.jpg')
  - img: <WIDTH x HEIGHT x 3 uint8> matrix
  - uint8: unsigned 8-bit integer
- Convert to grayscale image:
  - gray = rgb2gray(img)
- Normalize to [0,1]:
  - normalized = double(img) / 255

Write Image

- Function for writing Image
  - imwrite(img, ‘filename.jpg’)
- Use only the following matrix format for img
  - <WIDTH x HEIGHT x 3 uint8>
  - <WIDTH x HEIGHT x 3 double>
  - <WIDTH x HEIGHT uint8>
  - <WIDTH x HEIGHT double>
- uint8 in [0, 255]; double in [0, 1]

Show Image

- Function for showing image
  - imshow(img)
- Show multiple images
  - figure
  - imshow(img1)
  - figure
  - imshow(img2)

Matrix - Creation

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 6 & 8 \\ 5 & 7 & 9 \end{pmatrix}$$

- [, ;] operator
  - M = [1, 2, 3; 4, 6, 8; 5, 7, 9]
  - M = [1 2 3; 4 6 8; 5 7 9]
- : operator
  - START : STEP (1 by default) : END
  - M = [1:3; 4:2:8; 5:2:9]
- zeros
  - M = zeros(3, 3)
  - M = zeros(3, 3, ‘uint8’)
- rand
- magic

Matrix - Access

$$M=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 6 & 8 \\ 5 & 7 & 9 \end{pmatrix}$$

- size funtion
  - size(M) = [3, 3]
  - size(M, 1) = 3; size(M, 2) = 3
- () operator
  - M(2, 3) = 8
  - Index start from 1 not 0!
- : operator, end keyword
  - M(1:2, 2:3) = [2, 3; 6, 8]
  - M(2, :) = [4, 6, 8]
  - M(1:end, 2:end) = [2, 3; 6, 8; 7, 9]
  - M(2, 1:end-1) = [4, 6]
  - M(1:2:3, 1:2:3) = [1, 3; 5, 9]
- Using matrix to index matrix
  - M([1, 3], [1, 3]) = [1, 3; 5, 9]

Matrix - Arithmetic

- Arithmetic operators
  - A + B → A + B
  - A – B → A – B
  - A * B → AB
  - A \ B → $A^{-1}B$ (A needs to be invertible)
  - A / B → AB^{-1} (B needs to be invertible)
  - A ^ n → A^n
  - A ’ → A^* (conjugate transpose)
  - A .* B → element-by-element multiplication
  - A ./ B → element-by-element division
  - A .\ B → element-by-element left division
  - A .^ B → element-by-element power
  - A .’ → A^T (transpose)

Matrix - Arithmetic

- Arithmetic functions
  - inv(A) → A^{-1}
  - det(A) → det(A)
  - rank(A) → rank(A)
  - eig(A) → eigenvalues
  - sqrt(A) → element-by-element square root
  - abs(A) → element-by-element absolute value
  - sin(A) → element-by-element sine
  - …
- Refer the Matlab help for more functions

Flow Control - if

- "if" statement
  - `if` *condition1*

        statement1

    `elseif` *condition2*

        statement2

    `else` *condition3*

        statement3

    end
- Relational operators for conditions

```
< > <= >= == ~=
```

Flow Control - for

- "for" statement
  - for *variable = start[:step]:end*

    statement

    end
- : operator
  - step is 1 by default, similar to the usage in matrix indexing

Flow Control - while

- "while" statement
  - `while` *condition*

    statement

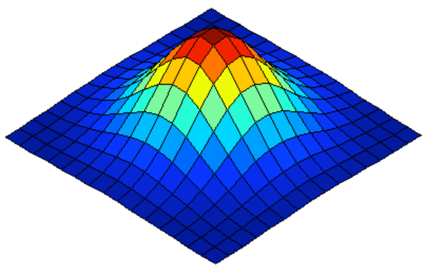
    `end`

Flow Control – break, continue

- "break" statement
  - for / while …
    ```
    for / while …
          break
    end
    ```
- "continue" statement
  - ```
    for / while …
          continue
    end
    ```
- Effective only for the innermost loop (the same with C/C++)
    ```

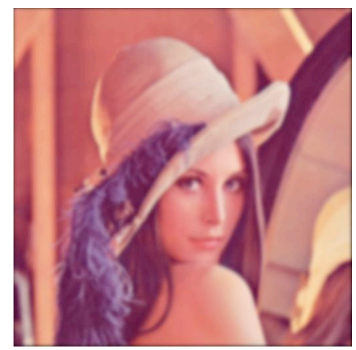
Useful Functions – conv2

- 2D convolution: $c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2)\, b(n_1 - k_1, n_2 - k_2)$
  - Gaussian filtering



$*$

$=$

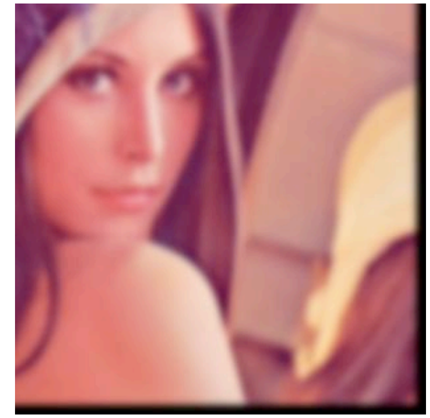


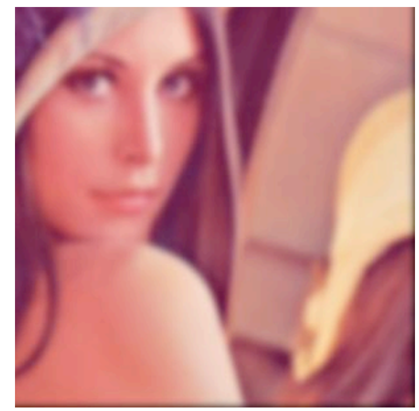
G I Ib

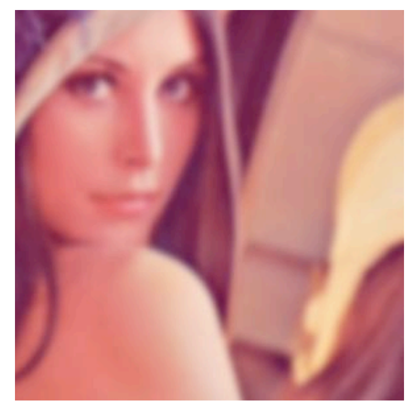
  - Ib = conv2(I, G)

Useful Functions – conv2

- Shape parameter
  - Ib = conv2(I, G, 'full')
    size(Ib)=$[H_I+H_G-1, W_I+W_G-1]$
  - Ib = conv2(I, G, 'same')
    size(Ib)=$[H_I, W_I]$
  - Ib = conv2(I, G, 'valid')
    size(Ib)=$[H_I-H_G+1, W_I-W_G+1]$







Useful Functions – conv2

- Column-wise & Row-wise 1D convolution
  - Differentiation

$$F_x' = \frac{\partial F}{\partial x} = \frac{F(x+\Delta h, y) - F(x-\Delta h, y)}{2\Delta h} + O(\Delta h^2)$$

$$F_y' = \frac{\partial F}{\partial y} = \frac{F(x, y+\Delta h) - F(x, y-\Delta h)}{2\Delta h} + O(\Delta h^2)$$

  - Fx = conv2(1, [1 0 -1], F)
  - Fy = conv2([1 0 -1], 1, F)

Useful Functions - eig

- Eigenvalue decomposition
  - $M = VDV^{-1}$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} -0.8246 & -0.4160 \\ 0.5658 & -0.9094 \end{pmatrix} \begin{pmatrix} -0.3723 & 0 \\ 0 & 5.3723 \end{pmatrix} \begin{pmatrix} -0.8246 & -0.4160 \\ 0.5658 & -0.9094 \end{pmatrix}^{-1}$$

  - Principal Component Analysis (PCA)
  - Useful for corner detection
- Usage
  - [V, D] = eig(M)

Useful Functions - sortrows

- Sort matrix rows
  - minus sign indicates descending order
  - sortrows(M, 1) $\begin{pmatrix} 5 & 7 & 3 \\ 1 & 7 & 4 \\ 1 & 2 & 8 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 7 & 4 \\ 1 & 2 & 8 \\ 5 & 7 & 3 \end{pmatrix}$
  - sortrows(M, [1, 2]) $\begin{pmatrix} 5 & 7 & 3 \\ 1 & 7 & 4 \\ 1 & 2 & 8 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 2 & 8 \\ 1 & 7 & 4 \\ 5 & 7 & 3 \end{pmatrix}$
  - sortrows(M, [2, -1]) $\begin{pmatrix} 5 & 7 & 3 \\ 1 & 7 & 4 \\ 1 & 2 & 8 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 2 & 8 \\ 5 & 7 & 3 \\ 1 & 7 & 4 \end{pmatrix}$

Useful Functions - find

- Return nonzero element indices of an array
  - For example:

    A = [0, 12, 13, 0, 15, 17, 18, 0, 19]

    find(A) = [2, 3, 5, 6, 7, 9]

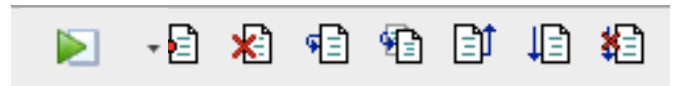
    A(find(A)) = [12, 13, 15, 17, 18, 19]
  - Essentially remove all zero elements
  - Also:

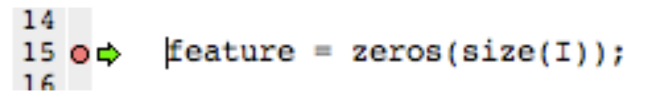
    find(A>15) = [6, 7, 9]

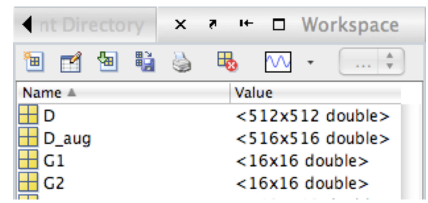
    A(find(A>15)) = [17, 18, 19]

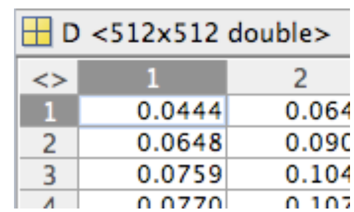
Debugging

- Use debug panel in editor
  - Set a breakpoint
  - Run the code
  - See what is going wrong
- Workspace window
  - You can view all the matrices not too large
- Command window
  - Immediate query into the current data
- Cancel output suppression
  - Don't put a semicolon ";" behind the expression
  - The value of expression will display in command window after evaluation









Where to get help

- Matlab product help
  - MATLAB → Getting started
  - Search box
- Online Matlab forum
  - http://www.mathworks.com/matlabcentral/newsreader/
- Google
- Email me

Questions?