Lab Work - Week 1

Math 105B Lab

Summer Session II - 2022

1 MATLAB Basics

1.1 Download and installation

MATLAB is free to install on personal computers for UCI students and faculty, along with many other expensive software suites. To download MATLAB, follow the link https://www.oit.uci.edu/services/end-point-computing/software-hardware-resources/, click software catalogue, log in, find MATLAB in the list, and then follow the instructions on that page.

1.2 Getting Started with MATLAB

MATLAB is a scripting language that allows for us to rapidly prototype algorithms without having to deal with the difficulties of compiled languages. To get started with MATLAB, open up the environment and click 'New', then click 'Script' in the upper left hand corner of the screen. A fresh script window is now open, where you can write and run code. To run the script, click the 'Run' button in the top center of the screen.

Like most computing languages, MATLAB has its own quirks and unique syntax. The first quirk that you will encounter is that arrays are 1-indexed. Meaning, if $v = [10, 20, 30]$ is an array, to access the first entry of this array we need to evaluate v at 1, i.e. $v(1) = 10$.

MATLAB also has its own way of configuring for loops. Below is some sample MATLAB code demonstrating how to use for-loops.

```
%% Demonstration: For-loop functionality in MATLAB
% Always comment your code, for your own sake and the sake of your
% employer!
for i=1:10
  fprintf('We are at index %i in the for loop\n', i);
end
```

The code above defines a variable i then iterates the loop 10 times, where i ranges from 1 to 10. In each instance of the loop, the code prints a string to the MATLAB terminal listing the current index i. Note: *Putting an fprintf command in your for loops is a great way to debug code.*

Conditional logic in MATLAB is similar to other languages, with minor syntactical changes.

```
%% Demonstration: Conditional logic in MATLAB
a = randi(100, 1); % Generate a random positive integer no larger than 100
if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
```

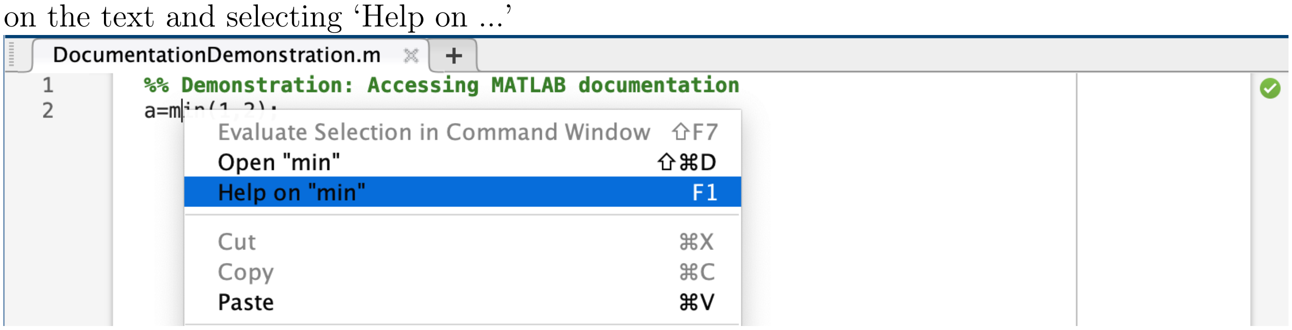
In this class, we will be developing many functions. To create a new function, click ‘New’, then click ‘Function’ in the upper left hand corner of the screen. An example function is below:

```
function [output] = minimum(input1,input2)
%MINIMUM This function computes the minimum of its two inputs.
if input1<input2
    output=input1;
else
    output=input2;
end
end
```

Functions are saved as .m files, similar to scripts, but we must name the function file the same name in the definition of the function. To run this function, if you are navigated to the folder that this function is located, you can type it into the command window. To execute this function in a script, you need to have the function located in the same folder that you have saved your script.

1.3 MATLAB Documentation

MATLAB has countless built in functions at your disposal, so do not reinvent the wheel! In the previous section we created a function which outputs the minimum of two numbers, but MATLAB has already created such a function. If we want to use this function we need to understand its inputs, outputs, and how it works. Thankfully, we do not need to leave the MATLAB environment to do so. If you have written in a function in your code, right clicking on the text and selecting ‘Help on ...’



will open up a new window in MATLAB with detailed documentation on the inputs/outputs of the given function, how it works, and code examples:

If you feel that you need to brush up more on the MATLAB basics, please take a look at https://www.mathworks.com/help/matlab/getting-started-with-matlab.html.

CODING ASSIGNMENT 1 - Writing and testing functions

Objective: produce a function which takes the sum of a given function evaluated at two points. Then test this function using the script MyFirstFunc_Test.m.

Produce a function with the following specifications:

NAME:	myFirstFunc_########
INPUT:	fun, a, b
OUTPUT:	y
DESCRIPTION:	The function computes and outputs the value $y = \text{fun}(a) + \text{fun}(b)$, where a,b are numbers and fun is any function passed to myFirstFunc_########.

IMPORTANT:

- Your function must be spelled exactly correctly, and include your 8-digit student ID at the end of its name. For example, myFirstFunc_12345678.
- The inputs and outputs of your function must appear in the same exact order that is stated in the coding assignment.
- Failure to follow the above requirements will result in your code receiving 0 credit.

EXERCISE

We will be turning these coding assignments. Please see canvas for the list of coding assignments to be collected and their due date. Test your code before submission by performing the following steps:

1. Download the test script MyFirstFunc_Test.m from canvas in the week 1 folder.
2. Place your function and the test script in the same folder.
3. Navigate MATLAB to that folder.
4. Open and run the test script.

2 Lagrange Interpolation

CODING ASSIGNMENT 2 - Implementing Lagrange Interpolation

Produce a function with the following specifications:

NAME:	lagrangePoly_########
INPUT:	X, Y, p
OUTPUT:	z
DESCRIPTION:	z=P(p) is the value of the Lagrange Polynomial P of the function f evaluated at p, where P is defined by the points $(x_1, y_1), \ldots, (x_n, y_n)$. We store the point coordinates in the vectors X=$(x_1, \ldots, x_n)$, Y=$(y_1, \ldots, y_n) = (f(x_1), \ldots, f(x_n))$.
REFERENCE:	Pg. 108. Note that MATLAB indexes vectors starting at 1, so in our case P(x)=$\sum_{k=1}^{n} f(x_k) L_{n-1,k}(x)$, $L_{n-1,k}(x) = \prod_{\substack{i=0,\\ i\neq k}}^{n} \frac{x - x_j}{x_k - x_j}$.

Useful MATLAB concepts/functions

- To find the dimension of a vector, use the length function

  n=length(X)

- To initialize a column u vector of dimension n, use the ones function:

  u=ones(n,1)

EXERCISES

Consider the following MATLAB CODE:

```
% Clear all variables in the workspace
clear variables;
% Clear the command window
clc;

% Create a vector of dimenision n, with entries linearly space between 0.1
% and 4.
n=3;
X=linspace(0.1,4,n);

% Evaluate the function 1/x at each value in the vector X, and store in a
```

```
% new vector Y
Y=1./X;

% Evaluate the Lagrange Polynomial at 100 points linearly spaced between
% 0.1 and 4, store the result in the vector z.
t=linspace(0.1,4,100);
for i=1:length(t)
    z(i)=lagrangePoly_########(X,Y,t(i)); % Replace ######## with student ID
end

% Plot 1/x, the Lagrange Polynomial at each of the points specified by
% the vector t, and the points of intersection specified by (X,Y).
figure(1);
hold on;
plot(t,1./t);
plot(t,z);
plot(X,Y,'.');
```

1. Let $f(x) = 1/x$. Find the difference $f(3) - P(3)$ using $n = 3$, as above.
2. Plot the difference $|f(3) - P(3)|$ with n on the x-axis, using $n = 3, \ldots, 20$.
3. **(Participation Assessment)** Produce a single plot showing the exact function $f(x) = 1/x$ and the Lagrange Interpolants using n=3, 4 and 6.

3 Divided Differences

CODING ASSIGNMENT 3 - Lagrange Interpolation via Divided Differences

---

Produce a function with the following specifications:

NAME:	newtonDivDiff_########
INPUT:	X,Y
OUTPUT:	F
DESCRIPTION:	Given data X=$(x_1, \ldots, x_n)$, Y=$(y_1, \ldots, y_n) = (f(x_1), \ldots, f(x_n))$, this function computes the vector F=$[f[x_0], f[x_0, x_1], \ldots, f[x_0, \ldots, x_n]]$.
PSUEDOCODE:	Pg. 124

Useful MATLAB concepts/functions

- To initialize an n-dimensional matrix A consisting of only zeros, use the command

  A=zeros(n)

- To get the diagonal entries of the matrix A, and arrange them in a column vector v in the same order, use the command:

  v=diag(A)

EXERCISES

1. Using your newtonDivDiff function to interpolate the tabular function given below at the point x=0.05.

x	$f(x)$
0.0	-6.0
0.1	-5.89483
0.3	-5.65014
0.6	-5.17788
1.0	-4.28172

   What is your estimate of the error of using the fourth degree Lagrange interpolating polynomial?

2. Add the point x=1.1, f(1.1)=-3.99583 to the table, and construct the interpolating polynomial of degree 5. Evaluate the Lagrange polynomial at x=1.05. What is your estimate of the error?