Introduction to MATLAB

- Very sophisticated "graphing calculator"
- High-level programming language for scientific computing
- Similar to PYTHON, with more built-in BUT not open-source
- vs. c or FORTRAN, many built-in commands, less-complex syntax
- vs. c or FORTRAN, Interpreted language (code translated during run), with some precompiled functions
- IF used carefully, good balance of speed and ease of use

Our introduction follows, in part, section 1 of the excellent "An introduction to Matlab for dynamic modeling" by Guckenheimer and Ellner:
www.cam.cornell.edu/ $\sim$ dmb/DMBsupplements.html, as well as material from Prof. Mark Goldman, UC Davis

Launch matlab!

Command window (graphing calculator) mode

```
>> 1+3
ans =
     4
```

Assigning **values** to **variables**

```
>> a=1+3
a =
     4
```

Displaying values

```
>> a
a =
     4
```

Suppressing display of output in MATLAB

```
>> a=1+3;
```

BASIC OPERATIONS:

Arithmetic

```
+
-
/   (divide)
*   (multiply)
^   (exponent)
abs(x)   (absolute value)
cos(x), sin(x), tan(x)
exp(x) exponential function e^x
log(x)   log to base e
sqrt(x) square root
```

The latter are built-in functions

```
>>   a = sqrt(5)
a =     1.4142
>> a=2 ; b=4 ; c=a^b
c =     16
```

There are also some built-in variables

```
>> pi
ans =     3.1416
>> cos(pi)
ans = -1
```

FUNDAMENTAL PROGRAMMING SYNTAX:

- LHS = RHS
- value RHS assigned to LHS, NOT other way around

```
>> c=2
c = 2

>> 2=c
??? 2=c
Error: The expression to the left of the equals sign is
not a valid target for an assignment.
```

Another example:

```
>> a=2;b=4;
>> a=b;
>> a,b
a =     4
b =     4
>> a=2;b=4;
>> b=a;
>> a,b
a =     2
b =     2
```

Variable names:

```
>> a1=2
>> my_favorite_variable_number_2=4
```

VALID: Letter followed by letters, numbers, underscore character.
NOTE! capitalization matters. $A \neq a$!!

NOT VALID:

```
>> my_favorite_variable#2=4
??? my_favorite_variable#2=1
>> 2a=1
Error: Unexpected MATLAB expression.
```

And using periods gives a different, more advanced type of variable (data structure) that we won't discuss now

```
>> my.favorite.variable.name=1
my =
    favorite: [1x1 struct]
```

ORDER OF OPERATIONS:

PEMDAS

parenthesis
exponentiation
multiplication
division
addition
subtraction

Say we want: $a = 2$, $b = 4$, $c = \frac{a}{a+b}$

```
>> a=1 ; b=4 ; c=a/a+b
c =      5
```

```
>> a=1 ; b=4 ; c=a/(a+b)
c =    0.2000
```

!! When in doubt, put lots of parentheses !!

CHECK: What do you get for the below, and why?

```
>> a=1 ; b=4 ; c =0 ;
>> d=a/0+b
>> d=a/(0+b)
>> d=-a^2
>> d=(-a)^2
```

Check: what variables stored in memory:

`whos` command (MATLAB)

```
>> whos
  Name        Size                Bytes  Class     Attributes

  a           1x1                     8  double
  b           1x1                     8  double
  c           1x1                     8  double
```

OR look in “workspace” to see variables and values
OR just type variable at command line

```
>> a
a =     1
```

GOOD PRACTICE: clear variables before starting to program
(MATLAB:)

```
>> clear all
>> whos
```

Commands stored in memory:

See command history, or just hit “up arrow”

Representation of numbers in scientific computing: finite precision

Standard: IEEE floating point arithmetic.

Important feature – finite precision: approx 16 significant digits

Display more digits:

```
 >> a=0.1
a =
    0.1000
>> format long
>> a
a =   0.100000000000000
```

Roundoff error:

```
 >> a=4/3 ; b=a-1 ; c = (3*b)-1

c =

    -2.220446049250313e-16
```

OVERFLOW AND UNDERFLOW:

Maximum number: $\approx 10^{308}$
Minimum number: $\approx 10^{-308}$
Overflow:

```
>> a=10^400
a =   Inf
```

Underflow

```
>> a=10^-400
a =   0
```

Another special number: not defined

```
>> 0/0
ans =
   NaN
```

HELP !! (MATLAB)

How does a command or function work?

```
>> 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.

   Reference page in Help browser
      doc sqrt

>> doc sqrt
```

Thanks anyway, but what SHOULD I be looking up? the lookfor command

```
>> lookfor exponent
EXP    Exponential.
EXPINT Exponential integral function.
EXPM   Matrix exponential.
...
```

HELP !!: Remember Google and stackoverflow, etc are your friends

VECTORS

A vector value is just a list of scalar values.
Arranged horizontally into a **row vector:**

$$\vec{x} = (6\ 12\ 5)\,. \tag{1}$$

Or vertically into a **column vector:**

$$\vec{x} = \begin{pmatrix} 6 \\ 12 \\ 5 \end{pmatrix}\,. \tag{2}$$

In MATLAB, row vector:

```
>>x=[6  12  5]
x =     6     12    5
```

OR

```
>>x=[6 ,12 ,5]
x =     6     12    5
```

In MATLAB, column vector:

```
>>x=[6 ; 12 ; 5]

x =

     6
    12
     5
```

The **transpose** operation switches between row and column vectors. This is given by dot prime in MATLAB. That is, in MATLAB:

```
>> y=x.'
```

Easy way to make (row) vectors [MATLAB,]

x = start : increment : end

```
>> x=0:1:10
x =
     0     1     2     3     4     5     6     7     8     9    10

>> x=1:.1:1.5     [MATLAB]
x =
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
```

Accessing vector elements (or components):
MATLAB:

```
>> x5=x(5)
x5 =    1.4000

>> x(7)
??? Index exceeds matrix dimensions.
```

Indexing starts with 1; x(0) does not work.

Exercise 0.1 : Have MATLAB compute the values of

1. Make a list numbers spaced by 0.2, between a minimum value of 1 and a maximum value of 20. Assign that list to the variable name `myvector`.
2. use help to look up the `linspace` command, and repeat the previous command using `linspace`
3. pick out the 4th value of `myvector` and assign it to the variable name `fourthelement`

TWO BASIC OPERATIONS ON VECTORS:

Multiplication by scalar

$c\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} cx_1 \\ cx_2 \\ cx_3 \end{pmatrix}$; that is, $x_j \to c\,x_j$

Matlab *

```
>> x=[1;2;3] ; 3*x

ans =
     3
     6
     9
```

Addition of two vectors

$$\vec{x} + \vec{y} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \end{pmatrix}$$

Subtraction similar

Works the same for row and column vectors

Matlab + and -

```
>> x=[1;2] ; y=[0;-2]; z=x+y
z =
     1
     0
>> z+ [2 2]
??? Error using ==> plus
```

Matrix dimensions must agree: add row + row or col + col

CAUTION! Multiplication of vectors does NOT work the same way ... more later (matrix- vector multiplication)

Matrices

Think of matrices as $N \times M$ tables of numbers N rows, M columns

MATLAB:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}$$

entries A(n,m)

```
>> A=[1,2,3 ; 4 6 7 ; 1 3 4]

A =
     1     2     3
     4     6     7
     1     3     4

>> A23=A(2,3)
A23 =     7
```

N-element col. vector: N, M=1

M-element row. vector: N=1, M

Otherwise, we will mostly consider square matrices (N=M)

Exercise 0.2: Have MATLAB compute the values of

1. 3*A. From this, write down a rule for what matrix multiplication by a single number (scalar) means.
2. A+A. From this, write down a rule for what summing matrices means.
3. A*A. From this, conclude that multiplying two matrices (like multiplying two vectors) means something very different indeed (and to be cautious about)!
4. Experiment with the command .* with both vectors and matrices. What does THIS command do?

FUNDAMENTAL CONCEPT: Matrix-vector multiplication

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 \begin{pmatrix} A_{1,1} \\ A_{2,1} \end{pmatrix} + x_2 \begin{pmatrix} A_{1,2} \\ A_{2,2} \end{pmatrix}$$

e.g.

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

In general,

$$\begin{pmatrix} | & \cdots & | \\ a_1 & \cdots & a_n \\ | & \cdots & | \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_j x_j \begin{pmatrix} | \\ a_j \\ | \end{pmatrix}$$

Exercise 0.3 : Compute the below by hand ...

- $$\begin{pmatrix} 2 & -3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$
- $$\begin{pmatrix} 2 & -3 & 2471 \\ 0 & 1 & 4 \\ 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix}$$

MATLAB * operator

```
>> A=[1 1 ; 1 2] ; A*[1 ; 2]

ans =

     3
     5
```

In $y = Ax$, A must have same number of columns as x has rows.

Nonsense:

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

```
>> A=[1 1 ; 1 2] ; A*[1 ; 2 ; 4]
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Exercise 0.4 :

- Check your answers to the hand calculations from the previous exercise, using MATLAB.

PLOTTING

- Plotting is MATLAB is great, and best learne just by doing ...
- Write a program `plot_a_sin_wave.m` tha plots `sin(x)` from $x = -\pi$ to $x = \pi$. HINT: type `doc plot`!

The .m file: time to code!

- Type edit at command line
- Put a few of your favorite commands in the editor, and save it as `myprogram.m` Remember the folder where you saved it.
- You've made a .m file — that is, a MATLAB program!
- Navigate (click on the ..., or use the `cd` command) in the command window to the location where you stored the program
- To run it, type `myprogram` at the command line

The for loop

Use for repeated operations.

Basic structure:
MATLAB CODE

```
 for n= 1:9
    disp(n)
end
```

- Use edit to code this into a program `myloop.m`, save it and run it!

COMPONENTS OF THE FOR LOOP:

n loop variable
1:3 loop vector
disp(n) or print(n) command

Code starts with n equal to first element in loop vector

runs command

advances n to next element

repeats

quits when have covered all elements of loop vector

In more detail:

- in any for loop, we define a list of numbers, here 1 through 9. think of this as the ”loop vector.” The loop vector can be **any list of numbers** – it does not have to be “integers counting up.”
- the loop variable, n, starts at the first number in the list. it is set equal to that value
- the commands (here, just printing the value of n to the screen) are run
- then when end is reached, n is reset to the NEXT number in the list, the commands are run, and the process is repeated
- it terminates when all entries of the loop vector have been used.

Run and describe what happens for these examples:

```
for p=[4 6 67 -1]
    disp(p)
end
```

```
for k=1:2:5
    disp(k)
end
```

```
a=0
for k=1:5
    a=a+k
end
```

CULMINATION (1)

Fibonacci numbers, matrix multiplication, and eigenvalues: (From Strang, Linear Algebra and its Applications). The Fibonacci sequence is

$$0, 1, 1, 2, 3, 5, 8, 13, ...$$

and occurs all over biology, e.g. in the number of seeds in subsequent "rings" of a sunflower (D. O'Connell, Scientific American, 1951). The $k + 2^{nd}$ element in the sequence is defined by being the sum of the former two elements:

$$n_{k+2} = n_{k+1} + n_k \ .$$

Define a two-element vector

$$x_k = \begin{pmatrix} n_{k+1} \\ n_k \end{pmatrix}$$

and write a matrix multiplication that finds x_{k+1} by multiplying x_k by a 2×2 matrix A. Specifically, write down A.

Write a MATLAB program that computes the first 300 elements in the Fibonacci sequence. Also, compute the ratio of subsequent elements in the sequences: $r_k = \frac{n_{k+1}}{n_k}$ vs. k, for k from 1 to 299. What is the behavior of r_k as k grows?

HINT: If you did this right, your answer will (incredibly) involve the golden mean $\frac{1+\sqrt{5}}{2}$.

CULMINATION (2)

- Imagine that you have a giant neural network, and each cell is either firing ("on") or not ("off"). Each second, for every neuron that is already on, two more switch on. This is a model of EXCITATORY SYNAPTIC COMMUNICATION from the "on" neurons. At time $t = 0$ seconds, 1 neuron is "on." Write a program, called `neural_explosion.m` that does the following:
  - using a for loop, compute a vector `number_on` that is the number of neurons on at each second, from $t = 0$ to $t = 30$ seconds.
  - Make a plot of the number of neurons on vs. time. Label the axes "time" and "number on." Hint: type `help plot`!

IF STATEMENTS

Logical conditions also allow the rules for "what happens next" in a model to be affected by the current values of state variables. The `if` statement lets us do this; the basic format is

```
if(condition);
    commands
else;
    other commands
end;
```

Here's a simple example or two. Code this into MATLAB:

```
x=2
if (x==2);
    disp('OK, x is 2')
else;
    disp('umm, x is not 2')
end;

x=3
if (x<2);
    disp('OK, x is less than 2')
else;
    disp('umm, x is not less than 2
end;
```

You get the picture: the condition is ANYTHING you want that is true or false.

If the "else" is to do nothing, you can leave it out:

```
if(condition);
    commands
end;
```

As in

```
x=3
if (x<2);
    disp('OK, x is less than 2')
end;
```

More complicated decisions can be built up using `elseif`. The basic format for that is

```
if(condition);
   commands
elseif(condition);
   other commands
else;
other commands
end;
```

- Modify your previous program to make a new one: `neural_explosion_2.m` Here are the new DYNAMICS: at each second, test to see if the total number of cells is less than 100. Of so, for every neuron that is already on, two more switch on. If not, for every neuron that is already on, one more switches on. Plot the result as before!

Random numbers

- Here is the MATLAB command to make a single “pseudo” random number: `rand`. Type it and see what you get. Write it down.
- Compare it with what your neighbor got.
- Quit matlab, then restart it. Repeat the above.
- Are you upset with the outcome?
- Repeat this again ... this time, as soon as MATLAB begins, type `rand('state',sum(100*clock))`. That resets the “state” of the random number generator to a unique starting point that has to do with EXACTLY what time it is when you type it in. Thus, you’ll end up with different random numbers each time ... as needed. CONCEPT: ALWAYS ALWAYS ALWAYS use this command before your first use of a random number generator.

Next question – how random ARE those random numbers?

- Here is the MATLAB command to make a vector of n "pseudo" random numbers: `r_vector=rand(1,n).`
  Try it, for $n = 100$.
- Make a plot of these 100 random variables ... on horizontal axis, you should just have the integer 1 through 100. On the vertical, you should have a "*" above each of these numbers, giving the value of the corresponding random number.
- Write down your estimate of how you think these random numbers are "distributed" – what range they cover, with what frequency.
- Type in the code below and save it as `hist_demo.m` What does this code do? (Remember, `help hist` is your friend!) Does it confirm your estimate?

```
M=1000;
sample_list=rand(1,M);
[nlist,centerlist]=hist(sample_list,50);
figure
bar(centerlist,nlist/(M));
```

Coin tossing

- Next, say we want to simulate the tossing of an unfair coin, which comes up heads with probability, or frequency, p (a number between 0 and 1 that gives the fraction of times that a heads occurs).
- Write a for loop with an if statement that turns `r_vector` into vector `heads_and_tails_vector` full of 0's and 1's, where a 1 corresponds to a coin toss that came out heads. Use $p = 0.5$. Then repeat with $p = 0.1$. Do you results make sense? Which corresponds to a "fair" coin toss?

Neuron explosion, continued

- Modify your previous program to make a new one: `neural_explosion_3.m` Here are the new DYNAMICS: at each second, flip a coin with $p = 0.1$. If you get a heads, then the synapse SUCCEEDED IN COMMUNICATING. For every neuron that is already on, two more switch on. If not, the synapse FAILED. For every neuron that is already on, no more switch on. Plot the result as before!