Lecture 1:

Review of Matlab Programming

What is Matlab?

- Matlab stands for ________ __________.
- Matlab is a programming language optimized for linear algebra operations.
- It is very useful for numerical computation and is commonly used by mathematicians and engineers in academia and industry.

Basic Arithmetic

- If we type a calculation, when we press ENTER the result appears as ans.

```
2+3
ans =

        5
```

- Pressing the up arrow repeats the last input.
- In addition to basic + - * / there are basic mathematical functions

```
exp(2)    sin(pi)    asin(2)
```

- There may be round-off errors.
- Watch out for the values Inf and NaN. If you see these values, then you probably __division by zero__.

Variables

- To assign a value to a variable, just type it. We don't need to declare variables first: x=2
- Matlab is *weakly typed*, which means the variable type is flexible.

  x=2 x=2.3 x=2+4i x='hello'
- To see what variables are available, look in the Workspace window or type: who
- To delete a variable x: clear x
- If you type just clear, all variables will be deleted.

Vectors

- We enclose vector values in square brackets.

v = [2 3 4 5 6]

- We can look up a value at a position: v(2)
- The colon operator takes on a range of values

v = 2:6

- More generally we set start:step:end (default step=1)
- Ex Make a vector of all multiples of 3 less than 1000.

w = 3 : 3 : 1000

- What is the last entry in w? 999

Appending Vectors

- We can append one vector onto another by enclosing them in brackets, separated by commas.

```
v = [1:5, 10,  2:2:16]
```

- This trick also works for strings.

```
s = [‘pokemon’, ’rule!’]
```

- This is particularly useful when we want to combine strings and numbers.
- We can convert a number to a string using the command num2str. (Or go the other way with str2num.)

```
disp(['The value of x is ', num2str(x)])
```

Matrices

- To make a 2D matrix, the semi-colon skips to the next row:

  A = [2 3 4; 5 6 7]

- We look up values by row,column: A(2,3) = 7;
- You can look up a submatrix with the colon: A(1:2,2:3)
- Using just a colon gets all possible values: A(:,2:3)
- Special matrices

  rand(10,20)   eye(3)   zeros(4,5)   ones(3,2)

- As matrices get large, you can suppress output with a semi-colon at the end.

  R = rand(20,20);

Matrix Operations

- Matrix multiplication: A*B A^3
- Component-wise operations: A.*B A.^3
- Inverse: inv(A)
- Transpose: A'
- Look up matrix size: size(A)
- Eignenvalues: [v,d] = eig(A);
- Linear solver Ax=b: x = linsolve(A,b);
- Vectorize a matrix: A(:)
- Change matrix size: reshape(A,[r,c]);

Linear Algebra Pop Quiz

- Suppose we make a matrix: A=[1 2; 3 4];
- Write what each command below does.

A'=

A(:) =

A^2 =

A.^2 =

Basic Flow Control

```
while i > 0
    ...
end

for i = 1:10
    ...
end
```

```
if x >0
    ...
elseif x<0
    ...
else
    ...
end
```

An Odd Example

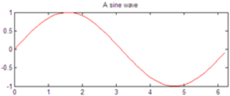
- The function mod(a,b) tells the remainder after a is divided by b.
- So a is multiple of b if mod(a,b)=____.

```
for i = 1:100
        if mod(i,2) == 0
                disp([num2str(i), ' is even.']);
        else
                disp([num2str(i), ' is odd.']);
        end;
end;
```

Basic Plotting

- The plot function takes two vectors as input. The first vector goes on the horizontal axis (x) and the second on the vertical (y).
- Ex Plot a sine wave on $[0, 2\pi]$.

```
x=0:0.1:2*pi;
y=sin(x);
plot(x,y,'r')
axis([0,2*pi,-1,1])
title('A sine wave')
```



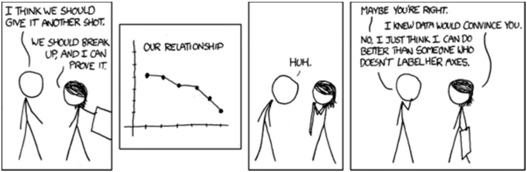
- You can suprress the axis numbers with: axis off
- Note the axis command sets the x and y bounds:

  axis([xmin, xmax, ymin, ymax])
- You can reset the axis and tick marks manually too.
- You can (and should) add text to the plot:

  title xlabel ylabel gtext legend

Label Label Label

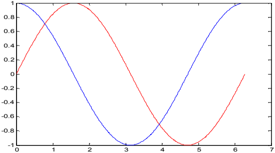
- I will deduct points if you do not label your plot axes or title your images.


I THINK WE SHOULD GIVE IT ANOTHER SHOT.
WE SHOULD BREAK UP, AND I CAN PROVE IT.
OUR RELATIONSHIP
HUH.
MAYBE YOU'RE RIGHT.
I KNEW DATA WOULD CONVINCE YOU.
NO, I JUST THINK I CAN DO BETTER THAN SOMEONE WHO DOESN'T LABEL HER AXES.

Plotting on Common Axis

- The hold command tell Matlab to plot things on top of each other, rather than erasing the previous picture.
- hold on forces all subsequent plots to appear on top of the last plot.
- hold off releases the plot, so any new plots will erase the current picture.

```
x=0:0.01:2*pi;
plot(x,sin(x),'r');
hold on;
plot(x,cos(x),'b');
hold off
```



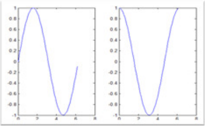
Subplots

- The subplot command divides the figure into windows.

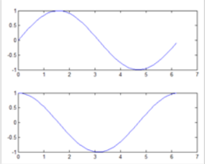
  subplot(TotalNumRows, TotalNumCols, index)
- The index goes from left to right, top to bottom (raster order).
- Which box would subplot(2,3,4) get?
- *Pro Tip:* If the numbers are all single digits, we can omit the commas: subplot(234)

Subplots

```
x=0:0.1:2*pi;
subplot(1,2,1); plot(x,sin(x));
subplot(1,2,2); plot(x,cos(x));
```



```
x=0:0.1:2*pi;
subplot(2,1,1); plot(x,sin(x));
subplot(2,1,2); plot(x,cos(x));
```

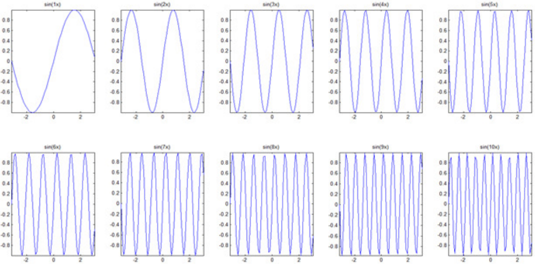


Subplot Example

- Ex Plot the graphs of $\sin(Nx)$ on $[-\pi, \pi]$ for N=1 to 10.

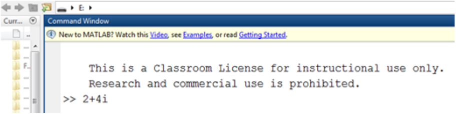
```
x = -pi:0.1:pi;

for i = ______________________________

   subplot( ___________ , ___________ , ___________ );

   plot( _____________________________________

   title( _____________________________________
end
```

Subplot Example



Navigation

- You can change directories by clicking the little folder icon at the top.



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

```
    This is a Classroom License for instructional use only.
    Research and commercial use is prohibited.
>> 2+4i
```

- Check current position: pwd
- Print all files in the current folder: ls

Saving & Loading Data

- You can save your current variables to a Matlab save file *(.mat file)*.

```
save my_data x y z
```

- You will see the file my_data.mat appear in the current folder.
- You can quit Matlab, come back a couple days later, and load the variables x,y,z to the workspace.

```
load my_data
```

Reading & Writing Images

- Load images into a matrix with imread.

```
A = imread('mypic.jpg');
```

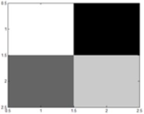
- Write a matrix to an image with imwrite.
- You need to specify the image format. Bitmaps could be used to avoid compression artifacts.

```
imwrite(A, 'mypic.bmp', 'bmp');
```

Grayscale Images

- A grayscale image is given by a 2D matrix with the low value being black, the high value being white, and everything in between denoting a shade of gray.
- Typically, optical images are 8-bit images which take on integer values in the range [0,255]. (0=black, 255=white)
- The top left corner of the image is position (1,1).
- Note Matlab records matrix position as (row,col), so it's (y,x) with the y values inverted.

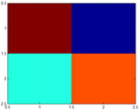
```
A(1,1)=255;
A(1,2)=0;
A(2,1)=100;
A(2,2)=200;
```



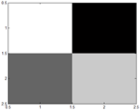
Displaying Images

- You can display a matrix with the command imagesc.
- Matlab defaults to an annoying red-blue "jet" colormap. So we need to tell Matlab to use a grayscale display by typing: colormap gray
- Often we prefer not to display the axis numbers on images: axis off

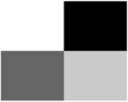
```
imagesc(A);
```



```
imagesc(A);
colormap gray;
```



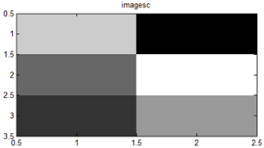
```
imagesc(A);
colormap gray;
axis off;
```

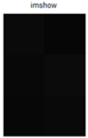


imagesc vs. imshow

- The imagesc command fills the window with the image, ignoring the aspect ratio.
- It draws grayscale images from min=black to max=white.
- If you want to see how the image would appear like in a web browser with proper aspect ratio and in 8-bit format, use the imshow command.

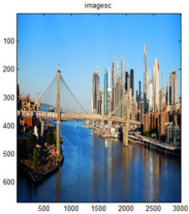
A(1,1)=10; A(1,2)=6; A(2,1)=8; A(2,2)=11; A(3,1)=7; A(3,1)=9;

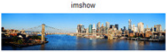




imagesc vs. imshow

- The difference between imagesc and imshow is most obvious when the length and width of the image are very different.





Data Format

- Images typically come in 8-bit uint8 format.
- But we can't do math on the images in this format.
- So we cast to double before we do our arithmetic tricks.

```
A = double(A);
```

- Then when we're done, we cast back to 8-bit image format.

```
A = uint8(A);
```

- Some Matlab functions require 8-bit images as input, others prefer double images.

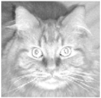
Image Arithmetic

- To brighten a grayscale image A by 60%, we multiply all values by 1.6.

```
A = 1.6 * A;
```

- But multiplying an integer by 1.6 does not necessarily give an integer in the range [0,255].
- To preserve image formats, we need to cast to double and then back to integer 8-bit.

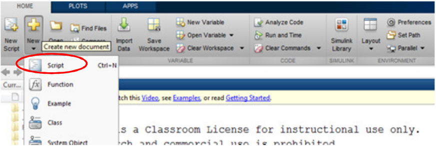
```
A = double(A);
A = 1.6 * A;
A = uint8(A);
```



- To see the effect of brightening an image, you really should use the imshow command, not imagesc.

Writing Scripts

- A Matlab script is a set of commands that you can save as .m file.
- Select Script from the New dropdown menu.



tch this Video, see Examples, or read Getting Started.

s a Classroom License for instructional use only.

ch and commercial use is prohibited

- Ex Display a 8x8 chessboard where each square is 10x10 pixels.

Writing Functions

- We can create user-defined functions that Matlab can call. We call these *functions* or *m-files*.
- The first line is:

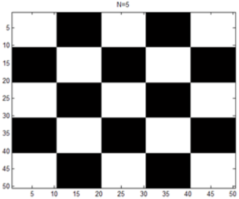
```
function [out1, out2] = function_name (in1, in2)
```

- Comments start with a % sign. Matlab will ignore these lines, but they are helpful for people who read your code later. Matlab highlights comment lines in green.

- Ex Write a function that returns the image of a $N \times N$ chessboard.

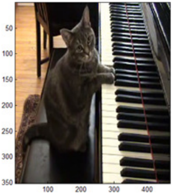
Chessboard Function

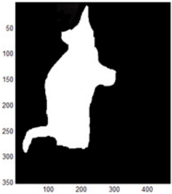
```
function [ A ] = chessboard ( N )
% Return image of NxN chessboard.
% Each square on the chessboard is 10x10 pixels.
for i=1:N
   if mod(i,2)==0
      color=0;
   else
      color=255;
   end;
   for j=1:N
      A(1+10*(i-1):10*i, 1+10*(j-1):10*j)=color;
      color = 255-color;
   end;
end;
A = uint8(A);
```



Binary Images

- A binary image is black or white, no shades of gray.
- A binary image typically has values of just 0 (black) or 1 (white).
- Binary images are useful for detection tasks, e.g. identify if each pixel belongs to a cat. These types of images are often referred to as a *mask*.

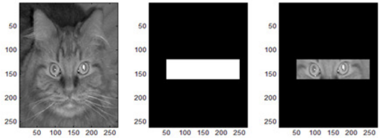




Binary Images

- We can mask out part of an image by doing component-wise multiplication by a binary image.

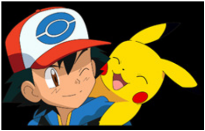
```
A = imread('cat.jpg');  A = double(A);
D = zeros(size(B));  D(120:160,50:250)=1;
subplot(131); imagesc(A);
subplot(132); imagesc(D);
subplot(133); imagesc(D.*A);
```



Color Images

- A standard <u>color image</u> has 3 channels indicating the intensity of Red, Green, and Blue light (RGB).
- A color image is a 3D matrix, with the 3rd dimension representing color.
- Think of a color image as being a stack of 3 grayscale images.

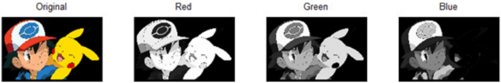
```
A=imread('ash.png');
size(A)
      ans =
              708      1129         3
```



Color Images

- We can access an individual color channel by the 3rd dimension.

```
A = imread('ash.png');
subplot(141);  imshow(A); title('Original');
subplot(142);  imshow(A(:,:,1));  title('Red');
subplot(143);  imshow(A(:,:,2));  title('Green');
subplot(144);  imshow(A(:,:,3));  title('Blue');
```



Color Images

- Each pixel in a color image is a 3D vector.

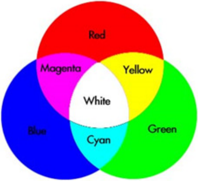
Black = (0,0,0)

White = (255,255,255)

Red = (255,0,0)

Green = (0,255,0)

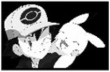
Blue = (0,0,255)



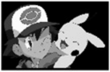
- The color of a pixel is determined by the mixture of the RGB values.

Original

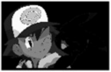
Red



Green



Blue



Color Image Example

B =

🟥	⬜	🟥	🟥	🟥
🟥	🟥	🟥	🟪	🟥
🟥	🟥	🟥	🟥	🟥

- Start with a red background.

B(1:3,1:5,1)=_____; B(1:3,1:5,2)=_____; B(1:3,1:5,3)=_____;

- Now make the white pixel.

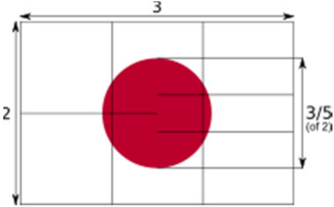
B(1,2,1)=_____; B(1,2,2)=_____; B(1,2,3)=_____;

- Finally make the purple pixel.

B(2,4,1)=_____; B(2,4,2)=_____; B(2,4,3)=_____;

Color Image Example 2

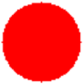
- Ex Make the flag of Japan.



Color Image Example 2

```
h = 50;                          % Height of flag.
w = round(3*h/2);                % Width of flag.
d = round(3/5 * h);              % Diameter of circle
A(1:h,1:w,1:3) = 255;              %Start with white background.
for i = 1:h
          for j = 1:w
               if (j- w/2)^2+(i-h/2)^2 <=(d/2)^2
                    A(i,j,2)=0;   A(i,j,3)=0;
               end;
          end
end
A = uint8(A);
imshow(A);
```

Color Image Example 2



- If you look closely at the circle we produced, you may see the edges are not smooth.
- This phenomenon of having "blocky" or "staircased" edges is called aliasing.
- How can we remove it?

Processing Color Images

- In this course, we mostly deal with how to process grayscale images.
- Suppose you have a Matlab function that processes a grayscale image.
- To process a color image, you just need to run your function 3 times.

```
for i = 1:3
        New_A(:,:,i) = MY_FUNCTION ( A(:,:,i) );
end
```

- You can turn a 3-channel color image into 1-channel grayscale image using rgb2gray.

Matrix Operations

- In Matlab, we want to avoid loops and make use of matrix operations to make it run faster.

- Ex Write a function that returns the average of the color bands.
- Note: This is a crude way to turn a color image into grayscale. The Matlab command rgb2gray actually does a weighted average.

Averaging 3 Bands

- Bad Answer

```
function [A] = average(B)
B = double(B);
[m,n,k] = size(B);
for i=1:m
    for j=1:n
          A(i,j) = (B(i,j,1)+B(i,j,2)+B(i,j,3)) / 3;
    end;
end;
```

- Better Answer

```
function [A] = average(B)
B = double(B);
A = (B(:,:,1)+B(:,:,2)+B(:,:,3)) / 3;
```

- Best Answer

```
function [A] = average(B)
B = double(B);
A = mean(B,3);
```