BME 313L: INTRODUCTION TO NUMERICAL METHODS IN BIOMEDICAL ENGINEERING

LAB REPORT

Lab #11: Numerical Differentiation

Last name, First name: **Kirk, Andrew**

EID: **alk2488**

Lab Section: Friday

Problem 1.

positionData.txt contains the recorded position data over time from an experiment where an electric car was driven to test it's motors. Copy the code provided in DataImport.m to import the data into MATLAB, units are seconds and meters.

Employ the MATLAB function *diff* to find the speed and acceleration of the car over time.

(1) Plot position vs. time, speed vs. time, and acceleration vs. time in 3 separate graphs

(2) What is the speed and acceleration of the car at 5, 10 and 15 seconds?

Things to discuss: (100 word minimum for each question, 50 word minimum for discussing what you learned, what was reinforced)

(1) What is the difference between `diff` and `gradient`?

(2) What is the relationship between the length of increment and truncation error.

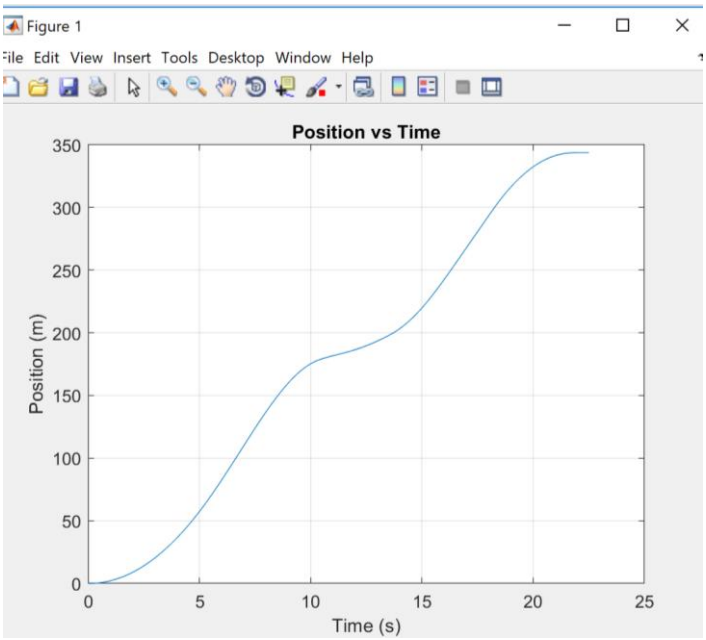
Code

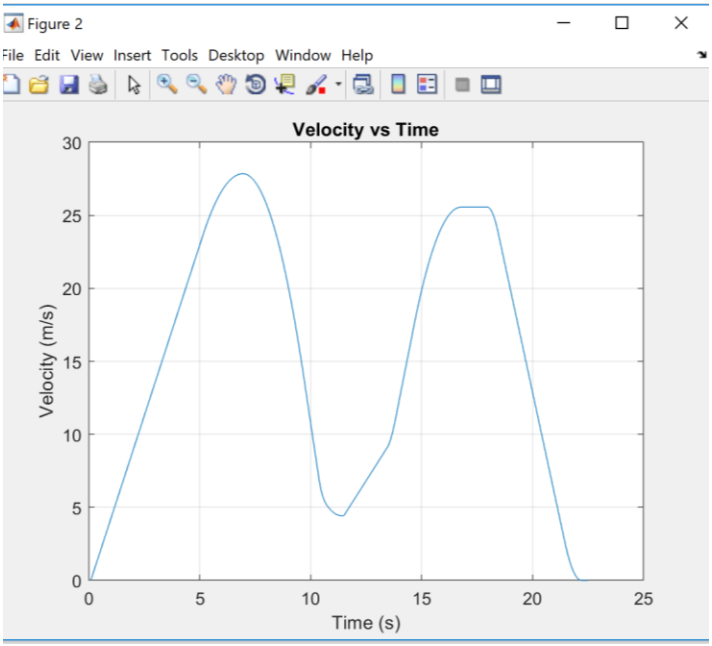
```
clc;
close all;
clear all;

data = dlmread('positionData.txt');
time = data(:,1)    ;
position = data(:,2) ;
vel = diff(position)./diff(time);
acc = diff(vel);

figure(1) % position vs time graph
plot (time,position)
ylabel('Position (m)');
```

```
xlabel('Time (s)');
title('Position vs Time');
grid on

velx = (.1:.1:22.5) ; % velocity x values
figure(2)
plot (velx,vel)
xlabel('Time (s)');
ylabel('Velocity (m/s)');
title('Velocity vs Time');
grid on

figure(3)
accx = (.1:.1:22.4); % acceleration x values
plot (accx,acc)
ylabel('Acceleration (m/s^2)');
xlabel('Time (s)');
title('Acceleration vs Time');
grid on

Speed_5sec = vel(50)
Speed_10sec = vel(100)
Speed_15sec = vel(150)
Acceleration_5sec = acc(50)
Acceleration_10sec = acc(100)
Acceleration_15sec = acc(150)
```

Results

```
Speed_5sec =

   22.9320


Speed_10sec =

   10.8060


Speed_15sec =

   19.8580


Acceleration_5sec =

    0.4680


Acceleration_10sec =

   -1.0180


Acceleration_15sec =

    0.6230
```





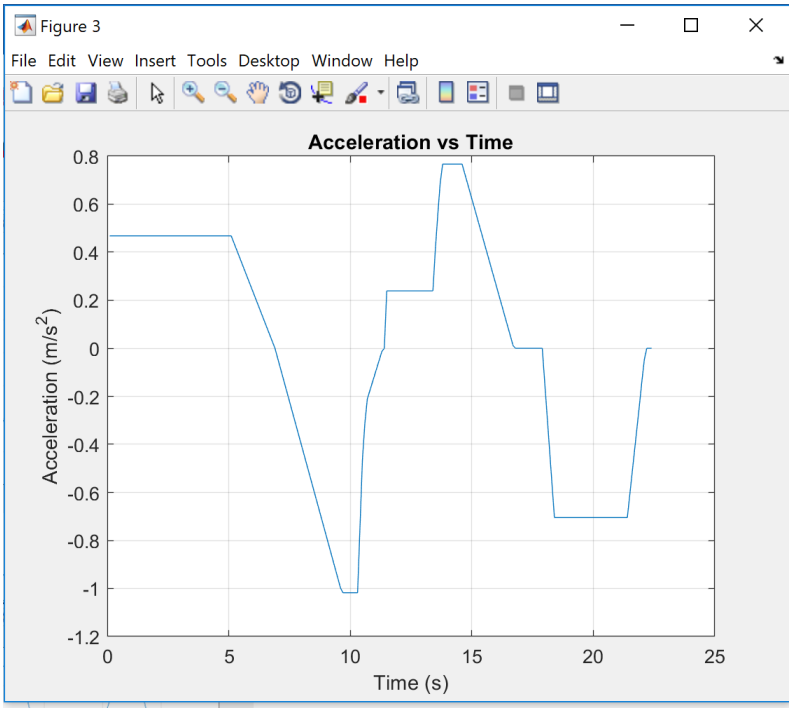


Figure 3
File Edit View Insert Tools Desktop Window Help
Acceleration vs Time
Acceleration (m/s^2)
Time (s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
5
10
15
20
25

Discussion

Diff and gradient will both give you the derivative of the data you are using but they will accomplish this in different ways. Diff just subtracts adjacent x values and outputs that into an array that is 1 smaller than the array of x values. Gradient uses the central difference of adjacent points and because of this, gradient is actually more precise.

Truncation error increases with larger intervals. This is because as you increase the interval size, you lose specificity and you will not be able to notice when the data changes in trend. Due to this, the interval should always remain relatively small so that each change in the trend of the data can be observed.

This problem taught me how to use the diff function for finding the derivatives of values. I also learned how to import data from a txt file.

Problem 2. From textbook problem 21.19

$$f(x) = e^{-2x} - x$$

Calculate derivative of $f(x)$ at x=2 with four different formulas:

(1) Improved forward finite difference approximation $\frac{-f(x_{i+2})+4f(x_{i+1})-3f(x_i)}{2h}$

(2) Improved backward finite difference approximation $\frac{3f(x_i)-4f(x_{i-1})+f(x_{i-2})}{2h}$

(3) Centered finite difference approximation $\frac{f(x_{i+1})-f(x_{i-1})}{2h}$

(4) Improved centered finite difference approximation $\frac{-f(x_{i+2})+8f(x_{i+1})-8f(x_{i-1})+f(x_{i-2})}{12h}$

Change the increment '*h*' from 0.5 to 0.01 with -0.01 intervals (dx=0.5:-0.01:0.01). Generate 4 plots in one graph *(improved forward, improved backward, improved centered, centered finite difference approximation* vs *dx*).

Things to discuss: (100 word minimum for each question, 50 word minimum for discussing what you learned, what was reinforced)

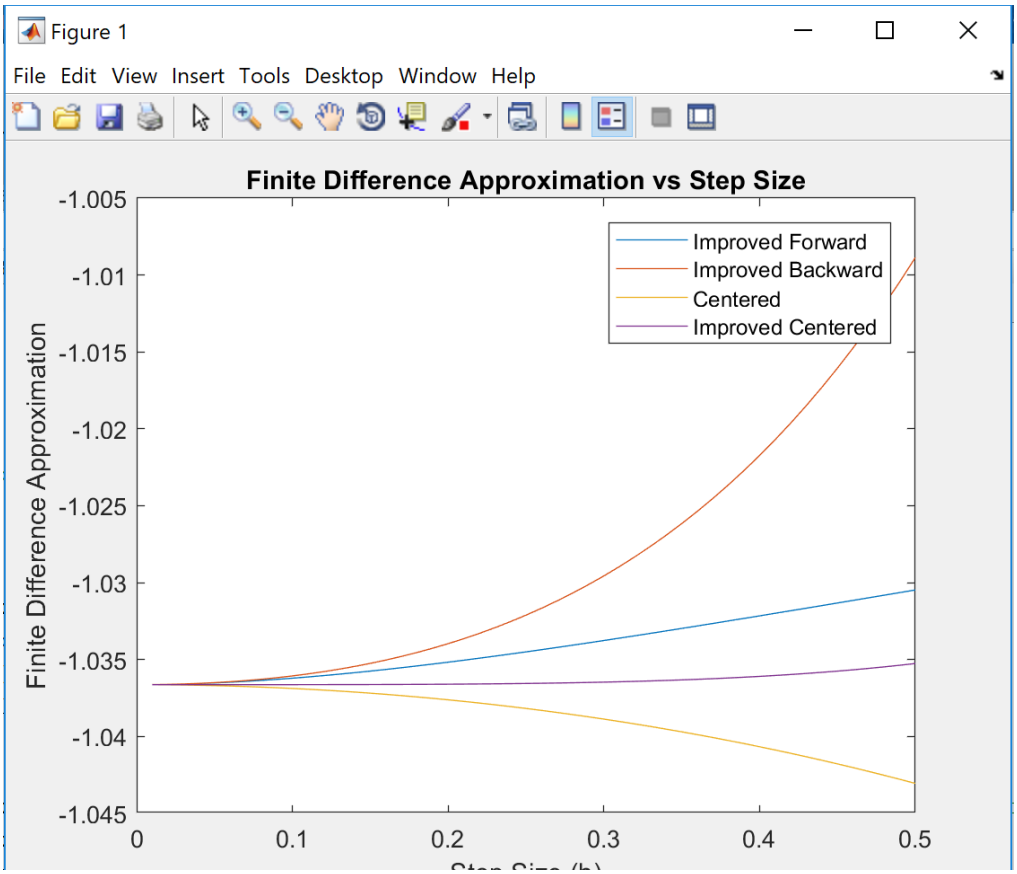
(1) How could we improve these finite difference approximations?

(2) Which finite difference approximation could give you the most accurate estimate?

Code

```
clc;
close all;
clear all;

dx = [0.5:-0.01:0.01]; %given inputs
xvalue = 2;
f=@(x)exp(-2*x)-x;

forward = zeros(length(dx),1); %make array to store calculations
backward = zeros(length(dx),1);
center = zeros(length(dx),1);
improved_center = zeros(length(dx),1);

for i=1:length(dx)
    forward(i,1) = (-1*f(xvalue+2*dx(i))+4*f(xvalue+dx(i))-3*f(xvalue))/(2*dx(i)); %calculate
appproximations
    backward(i,1) = (3*f(xvalue)-4*f(xvalue-dx(i))+f(xvalue-2*dx(i)))/(2*dx(i));
    center(i,1) = (f(xvalue+dx(i))-f(xvalue-dx(i)))/(2*dx(i));
    improved_center(i,1) = (-1*f(xvalue+2*dx(i))+8*f(xvalue+dx(i))-8*f(xvalue-dx(i))+f(xvalue-
2*dx(i)))/(12*dx(i));
end

% Plot
plot(dx,forward,dx,backward,dx,center,dx,improved_center);
ylabel('Finite Difference Approximation');
xlabel('Step Size (h)');
title('Finite Difference Approximation vs Step Size');
legend('Improved Forward','Improved Backward','Centered','Improved Centered');
```

Results



Discussion

To improve the difference approximations it would be beneficial to use taylor series expansion in order to have more points to approximate the difference with. It can be shown from the results that improved center approximation is the best one because it uses this method. This is because it expands the difference calculation allowing for smaller step size and more accuracy in finding the true value.

The best difference approximation is the improved center difference calculation. This is because it utilizes more terms in the taylor series expansion which in turn increases accuracy. In the non improved difference approximations, the center approximation is also the best because it places equal weight on the two data point it is finding the difference for. This is not true for the forward and backward difference approximations.

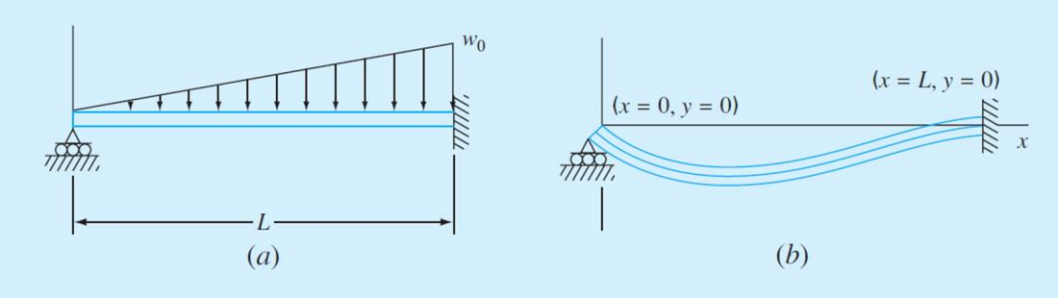
This problem taught me how using taylor series expansion, as seen in the centered improved finite difference calculation, can make the results much more accurate. This is proven by the figure above.

Problem 3 From textbook problem 21.37

The following relationships can be used to analyze uniform beams subject to distributed loads:

$$\frac{dy}{dx} = \theta(x) \quad \frac{d\theta}{dx} = \frac{M(x)}{EI} \quad \frac{dM}{dx} = V(x) \quad \frac{dV}{dx} = -w(x)$$

where x = distance along beam (m), y = deflection (m), $\theta(x)$ = slope (m/m), E = modulus of elasticity (Pa = N/m2), I = moment of inertia (m^4), $M(x)$ = moment (N m), V(x) = shear (N), and $w(x)$ = distributed load (N/m). For the case of a linearly increasing load (Fig. 1).



The slope can be computed analytically as

$$\theta(x) = \frac{w_0}{120EIL}(-5x^4 + 6L^2x^2 - L^4)$$

You measure the following deflections along the length of a simply supported uniform beam

x, m	0	0.375	0.75	1.125	1.5
y, cm	0	−0.2571	−0.9484	−1.9689	−3.2262
x, m	1.875	2.25	2.625	3	
y, cm	−4.6414	−6.1503	−7.7051	−9.275	

Employ numerical differentiation to compute the slope, the moment (in N m), the shear (in N) and the distributed load (in N/m). Use the following parameter values in your computation: E = 200 GPa, and I = 0.0003 m^4.

Things to discuss: (100 word minimum for each question, 50 word minimum for discussing what you learned, what was reinforced)

(1) What kind of finite difference approximation would you use for the data points on boundaries? Why?

Code

```
clc;
close all;
clear all;

x = [0:0.375:3]; %inputs
h = 0.375;
y = [0 -.002571 -.009484 -.019689 -.032262 -.046414 -.061503 -.077051 -.09275];
E = 200*(10^9);
I = 0.0003;

theta = gradient(y,h); % Theta
dydx2 = gradient(theta,h);
moment = dydx2*E*I; % The Moment (Nm)
shear = gradient(moment,h); % Shear (N)
load = -1.*gradient(shear,h); % load (N/m)

% Print results
fprintf('\t\tDist. (m)\t\tDeflection (m)\t\tTheta\t\t\tMoment (Nm)\t\tShear Force (N)\t\tDist. Load (N/m)\n');
for i=1:length(load)
    fprintf('\t\t%7.3f\t\t\t%7.3f\t\t\t    %7.3f\t\t% 10.4d\t\t % 10.4d\t\t    %
10.4d\n',x(i),y(i),theta(i),moment(i),shear(i),load(i));
end
```

Results

Dist. (m)	Deflection (m)	Theta	Moment (Nm)	Shear Force (N)	Dist. Load (N/m)
0.000	0.000	-0.007	-9.2629e+05	-9.3639e+05	-7.4866e+05
0.375	-0.003	-0.013	-1.2774e+06	-6.5564e+05	-1.6978e+06
0.750	-0.009	-0.023	-1.4180e+06	3.3692e+05	-2.1696e+06
1.125	-0.020	-0.030	-1.0247e+06	9.7152e+05	-6.3071e+05
1.500	-0.032	-0.036	-6.8939e+05	8.0996e+05	4.5018e+05
1.875	-0.046	-0.039	-4.1728e+05	6.3388e+05	4.8242e+05
2.250	-0.062	-0.041	-2.1397e+05	4.4814e+05	5.2205e+05
2.625	-0.077	-0.042	-8.1173e+04	2.4235e+05	4.2344e+05
3.000	-0.093	-0.042	-3.2213e+04	1.3056e+05	2.9810e+05

Discussion

On the boundaries it is best to use two different approximations. On the left boundary you should use the forward difference approximation. This is because, the forward difference approximation puts more weight on the data points in front of the x value it is calculation for, which is ideal for the left boundary since there is no data before that point. On the right boundary it is better to use the backward difference approximation. This is because the backward difference approximation places more weight on the data behind the data point you are calculation for which is ideal for the right boundary since there is not data after that point.

This taught me how to use the gradient function and it also showed me how it can be more accurate than the diff function that was previously used.