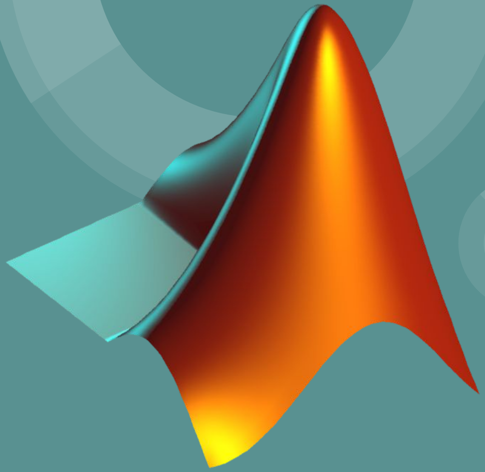
Introduction to MATLAB

July 5th, 2022, 4-5PM @ Hillel House

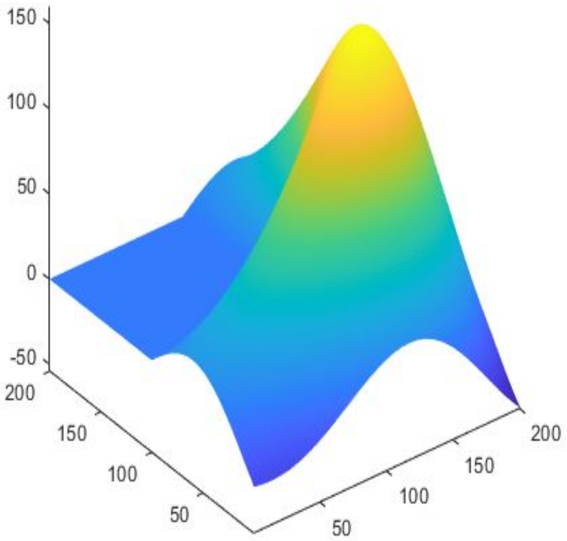
A Software Mini-Series by the
Summer Academic Resource Center

By Giulia Scagliotti and Samuel Low

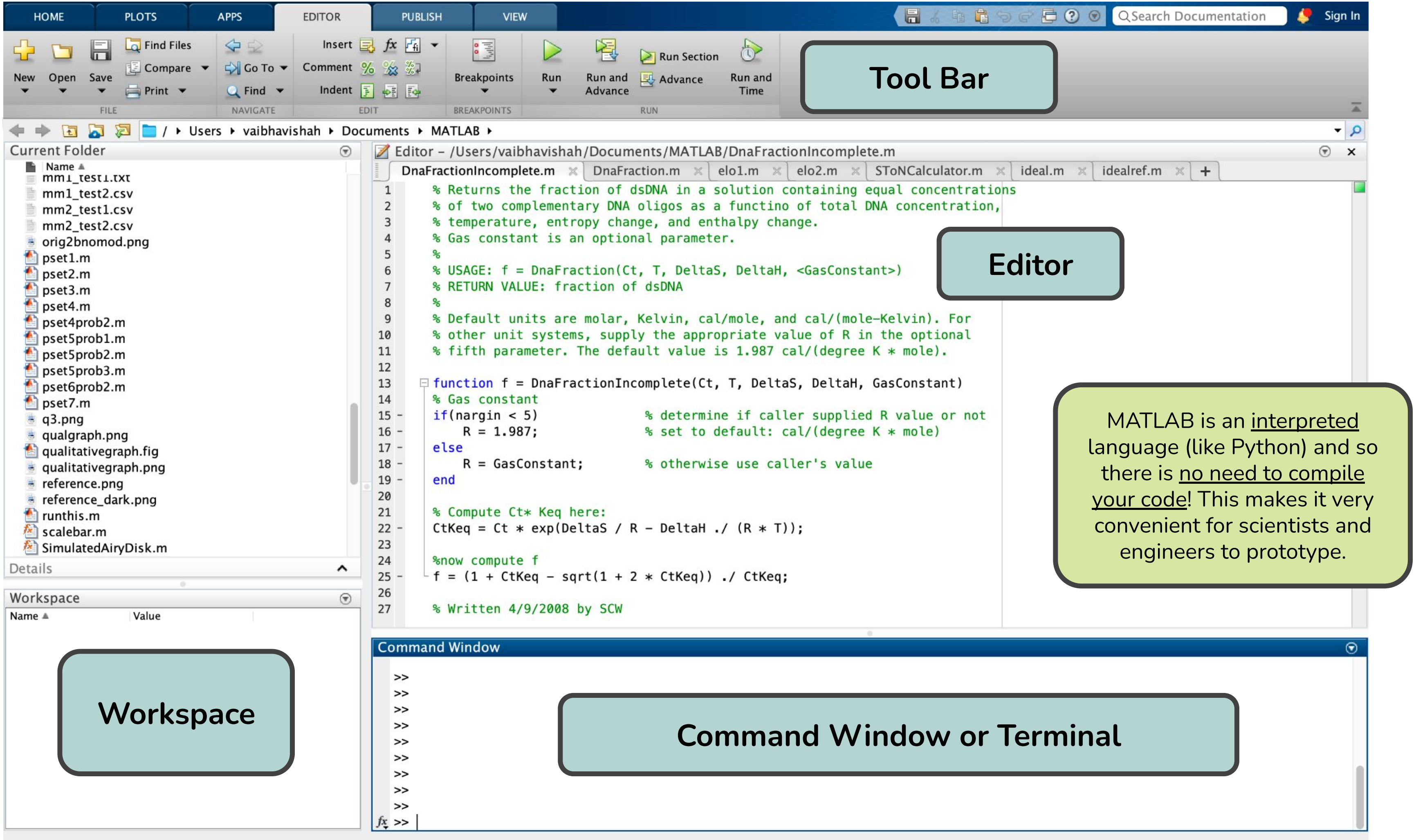


Agenda

1. Initializing and Installation (5 minutes)
2. What is Matlab, and who uses it?
3. Let’s write your first Matlab program!
    a. Variables, Expressions and Operators
    b. Working with Vectors!
    c. Scripts and Functions
    d. Plotting and Visualization
    e. Conditionals and Control Flow



License file: https://www.mathworks.com/academia/tah-portal/stanford-university-30569029.html



mm1_test1.txt
mm1_test2.csv
mm2_test1.csv
mm2_test2.csv
orig2bnomod.png
pset1.m
pset2.m
pset3.m
pset4.m
pset4prob2.m
pset5prob1.m
pset5prob2.m
pset5prob3.m
pset6prob2.m
pset7.m
q3.png
qualgraph.png
qualitativegraph.fig
qualitativegraph.png
reference.png
reference_dark.png
runthis.m
scalebar.m
SimulatedAiryDisk.m

```
% Returns the fraction of dsDNA in a solution containing equal concentrations
% of two complementary DNA oligos as a functino of total DNA concentration,
% temperature, entropy change, and enthalpy change.
% Gas constant is an optional parameter.
%
% USAGE: f = DnaFraction(Ct, T, DeltaS, DeltaH, <GasConstant>)
% RETURN VALUE: fraction of dsDNA
%
% Default units are molar, Kelvin, cal/mole, and cal/(mole-Kelvin). For
% other unit systems, supply the appropriate value of R in the optional
% fifth parameter. The default value is 1.987 cal/(degree K * mole).

function f = DnaFractionIncomplete(Ct, T, DeltaS, DeltaH, GasConstant)
% Gas constant
if(nargin < 5)              % determine if caller supplied R value or not
    R = 1.987;              % set to default: cal/(degree K * mole)
else
    R = GasConstant;        % otherwise use caller's value
end

% Compute Ct* Keq here:
CtKeq = Ct * exp(DeltaS / R - DeltaH ./ (R * T));

%now compute f
f = (1 + CtKeq - sqrt(1 + 2 * CtKeq)) ./ CtKeq;

% Written 4/9/2008 by SCW
```

```
>>
>>
>>
>>
>>
>>
>>
>>
>>
fx >>
```

Tool Bar

Editor

Workspace

Command Window or Terminal

MATLAB is an interpreted language (like Python) and so there is no need to compile your code! This makes it very convenient for scientists and engineers to prototype.

Variables, Expressions, Comments

A **variable** is any symbolic representation for a “value”. It must begin with a letter, but can contain letters, numbers and underscores.

An **assignment** gives the variable its value.

An **expression** is a mathematical statement which evaluates to a value.

A **comment** is a note you can add for yours or others reference, and has no effect whatsoever on your code. To define a comment line use %.

Example Code

```
x1 = 5.71; % This is a comment about my variable!

x2 = [5.71]; % This is a variable with identical value!

x3 = x1 + x2; % This is an expression!

% The ‘=’ is also known as the assignment.
```

Working with Vectors!

Row vectors:

- Use **brackets** with comma or simply space separated elements.
- Use the **colon operator** choosing a beginning value, increment size, end value.
- Use **linspace(start, end,n)**, where n is the number of objects in the vector.

Example Code

```
x4 = [1 2 3] % This creates a vector.

x5 = [1, 2, 3]; % This is an identical vector.

x6 = 1:1:3; % Create a regularly space vector.

% 1:1:3 means create a vector that starts at 1, increases in steps of 1, and ends (inclusively) at the value 3

x7 = linspace(1,3,3); % Linspace can achieve the same vector, except it means to start at 1, end (inclusively) at 3, and the length of the vector is 3.
```

Working with Vectors!

Column vectors:

- Use **brackets** with semicolon separated elements.
- **Transpose** operator acting on row vector.

Strings (vectors of characters):

To specify single pieces of text, such as file names and plot labels

Example Code

```
% First let’s take a look at transposing...

x8 = [1; 2; 3]; % Transposing a vector

x9 = [1 2 3]’; % Also transposing a vector

x10 = transpose(x); % This works too!

% Now let’s take a look at strings.

example1 = ‘Hello world!’;  %This is a char vector

example2 = “Hello world!”;  %This is a string
```

Operators and Basic Functions

Operator symbols: addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^). Prefix the operators with a period (.) for element-wise operations.

Some **built-in functions**:

Trigonometry: sin(), cos(), tan(), atan()

Squares and Exponentials: exp(), log(), sqrt()

Complex numbers: abs(), imag(), real(), conj()

Rounding: round(), ceil(), fix()

Example Code

```
x11 = 4*(x1); % * is an operator for multiplication

x12 = exp(x2); % exp() is a built-in function

% Let’s try out some basic operators with vectors!

x13 = [ 1 4 9 16 25 ];

x14 = sqrt(x13); % This is a square-rooted vector

x15 = x13.^0.5; % This is an identical square root
```

Scripts and Functions

The power of Matlab is the ability to write a complex program which can be run repeatedly.

Scripts and functions store a sequence of Matlab instructions in a .m text file.

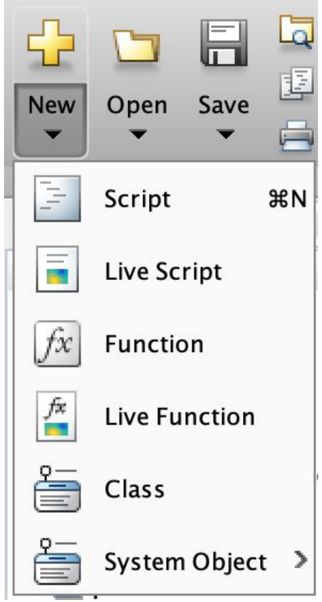
The **script** m-files have no input arguments and will operate on the variables in the workspace.

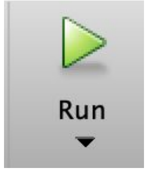
The **function** m-files contain a function definition line, can accept inputs and variables are local to the function.

Scripts and Functions

Create, Save and Run:

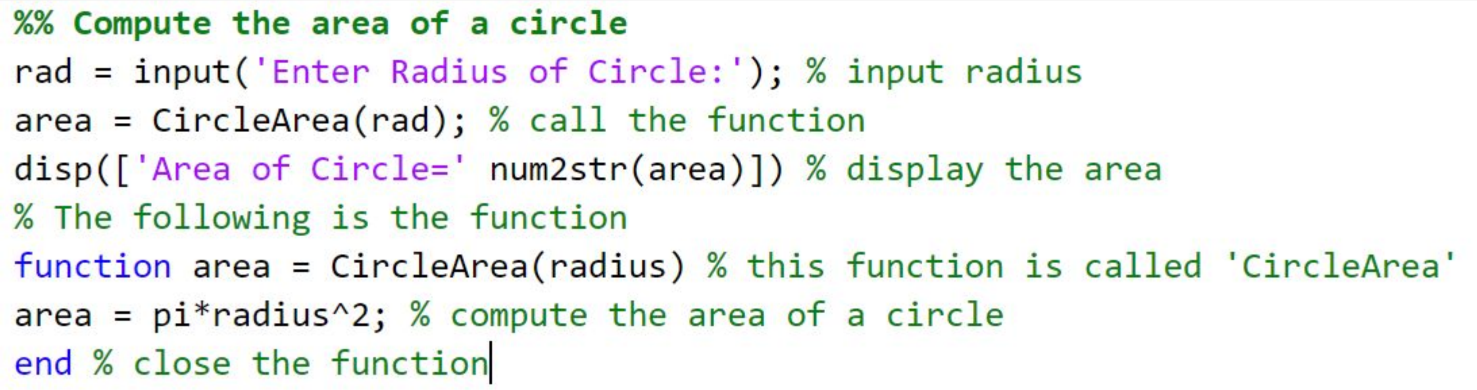
- Create a new script or function
- Save a new script or function
- Run a script
- Run a function: if input arguments are defined, enter <filename>(input arguments) on the command line



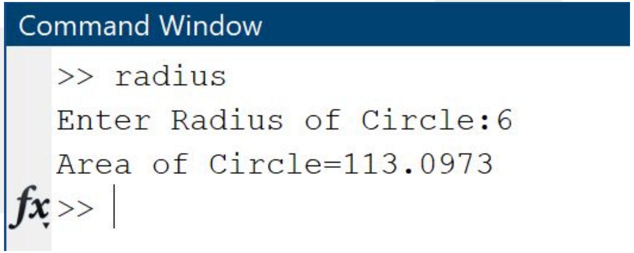


Scripts and Functions

Example Code



```
%% Compute the area of a circle
rad = input('Enter Radius of Circle:'); % input radius
area = CircleArea(rad); % call the function
disp(['Area of Circle=' num2str(area)]) % display the area
% The following is the function
function area = CircleArea(radius) % this function is called 'CircleArea'
area = pi*radius^2; % compute the area of a circle
end % close the function
```



```
>> radius
Enter Radius of Circle:6
Area of Circle=113.0973
>>
```

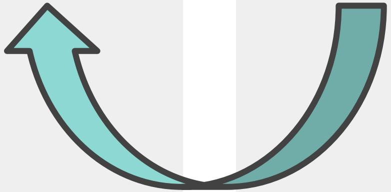
Scripts and Functions

Example script

```
rad = input('Enter radius of circle');
area = CircleArea(rad);
disp([ 'Area of the circle = ' num2str(area) ])
```

Example function

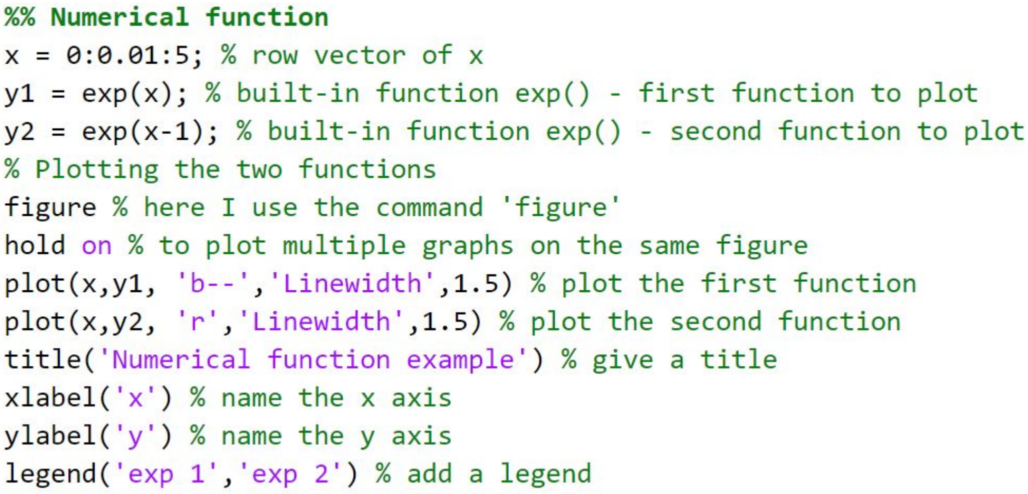
```
function area = CircleArea( radius )
   area = pi*radius.^2;
end
```



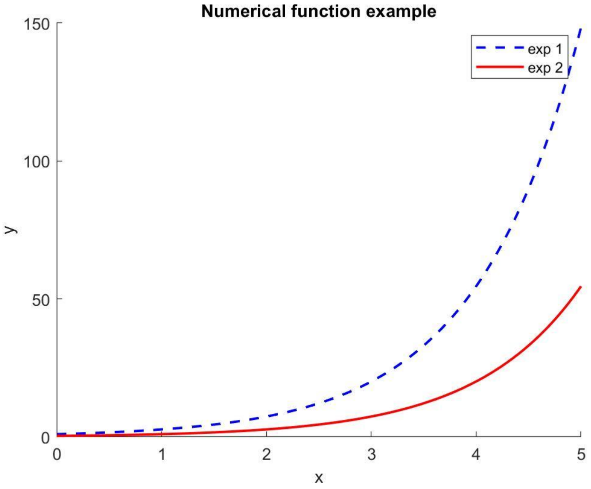
Plotting and Visualization

- You can define a new figure by using the **command ‘figure’**
- Figures can be numbered such as figure(1), figure(2), …
- To plot multiple graphs in the same figure use **‘hold on’**
- **‘xlabel’**, **‘ylabel’**, **‘zlabel’** are used to label the axes
- Numerical functions: use the command **‘plot(x,y)’**
- Other types of plots: bar, scatter, 3D
- Plotting options: ‘LineStyle’, ‘Color’, ‘Linewidth’, ‘MarkerSize’, …

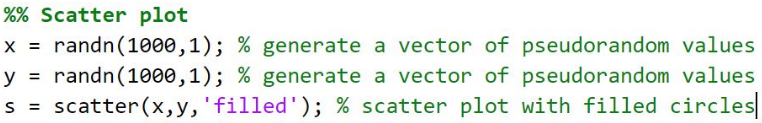
Plotting and Visualization Features



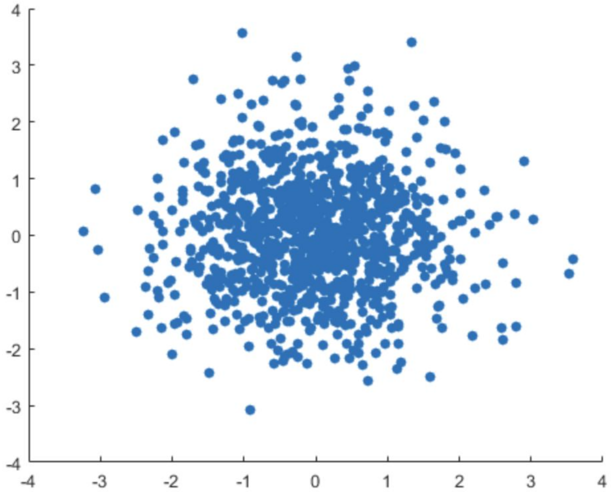
```
%% Numerical function
x = 0:0.01:5; % row vector of x
y1 = exp(x); % built-in function exp() - first function to plot
y2 = exp(x-1); % built-in function exp() - second function to plot
% Plotting the two functions
figure % here I use the command 'figure'
hold on % to plot multiple graphs on the same figure
plot(x,y1, 'b--','Linewidth',1.5) % plot the first function
plot(x,y2, 'r','Linewidth',1.5) % plot the second function
title('Numerical function example') % give a title
xlabel('x') % name the x axis
ylabel('y') % name the y axis
legend('exp 1','exp 2') % add a legend
```

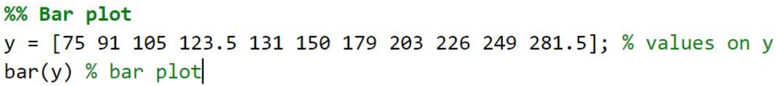


Plotting and Visualization (try this yourself!)

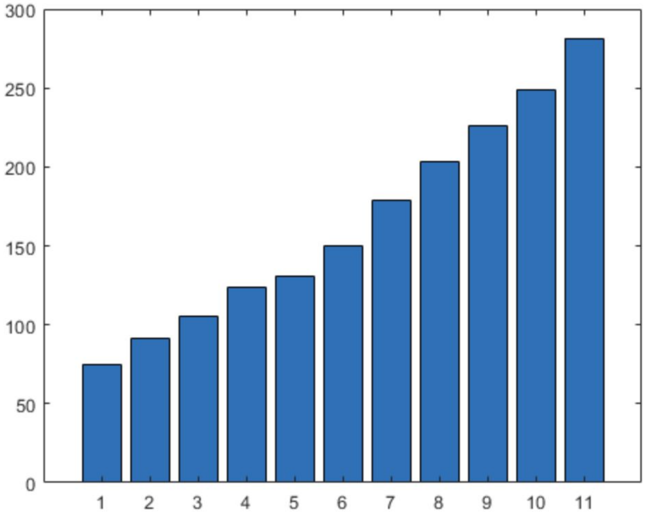


```
%% Scatter plot
x = randn(1000,1); % generate a vector of pseudorandom values
y = randn(1000,1); % generate a vector of pseudorandom values
s = scatter(x,y,'filled'); % scatter plot with filled circles
```





```
%% Bar plot
y = [75 91 105 123.5 131 150 179 203 226 249 281.5]; % values on y
bar(y) % bar plot
```



Getting Help!

- Matlab’s help facilities.

  Command line arguments include:

  help <function name> give a short description of the function

  doc <function name> give more extensive information

  lookfor ‘topic’

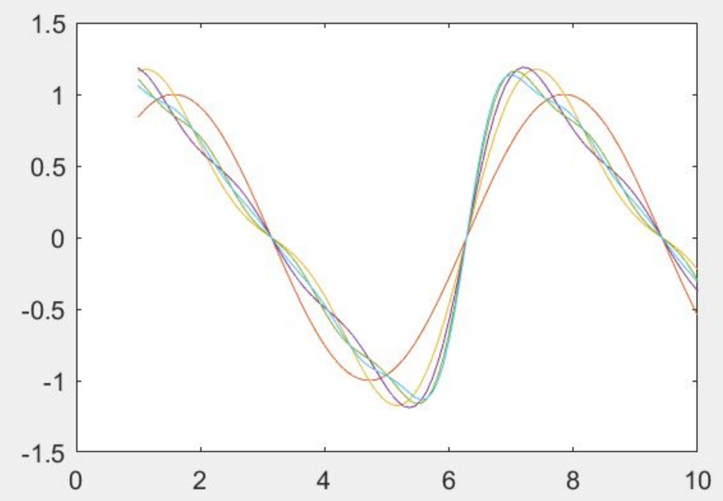
- Online resources

  https://www.mathworks.com/support.html?s_tid=gn_supp

Exercise #1: The Sum of a Signal

Problem Session #1

The key goal of this exercise is simply to learn to add vectors! Let's see what cool things we can plot out! We will learn about "linspace" to generate an x-axis, and in-built functions such as sines and cosines learn about linspace, sine plots, cosine plots, and what happens if you add multiple sines and cosines together (i.e. Fourier series)?

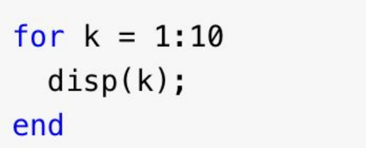


Conditionals and Control Flow

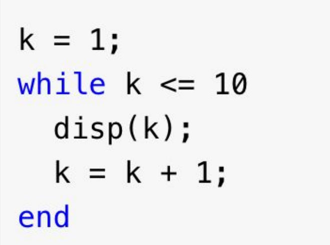
- Loops/iterations of both known (**for**) and indeterminate (**while**) length

- Conditional branching allows different blocks of code to be executed based on conditions determined during the run of the code (**if-elseif-else**)

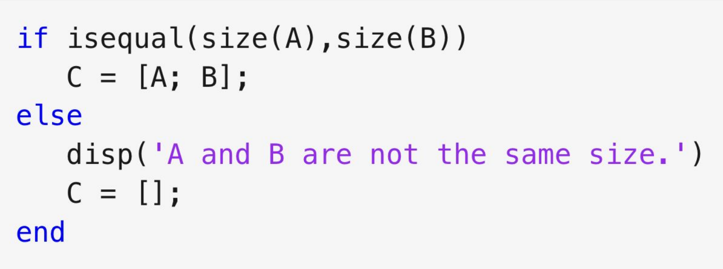
Example Code



```
for k = 1:10
  disp(k);
end
```



```
k = 1;
while k <= 10
  disp(k);
  k = k + 1;
end
```



```
if isequal(size(A),size(B))
   C = [A; B];
else
   disp('A and B are not the same size.')
   C = [];
end
```

Exercise #2: Compare the area of two circles

Problem Session #2

The key goal of this exercise is learn how to write a single function, use that function in a MATLAB script and use conditionals to sort outcomes based on different conditions.

>> Compute the area of two circles, then display if the area of circle 1 is greater than area of circle 2, the two areas are equal or the area of circle 1 is less than the area of circle 2.