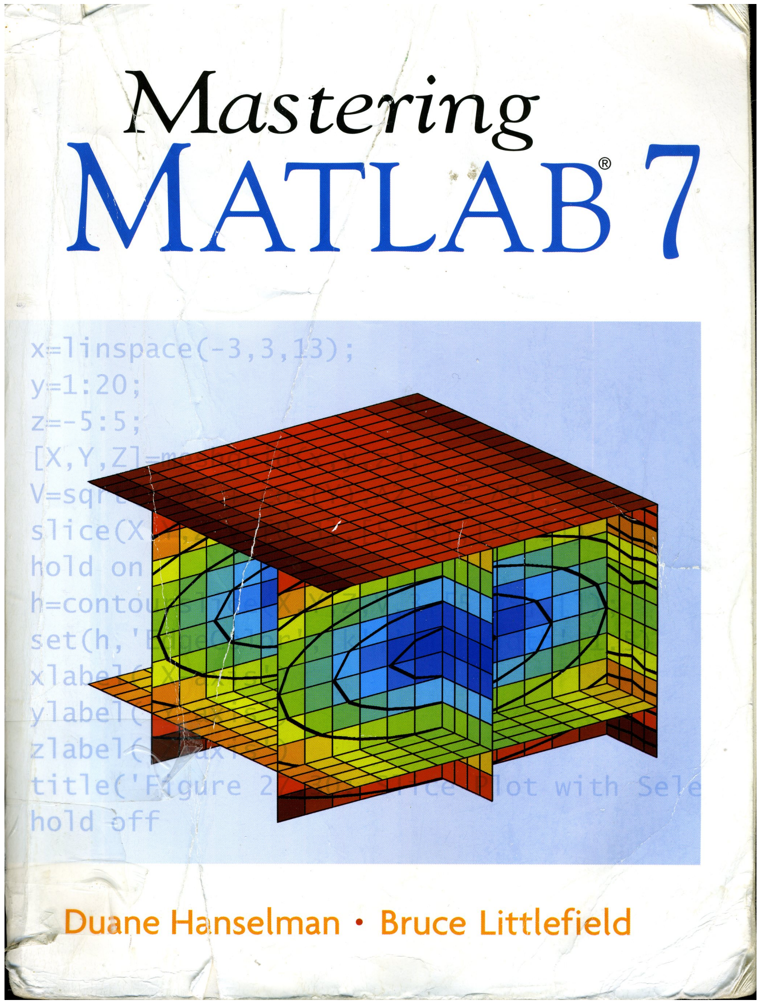
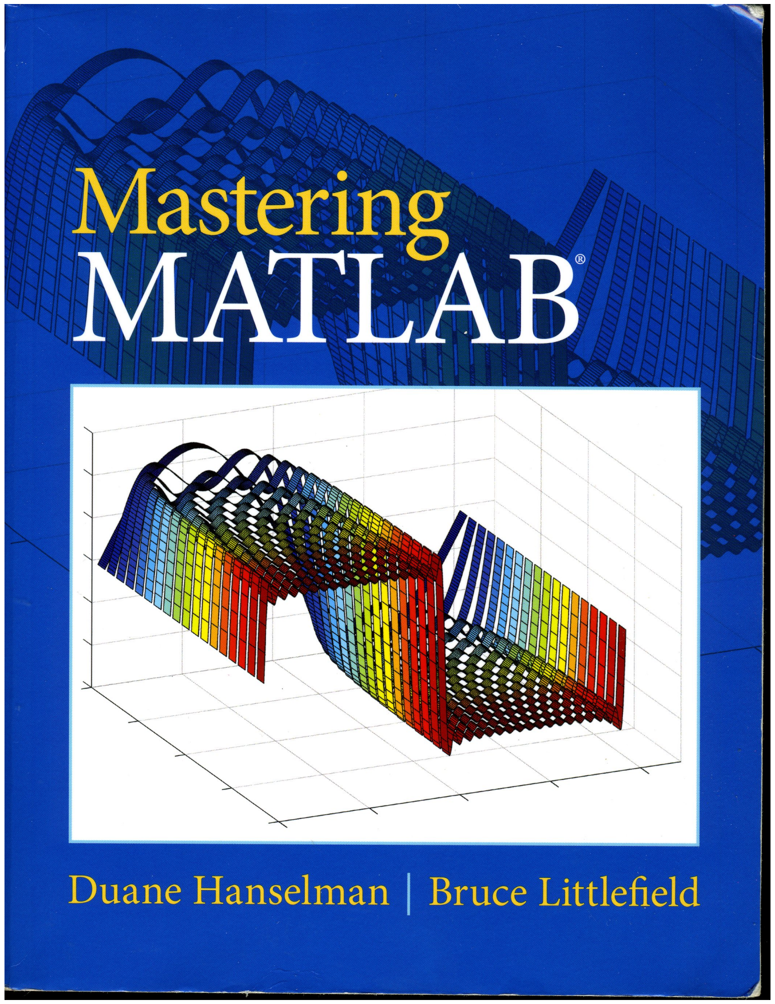
MatLab Textbooks

- Mastering MATLAB, Duane Hanselmann and Bruce Littlefield, Pearson (Prentice Hall), 2012.
- Mastering MATLAB 7, Duane Hanselmann and Bruce Littlefield, Pearson (Prentice Hall), 2005.
- The MatLab notes follow the texts closely in parts.
- The figure on the next slide show the covers of the 2 texts. Naturally, the 2012 edition is the best but it doesn't cover GUIs, whereas the earlier edition does. However, we suggest the use of **GUIDE** for GUI design and implementation as programming GUI directly by typing text is too involved.



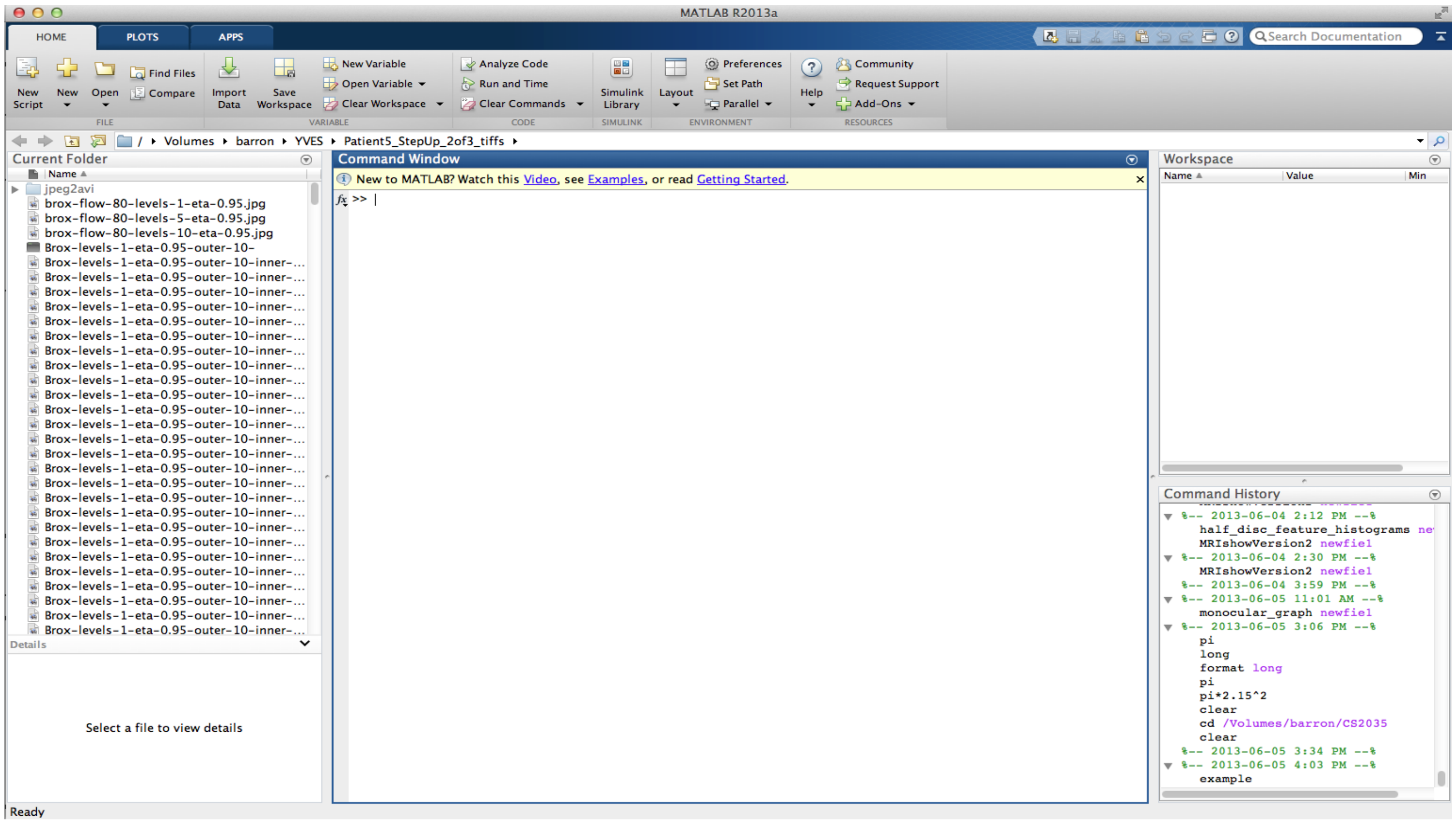
```
x=linspace(-3,3,13);
y=1:20;
z=-5:5;
[X,Y,Z]=m
V=sqr
slice(X
hold on
h=conto
set(h,'
xlabe
ylabel(
zlabel(
title('Figure 2                lot with Sele
hold off
```



MatLab Programming Environment

- MatLab has some basic windows as shown in Figure 1.
- The **MatLab desktop** is where MatLab puts you when you launch it. It has the following subwindows:

  1. **Command window**: This is the main MatLab window (has prompt >>) where application programs are launched from or commands are typed.
  2. **Current directory**: This is where your current files exist.
  3. **Workspace**: Where all the variables you have generated are shown (with their type and size) and where you can plot these variables



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>>
```

```
%-- 2013-06-04 2:12 PM --%
  half_disc_feature_histograms ne
  MRIshowVersion2 newfiel
%-- 2013-06-04 2:30 PM --%
  MRIshowVersion2 newfiel
%-- 2013-06-04 3:59 PM --%
%-- 2013-06-05 11:01 AM --%
  monocular_graph newfiel
%-- 2013-06-05 3:06 PM --%
  pi
  long
  format long
  pi
  pi*2.15^2
  clear
  cd /Volumes/barron/CS2035
  clear
%-- 2013-06-05 3:34 PM --%
%-- 2013-06-05 4:03 PM --%
  example
```

The MatLab environment consists of the MatLab desktop, a figure window and a editor window. The figure and editor windows must be invoked. Note that these windows are for MatLab2012b and later, earlier versions of MatLab have a slightly different feel and touch but are fine for this course.

(click on a variable) and then, using the right mouse, select the appropriate option.

4. **Command history**: All commands typed at a MatLab prompt are recorded and can be re-executed (across multiple sessions) by selecting a command with the mouse and executing it by double clicking on it. You can also create a M-file by selecting a set of commands from this window and or a right click of the mouse to select an option from the menu.

- The **Figure window** is where the graphics results are output. The user can create as many figure windows as memory will allow. When you use MatLab commands like `imshow` (to display images) or `plot` (to display graphs) these images will show in the figure window.

- The **Editor window** is where you write, edit, create and save your MatLab programs. You can use the built-in MatLab editor or use your favourite editor (for example, `vi program.m!` or `emacs program.m!` to invoke the `vi` or `emacs` editors). The built-in editor is simple to use: click on `New` (or `New Script`) and an editor window will pop up. If you used `New` select script to get an empty file. Type your MatLab code, using the `delete` key as necessary. You can point and click to go to specific locations in the file. Click on the `Run` button to execute your code. Look in the command window to see the results. Click on `save` to save your code to a specific filename.

- Commands `lookfor`, `help`, `helpwin` and `hekpdesk` provide online help. Command `demo` provides an introductory tutorial.

A Little Taste of MatLab

- MatLab example:

```
>>                  % MatLab prompt
>> 2+2              % Command
ans =               % MatLab response
    4
>> % pi*r^2 is area of circle of radius r
>> area = pi*2.15^2
area =
  14.5220
```

- A **Variable** is a named location in memory holding some data. Here `area` is the same of a variable and `area = pi*2.15^2` is an **assignment** statement: compute the value of the expression `pi*2.15^2` on the right hand side of the equals symbol and then assign it to the variable `area` on the left hand side of the equals sign.

- The expression `pi*2.15^2` has 2 operators (`*` is multiplication and `^` is exponentiation). Exponentiation has higher **precedence** than multiplication so it is done first. So 2.15^2 is evaluated first (4.6225) and then that is multiplied by `pi` (a built-in MatLab constant equal to π, which is 3.141592653589793 as a double precision number). `area` is a **double** precision number (the default MatLab data-type). A **double** precision number requires 2 words or 8 bytes in the computer to represent it. A **single** precision number, on the other hand, only requires 1 word or 4 bytes of space to represent it in the computer.

- Then the value of $pi * 2.15^2$ is computed as 14.522012041218817 and saved at the memory location for the MatLab variable allocated for `area`.
- Unless you specify otherwise (no semi-colon `;` at the end of the statement), Matlab prints the variable value as a floating point number with 4 digits to the right of the decimal point (this is `format short`).

Basic Output and Formatting in MatLab

- The output of a MatLab command is displayed on the screen unless MatLab is directed otherwise. A semicolon `(;)` at the end of a MatLab command suppresses screen output (except for graphics and online help).

Paged output can be obtained using `more on`. Using the command provides buffered MatLab output. The format of the numbers output is set by typing `format` *type*. Seven different formats are available for the value of 10π:

```
format short          31.4159
format short e        3.1416e+01
format short eng      31.4159e+000
format long           31.41592653589793
format long e         3.141592653589793e+01
format long eng       31.41592653589793e+000
format short g        31.416
format long g         31.4159265358979
format hex                403f6a7a2955385e

format rat                3550/113

format bank               31.42
```

The `format compact` and `format loose` commands control spacing above and below the displayed lines while `format +` displays a +, - or a blank for a positive, negative or zero number respectively. The default format is `short`.

- The formats `short e` and `long e` are used to print numbers in scientific notation. Suppose you have the number 12345.6789. This can be represented as 1.23456789×10^4. On the other hand, 0.000123456789 can be represented as $1.23456789 \times 10^{-4}$

- Numbers in scientific notation can be added by making the smaller number have the same exponent as the larger number. For example, 1.2345×10^5 + 1.2345×10^3 can be computed as $1.2345 \times 10^5 + 0.012345 \times 10^5$ which is equal to 1.246845×10^5. Rounding this to 4 digits after the decimal point we get 1.2468 as the 5^{th} digit (4) is less than 5 and so is truncated from the number.

- MatLab saves previously typed commands in a buffer. These commands can be recalled with an uparrow key $\uparrow$. You can also recall previously typed commands by typing a few letters of the command and then the $\uparrow$ key. You can also cut and paste commands in the usual way.

MatLab File Types

- A **file** in named collection of data stored in a contiguous location on some storage media (i.e. disk). Files can contain ASCII characters (your MatLab code, say ass1_2016.m), a Mex file (a C program or a Fortran program, say, ass1_2016.c or ass1_2016.f77), an executable file (a compiled version of ass1_2016.m, say, ass1_2016.p), a MatLab figure file (say ass1_2016.fig), a binary data file (stored in a MatLab format, say ass1_2016.mat) or an image file (say lena.jpg).

- For example, the following MatLab code:

```
>> imshow('lena.jpg');
>> title('My copy of lena.jpg')
>> print('-djpeg','x.jpg');
```

produces the window:

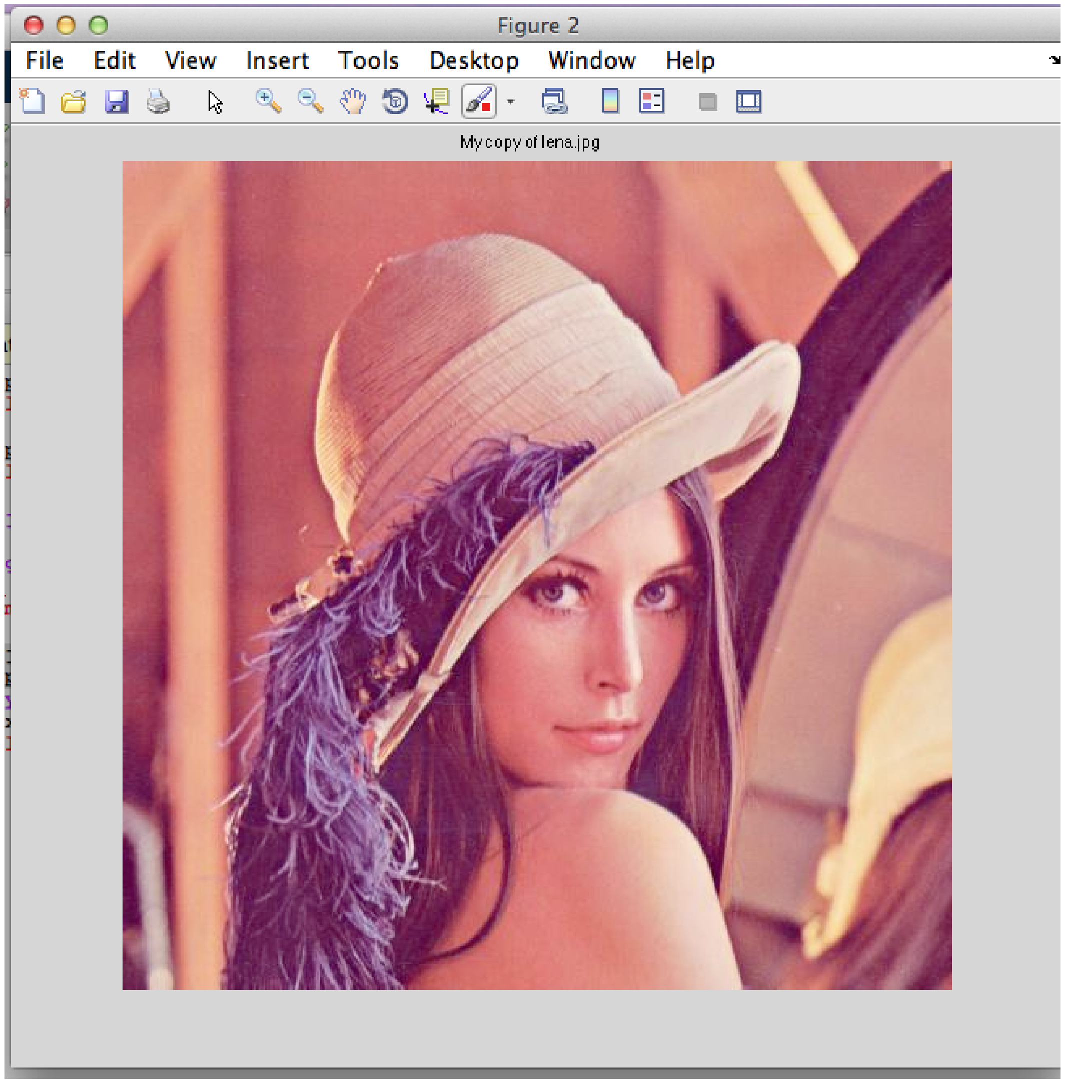


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
My copy of lena.jpg

- This code segment assumes `lena.jpg` is in the current directory `/Volumes/barron/CS2035`. A second jpeg image, `x.jpg` is created (has the title) by the print statement in the current directory. You can view it (in unix) by `xv x.jpg` (in a unix environment).

- The are 5 main file types in MatLab:

  1. **M-files** are ASCII files and have a `.m` extension. If you can read a **M-file** you can modify and copy it.

  2. **Mat-files** are binary files with the `.mat` extension. **Mat-files** are created by MatLab when you save/read data using the `save`/`load` command. This special format file can be read using the `load` command.

3. **Fig-files** are binary mat files files containing sufficient information to re-create figures and have a `.fig` extension. These files are created using **Save** and/or **Save As** commands from the **file** menu. **Fig-files** can be read using **open** *filename.fig*.
4. **P-files** are compiled **M-files** with extension `.p`. These files are created using the **pcode** command. These files cannot be read as ASCII files (so the source is hidden) but they can be executed.
5. **Mex-files** are MatLab callable Fortran or C programs and have a `.mex` extension after they have been compiled. Thus;

   ```
   mex ass1_2016.c
   ```

   would produce `ass1_2016.mexmaci64` on a 64 bit iMac workstation.

- Consider the MatLab code segment:

```
I=imread('lena.jpg');

[counts,x]=imhist(I(:));

stem(x,counts);

title('Histogram of lena.jpg');

saveas(gcf,'my_hist','fig');

openfig('my_hist.fig');
```

- `imhist` expects either a 1D vector or 2D image, but color images are actually 3D: three 2D images of the red, green and blue color planes. `I(:)` returns the values of `I` as a 1D vector of pixel values. `imhist` has no problem handling this whereas `imhist(I)` results in an error.

- `gcf` is a MatLab function that return the **handle** (a pointer) to the current figure being displayed.
- `stem` displays a stem plot of the data as lines extending from a baseline along the x-axis. `x` contains the x coordinates and `counts(x)` contains the number of times each pixel in `x` occurred in the image.
- `saveas` is a matlab function that, in this case, saves the histogram as `my_hist.fig`. It could be used to save the histogram in many other formats, such as jpeg (jpg), tiff (tif), bmp, ras, png, etc.
- Both `stem` and `openfig` cause the same identical histogram to be displayed (one for `stem` and a second for `openfig`).

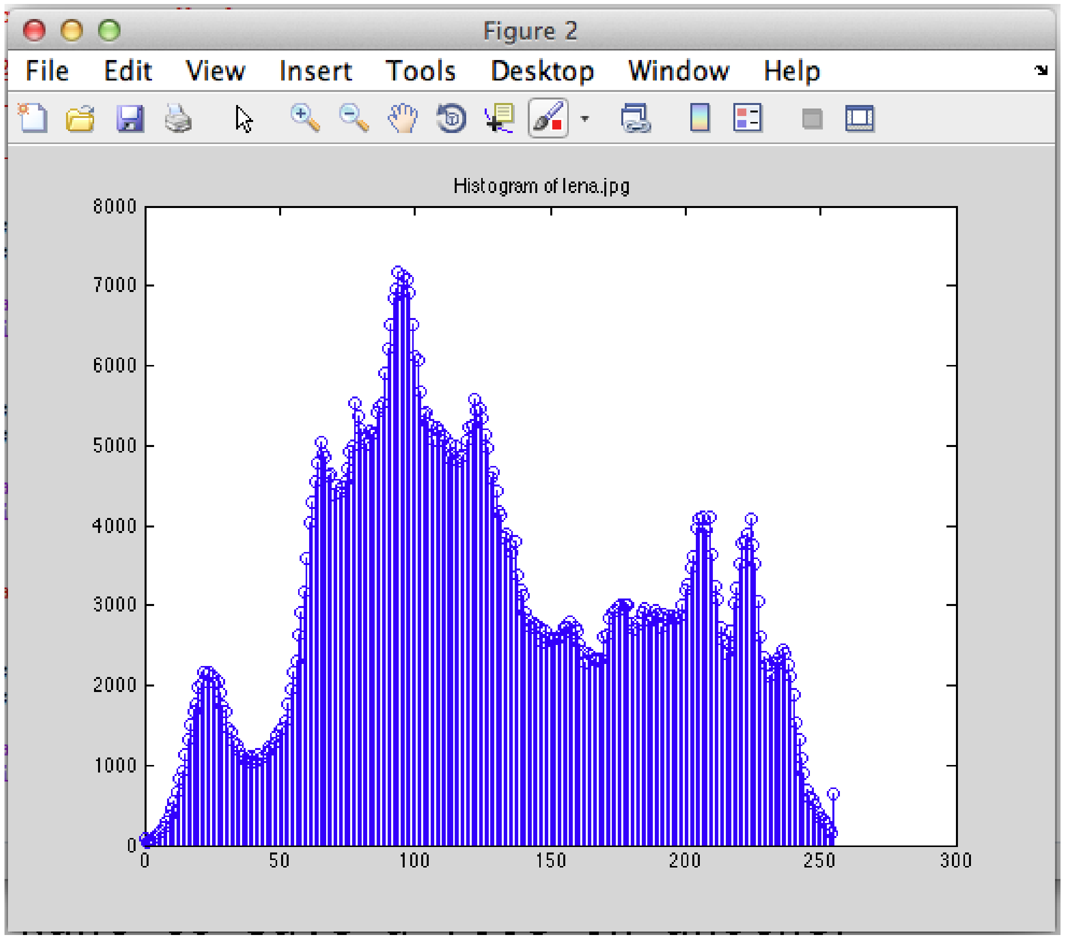


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
Histogram of lena.jpg
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300

Files and Directories

- A **file** is collection of data or information that has a name (the filename) and is stored on some computer media (disk, for example). Typically, files are stored in **directories** or **folders** in a tree like structure.
- `cd` allows you to change directories, `ls` list the files in the current directory, `pwd` prints the current working directory (where am I), `dir` is like `ls`, `mkdir` makes a new directory, `rmdir` removes an (empty) directory.
- `what` lists the M-files in the current directory and `which x.m` specifies the full pathname where `x.m` resides.
- `path` is a MatLab command that lists all pathnames MatLab can access

on your machine.

- You can add new directories to your path. For example, to add a path

  ```
  /Volumes/barron/MATLAB
  ```

  use `addpath /Volumes/barron/MATLAB`. Use `savepath` to save this modification of the pathnames permanently.

Some Things to Remember

- MatLab is platform independent, for the most part. MatLab executes the same way on different machines. Only commands that depend on the local operating system, such as an editor like **vi** or **emacs**, are different.

- **Not being in the right directory**: only files in the current directory can be accessed. Change to the appropriate directory to execute a particular MatLab file using the `cd`.

- **Not saving files in the correct directory**: by default you save files in the current directory. If you want to save a file in another directory, you need to you should change directories first (or include the pathname in the filename).

- **Not overwriting an existing file while editing**: If you run an M-file, do not like the results, edit the file and run it again, you may get the same results as before, as previously compiled code may execute instead. The simple cure is to clear the workspace of this function with `clear`.

MatLab Examples

```
>> 2+2

ans =

     4

>> x=2+2

x =

     4

>> y=2^2+log(pi)*sin(x);

>> y; % nothing happens as ; suppresses the output

>> y

y =
    3.1337

>>theta = acos(-1)

theta =

    3.1416

>> format short e

>> theta

theta =

    3.1416e+00

>>format long

>>theta

theta =

    3.14159265358979
```

```
>> 2^5/(2^5-1)

ans =

    1.0323

>> 3*(sqrt(5)-1)/(sqrt(5)+1)^2-1

ans =

   -0.6459

>> area = pi*(pi^(1/3)-1)^2

area =

    0.6781

>> exp(3)

ans =

   20.0855
```

```
>> log(exp(3))

ans =

    3

>> log10(exp(3))

ans =

    1.3029

>> log10(10^5)

ans =

   5

>> exp(pi*sqrt(163))

ans =

   2.6254e+17
```

```
>> x=log(17)/log(3)

>> x

   2.5789

>> sin(pi/6)

ans =

   0.5000

>> cos(pi)

ans =

   -1

>> tan(pi/2) % this number is undefined?

ans =

    1.6331e+16  % what a version of MatLab gives
```

```
>> (sin(pi/6))^2+(cos(pi/6))^2

ans =

    1

% cosh and sinh are hyperbolic functions

>> x=32*pi; y=(cosh(x))^2*(sinh(x))^2

>> y

y =

  2.7293e+173
```

Complex Numbers

- MatLab recognizes the letters i and j as the imaginary number $\sqrt{-1}$. A complex number has a real part and an imaginary part. For example, 2+5i has real part 2 and imaginary part 5.

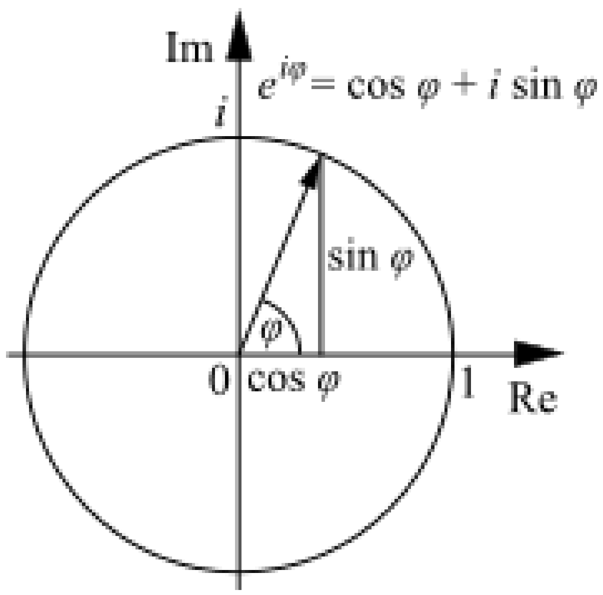
- Engineers: the Fourier Transform of a signal or image gives an array of 1D or 2D complex numbers - we'll see this later.
- Arithmetic operations are done by MatLab as follows:
  - Addition: $(a + ib) + (c + id) = (a + c) + i(b + d)$.
  - Subtraction: $(a + ib) - (c + id) = (a - c) + i(b - d)$.
  - Multiplication: $(a + ib) \times (c + id) = (ac - bd) + i(bc + ad)$.
  - Division: $\frac{(a+ib)}{(c+id)} = \frac{(a+ib)(c-id)}{(c+id)(c-id)} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2}$.

- Also the definitions may be handy:

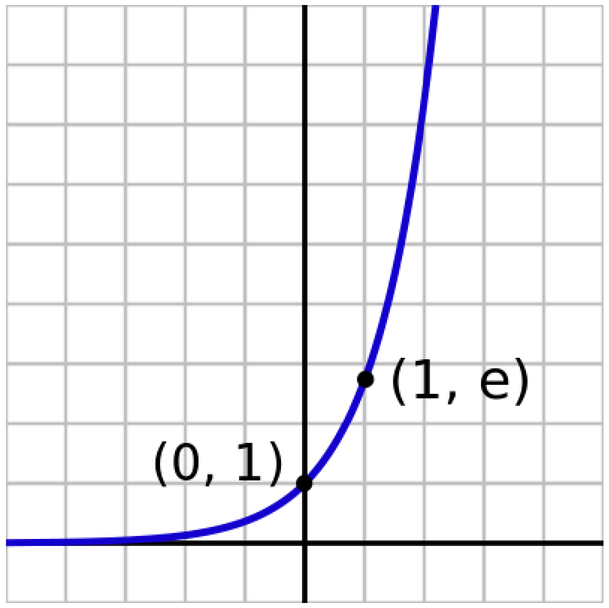
$$\begin{aligned} e^{ix} &= \cos x + i \sin x \\ e^{-ix} &= \cos x - i \sin x. \end{aligned}$$

- The exponential function is the function e^x, where e is the number (approximately 2.718281828459046) when $x = 1$.

- Figure A show the mathematical relation of $e^{i\phi}$ in terms of $\sin$ and $\cos$ and Figure B shows a plot of this function. Note that e^0 is 1 and e^1 is 2.718281828459046 (Figures from http://en.wikipedia.org/wiki/Exponential_function).



A (plot of $e^{ix} = \cos x + i\sin x$)



B (plot of e^x)

- MatLab Complex Numbers examples:

```
>> c = 2+3i

c =

   2.0000 + 3.0000i

>> real(c)
```

```
ans =

     2

>> imag(c)

ans =

     3

>> abs(c) % magnitude of c=sqrt(a^2+b^2)

ans =

    3.6056

>> % MatLab computes abs of c=2+3i as:

>> sqrt(2^2+3^2)

ans =

    3.6056
```

```
>> conj(c) % complex conjugate of a+bi is a-bi

ans =

   2.0000 - 3.0000i

>> c*conj(c) % (2+3i)*(2-3i)=4+6i-6i-9i^2=4+9=13

ans =

    13
```

- $\frac{(1+3i)}{(1-3i)}$ is computed by multiplying the numerator and denominator by $(1+3i)$, the complex conjugate of $(1-3i)$. The denominator now becomes real and we obtain the complex number $\frac{(1+3i)(1+3i)}{(1-3i)(1+3i)} = \frac{1+6i+9i^2}{10} = \frac{-8+6i}{10} = -0.8+0.6i$.

```
>> (1+3i)/(1-3i)

ans =
    -0.8000+0.6000i

>> exp(i*pi/4)

ans =

    0.7071+0.7071i

>> format long

>> exp(1) % e^1

ans =

   2.718281828459046

>> exp(pi/2) % 2.718281828459046^(pi/2)

ans =

   4.810477380965351

>> exp(-pi/2) % 2.718281828459046^(-pi/2)
```

```
ans =

   0.207879576350762

>> % Complex number: use e(ix)

>> % Consider x=pi/2*i and x=-pi/2*i

>> % Here pi/2 is multiplied by i

>> exp(pi/2*i)

ans =

   0.0000+1.000i

>> exp(-pi/2*i)

ans =

   0.0000-1.000i

>> % Consider exp(pi/2i)
>> % Here pi is divided by 2i ==> do complex division
>> % using the complex conjugate of 2i, -2i
>> % pi/2i * -2i/-2i = -2*pi*i/4 as 2i*-2i=4
>> % Therefore exp(pi/2i) is the same as exp(-2*pi*i/4)
>> % or exp(-pi/2*i)
>> exp(pi/2i)
ans =
    0.0000-1.0000i
```

Writing a Program in MatLab

- A program (in any language) has to input or create data, do some calculations on that data and output the result.
- Input and output can be from/to files. The input might also be generated in some way (say by mathematical formulae).
- The output can also be graphical entities, such as 2D and 3D graphs.
- Before we can write a useful MatLab programs, we need to understand the basic data structures (arrays) and various control statements, such as:
  - **loops** (we iterate on a loop index and at each iteration we perform some calculation, maybe storing the result in an array indexed by

the loop index),

– **if-then-else** statements (ok we have some data, we need to do calculations on that data according to whether some conditions are true or false) and

– **while** statements (repeat a calculation while some condition becomes false).

- We also need to:

  – understand boolean variables and expressions (having values **true** and **false** and how to combine such expressions),

  – know how to do basic I/O (Input/Output),

  - know how to label our output (both text and graphical) using character strings and
  - know how to do matrix manipulations to solve problems.

- We put our MatLab code in a script file with an extension `.m`, for example `ass1.m`. We then execute these script files by typing the script file name without the `.m` in the command window, for example `ass1`.
- It is also useful to write MatLab functions to modularize our code. These can then be called from other functions of our script files.
- Using these tools we can code algorithms to solve problems (i.e. write programs). Once we have these tools we can build on them, adding more advanced tools.