Getting Started with MATLAB

CS534 TA: Chunhui Zhu
czhu@cs.wisc.edu
Sep 16th, 2011

Thanks to the help from Tuo Wang and Prof. Dyer in making these slides.

Outline

- **Introduction to MATLAB**
  - **Basics & Examples**
- Image Processing with MATLAB
  - Basics & Examples

What is MATLAB?

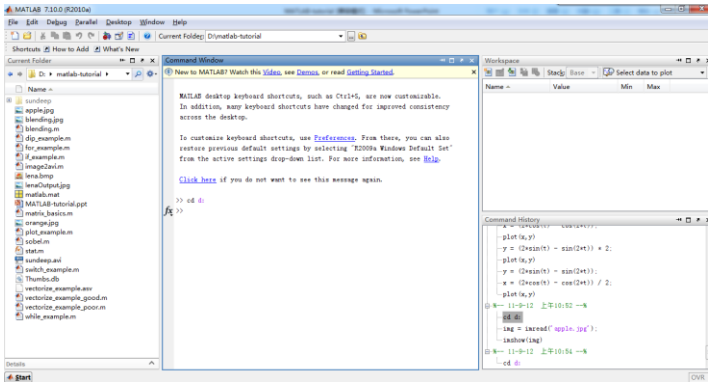
- MATLAB = Matrix Laboratory
- "MATLAB is an interactive, matrix-based system for scientific and engineering numeric computation and visualization. You can solve complex numerical problems in a fraction of the time required with a programming Language such as Fortran or C."

  ---- Matlab Primer

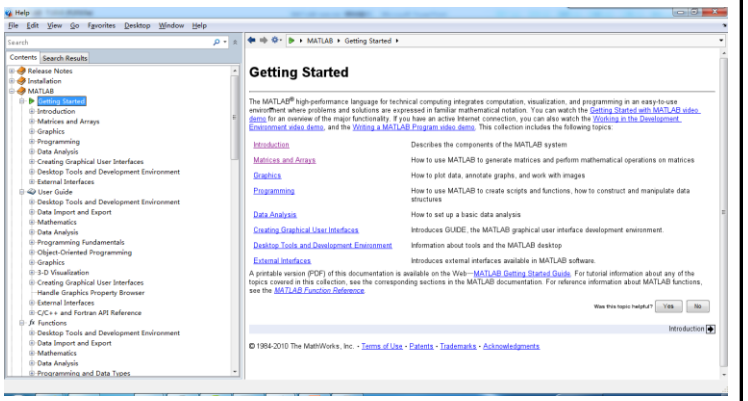
MATLAB vs. JAVA&C++

- Task: compute the eigenvectors and eigenvalues for matrix A
- JAVA/C++: OK, let me first define a matrix struct or class, then look up the eigenvalue definition in linear algebra book, figure out the solution, write the code and debug…Hopefully, get the right answer…
- MATLAB: One command line.
- MATLAB: quick implementation of your idea JAVA&C++: for development

The MATLAB Environment



MATLAB Help



MATLAB Help (cont.)

Search by keywords

```
>> help sqrt
SQRT   Square root.
   SQRT(X) is the square root of the elements of X. Complex
   results are produced if X is not positive.

   See also sqrtm, realsqrt, hypot.

   Overloaded methods:
      codistributed/sqrt

   Reference page in Help browser
      doc sqrt
```

Type in command window:
help functionname(eg. help sqrt)

Entering Matrices

- By an explicit list of elements
  - `A=[1,2,3; 4 5 6; 7 8, 9];`
- By built-in statements and functions:

  `Evenly distributed vector:`

  `0:0.5:2 generates[0,0.5,1.5,2]`

  `zeros(n,m), ones(n,m), eye(n,m), rand(), randn(),repmat()`
- Load from external data files or applications

Basic Operations on Matrices

- All operators in MATLAB are defined on matrices: `+, -, *, /, ^, sqrt, sin, cos`, etc.
- Element-wise operators defined with a preceding dot: `.*, ./, .^`
- Be sure about whether matrix operations or element-wise operators are using!

Logical Operators

- `==, <, >,` (not equal) ~=, (not) ~
- `find('condition')` – Returns indexes of A's elements that satisfy the condition

Logical Operators (cont.)

- Example:

```
>>A=[7 3 5; 6 2 1], [r,c] = find(A < 4)
  A=
    7 3 5
    6 2 1
  r =  1          c = 2
       2              2
       2              3
```

Some Built-in Matrix Funcs

- `[eigVec,eigVal]=eig(A), eigenvectors and eigenvalues for matrix A`
- `inv(A),inverse of matrix A`
- `det(A),determinant of matrix A`
- `rank(A),the rank of matrix A`
- `size(A,1),number of rows of Matrix A; size(A,2), number of columns`
- …

Variable Name in Matlab

- Variable naming rules

  - must be unique in the first 63 characters
  - must begin with a letter
  - may not contain blank spaces or other types of punctuation
  - may contain any combination of letters, digits, and underscores
  - are case-sensitive
  - should not use Matlab keyword
- Pre-defined variable names
  - pi

Scripts and Functions

- There are two kinds of M-files(.m):
  - Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace. When executed, it equals to orderly type the command lines in the command window.
  - Functions, which can accept input arguments and return output arguments. Internal variables are local to the function

Functions in MATLAB (cont.)

- Example:
  - A file called stat.m:

```
function [mean, stdev]=stat(x)
%STAT Interesting statistics.
n=length(x);
mean=sum(x)/n;
stdev=sqrt(sum((x-mean).^2)/n);
```

  - Defines a new function called stat that calculates the mean and standard deviation of a vector. Function name and file name should be the SAME!
  - CODE

Suggested Project Organization

- A script file + Several self-defined function files
- Script file acting like a main.cpp in C++, it calls the self-defined functions or built-in system functions.
- Self-defined function files implement details of algorithms.
- DEMO

Flow Control

- MATLAB has five flow control constructs:
  - `if` statement
  - `switch` statement
  - `for` loop
  - `while` loop
  - `break` statement

if

- IF statement condition
  - The general form of the IF statement is

```
IF expression
   statements
ELSEIF expression
   statements
ELSE
   statements
END
```

- CODE

switch

- SWITCH – Switch among several cases based on expression
- The general form of SWITCH statement is:

```
SWITCH switch_expr
   CASE case_expr,
      statement, …, statement
   CASE {case_expr1, case_expr2, case_expr3, …}
      statement, …, statement
      …
   OTHERWISE
      statement, …, statement
END
```

switch (cont.)

- Note:
  - Only the statements between the matching `CASE` and the next `CASE`, `OTHERWISE`, or `END` are executed
  - Unlike C, the `SWITCH` statement does not fall through (so `BREAK`s are unnecessary)

- CODE

for

- FOR repeats statements a specific number of times
- The general form of a FOR statement is:

```
FOR variable=expr
   statements
END
```

- CODE

while

- WHILE repeats statements an indefinite number of times
- The general form of a WHILE statement is:

```
WHILE expression
   statements
END
```

- CODE

It seems like I can use these loops as I do in C/C++/Java…

Try to AVOID THIS!

Time Cost Comparison

- Loop vs. No Loop

```
A = rand(1000,1000);B = rand(1000,1000);
for i = 1:size(A,1),
   for j = 1:size(A,2),
      C(i,j) = A(i,j) + B(i,j);
   end
end
```

Using loop: Elapsed time is 1.125289 seconds.

Time Cost Comparison(cont.)

- Loop vs. no loop

C = A + B

Elapsed time is 0.002346 seconds.

Visualization and Graphics

- `plot(x,y),plot(x,sin(x))` – plot 1D function
- `figure, figure(k)` – open a new figure
- `hold on, hold off` – refreshing
- `axis([xmin xmax ymin ymax])` – change axes
- `title('figure titile')` – add title to figure
- `mesh(x_ax,y_ax,z_mat)` – view surface
- `contour(z_mat)` – view z as topo map
- `subplot(3,1,2)` – locate several plots in figure

- CODE and Debug CODE

Saving your Work

- `save mysession`
  % creates mysession.mat with all variables
- `save mysession a b`
  % save only variables a and b
- `clear all`
  % clear all variables
- `clear a b`
  % clear variables a and b
- `load mysession`
  % load session

Debug Techniques

- Debug is essential.
- Easy and flexible access during debug.
- DEMO

Outline

- Introduction to MATLAB
  - Basics & Examples
- **Image Processing with MATLAB**
  - **Basics & Examples**

What is the Image Processing Toolbox?

- The Image Processing Toolbox is a collection of functions that extend the capabilities of the MATLAB's numeric computing environment. The toolbox supports a wide range of image processing operations, including:
  - Geometric operations
  - Neighborhood and block operations
  - Linear filtering and filter design
  - Transforms
  - Image analysis and enhancement
  - Binary image operations
  - Region of interest operations

Images in MATLAB

- MATLAB can import/export several image formats:
  - BMP (Microsoft Windows Bitmap)
  - GIF (Graphics Interchange Files)
  - HDF (Hierarchical Data Format)
  - JPEG (Joint Photographic Experts Group)
  - PCX (Paintbrush)
  - PNG (Portable Network Graphics)
  - TIFF (Tagged Image File Format)
  - XWD (X Window Dump)
  - raw-data and other types of image data
- Data types in MATLAB
  - Double (64-bit double-precision floating point)
  - Single (32-bit single-precision floating point)
  - Int32 (32-bit signed integer)
  - Int16 (16-bit signed integer)
  - Int8 (8-bit signed integer)
  - Uint32 (32-bit unsigned integer)
  - Uint16 (16-bit unsigned integer)
  - Uint8 (8-bit unsigned integer)

Images in MATLAB

- Binary images : {0,1}
- Intensity images : [0,1] or `uint8`, `double` etc.
- RGB images : $m \times n \times 3$
- Multidimensional images: $m \times n \times p$ (p is the number of layers)

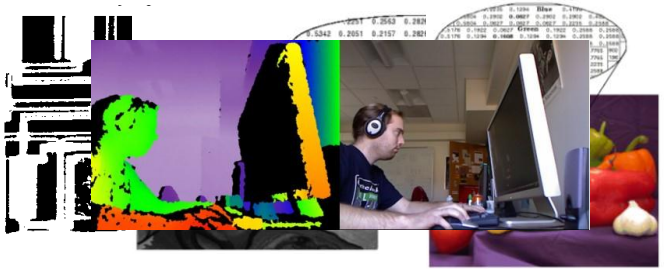


Image Import and Export

- Read and write images in Matlab

```
img = imread('apple.jpg');
dim = size(img);
figure;
imshow(img);
imwrite(img, 'output.bmp', 'bmp');
```

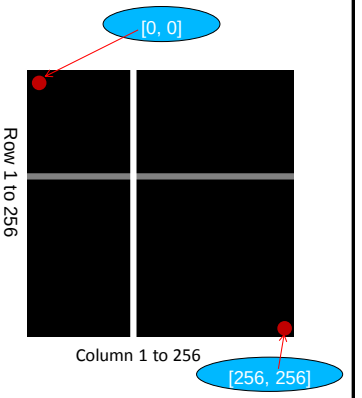
- Alternatives to `imshow`

```
imagesc(I)
imtool(I)
image(I)
```

Images and Matrices

How to build a matrix (or image)?
Intensity Image:

```
row = 256;
col = 256;
img = zeros(row, col);
img(100:105, :) = 0.5;
img(:, 100:105) = 1;
figure;
imshow(img);
```



Images and Matrices

Binary Image:

```
row = 256;
col = 256;
img = rand(row,
col);
img = round(img);
figure;
imshow(img);
```

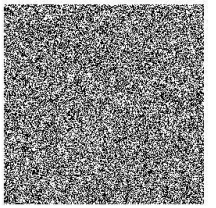


Image Display

- `image` - create and display image object
- `imagesc` - scale and display as image
- `imshow` - display image
- `colorbar` - display colorbar
- `getimage` - get image data from axes
- `truesize` - adjust display size of image
- `zoom` - zoom in and zoom out of 2D plot

Image Conversion

- `gray2ind` - intensity image to index image
- `im2bw` - image to binary
- `im2double` - image to double precision
- `im2uint8` - image to 8-bit unsigned integers
- `im2uint16` - image to 16-bit unsigned integers
- `ind2gray` - indexed image to intensity image
- `mat2gray` - matrix to intensity image
- `rgb2gray` - RGB image to grayscale
- `rgb2ind` - RGB image to indexed image

Image Operations

- RGB image to gray image
- Image resize
- Image crop
- Image rotate
- Image histogram
- Image histogram equalization
- Image DCT/IDCT
- Convolution

- CODE

Outline

- Introduction to MATLAB
  - Basics & Examples
- Image Processing with MATLAB
  - Basics & **Examples**

Examples working with Images (1/3)

Create AVI movie with a series of images

&

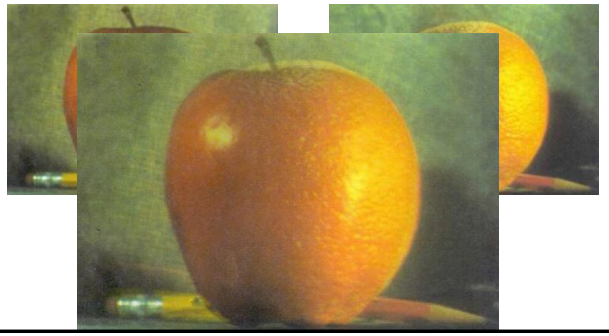
Read specific frame from video file

Related funcs: avifile, addframe, mmreader

DEMO

Examples working with Images (2/3)

Simplified version of image blending:
Each pixel in the apple-orange is:
w(i, j) * Apple(i, j) + (1-w(i, j)) * Orange(i, j)



Examples working with Images (3/3)

Sobel edge detection on image

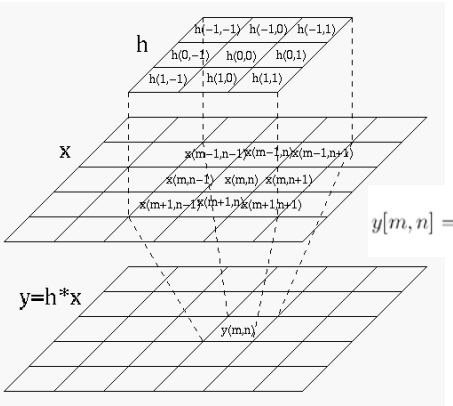
Examples working with Images (3/3)(cont.)

The use of imfilter function:
A great assistant to avoid loop in Matlab image processing

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2} \qquad \mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Examples working with Images (3/3) (cont.)



$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

$$y[m,n] = \sum_{i=-k}^{k} \sum_{j=-k}^{k} x[m+i, n+j] h[i,j]$$

y = imfilter(x,h)

<u>DEMO</u>

Performance Issues

- The idea: MATLAB is
  - very fast on vector and matrix operations
  - Correspondingly slow with loops
- Try to avoid loops
- Try to vectorize your code

  http://www.mathworks.com/support/tech-notes/1100/1109.html

THE END

- Thanks for your attention! ☺
- Questions?