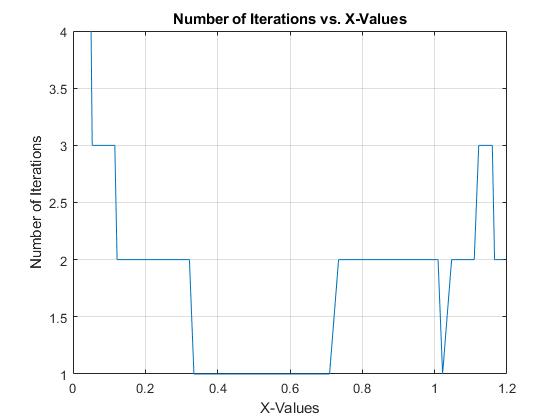
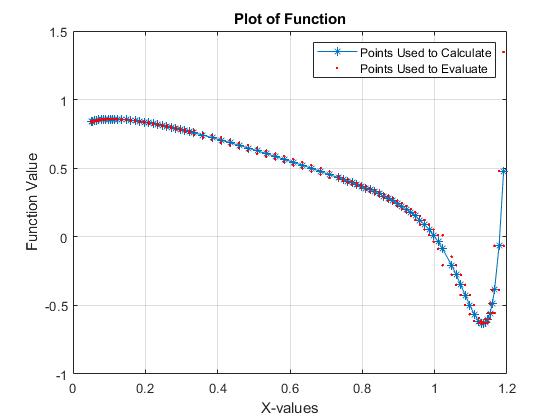
Johnny Donza MEC 320

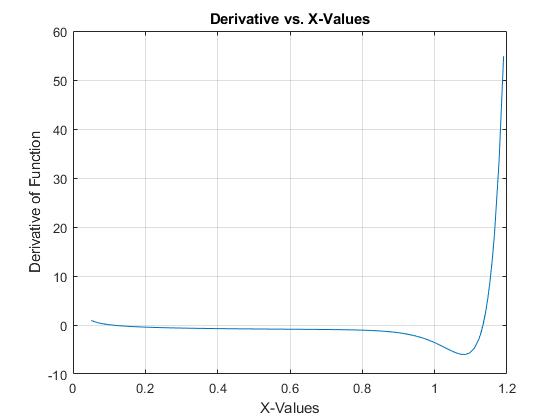
Programming Assignment 4 Matlab

Submission Number: **57a16e95-14c3-4e52-b827-5327f07ff787**

**Task 1:**

1. 



1. Between the x-values of 0 and 0.4, a relatively small step-sized was implemented because this is where we began the shooting method. Therefore, a small step size was required to get precision because we started with an initial guess. From 0.4 to 0.6, a large step size was implemented because there was not a lot of curvature in the function. Then, as the graph approaches x=1, the step size then decreases because of the curvature of the graph. Whenever there is a large amount of curvature in the graph, the step size decreases and vice-versa.
2. 

*Matlab Code:*

close all

clear

clc

%Step 1: Define Anonymous Function

fx = @(x) (x^0.1) \* (1.18-x) \* (1-(exp(20\*(x-1))));

%Step 2: Define Uppper and Lower x bounds

x0 = 0.05;

xf = 1.2;

%Step 3: Define intial h1 and h2 values

h1 = 0.05;

h2 = 0.025;

h = [h1;h2]; %Creates step size vector

%Step 4: Define Error Criterion

Es = 0.1; %Units in percent

%Step 5: Calculate O(h^2) using centered finite difference

k = 1;

j = 0;

Ea = 100;

f = 0; %Sets counter for plot for part B

x= x0;

p = 0;

while x < xf

h = [h1;h2]; %Creates step size vector

p = p+1;

i = 0; %iteration number

Ea = 100;

D(1,1) = (fx(x+h(1,1)) - fx(x-h(1,1)))/(2\*h(1,1));

while Ea > 0.1

i= i+1;

D(i+1,1) = (fx(x+h(i+1,1)) - fx(x-h(i+1,1)))/(2\*h(i+1,1)); %CFD

for k = 2:i+1

j = 2 + i - k;

D(j,k) = (((4^(k-1)) \* (D(j+1,k-1))) - (D(j,k-1))) / (4^(k-1) - 1); %Richardson's Extrapolation

end

Ea(j,1) = abs((D(j,k) - D(j+1,k-1))/ (D(j,k)))\*100; %Calculates Percent Error

if Ea > Es

g = size(h,1);

h(g+1,1) = h(g,1)/2;

end

end

f = f+1;

iter(p) = i;

derv(p) = x; %Counts the x-values (Points where derivative is calculated at)

xii(f) = h(k,1) + derv(p); %Calculates x-value one step size greater than x-value used to calculate derivative

xi(f) = derv(p) - h(k,1); %Calculates x-value one step size smaller than x-value used to calculate derivative

di(f) = D(1,k);

g = size(h,1);

x = x+h(g,1);

end

%Plots for part A

figure (1)

plot(derv,iter);

title 'Number of Iterations vs. X-Values'

xlabel 'X-Values'

ylabel 'Number of Iterations'

grid on

%Plots for part B

[n,m] = size(derv);

for i = 1:m

func(1,i) = fx(derv(1,i));

func1(1,i) = fx(xi(1,i)); %Evaluates function at xi-1

func2(1,i) = fx(xii(1,i)); %Evalutes function at xi+1

end

figure (2)

plot(derv,func,'-\*','MarkerFaceColor','blue');

title 'Plot of Function'

xlabel 'X-values'

ylabel 'Function Value'

grid on

hold on

scatter(derv,func1,'.','red');

hold on

scatter(derv,func2,'.','red');

legend('Points Used to Calculate','Points Used to Evaluate');

%Plot for part D

figure (3)

plot(derv,di);

title 'Derivative vs. X-Values'

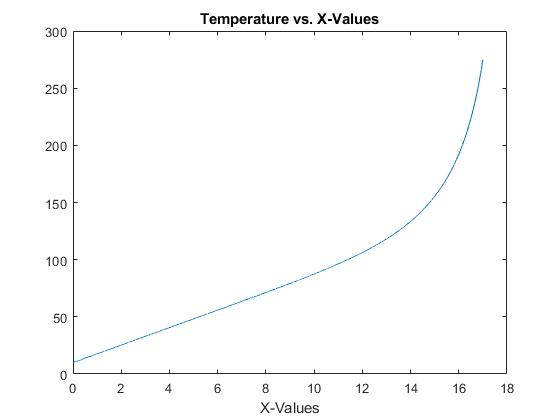
xlabel 'X-Values'

ylabel 'Derivative of Function'

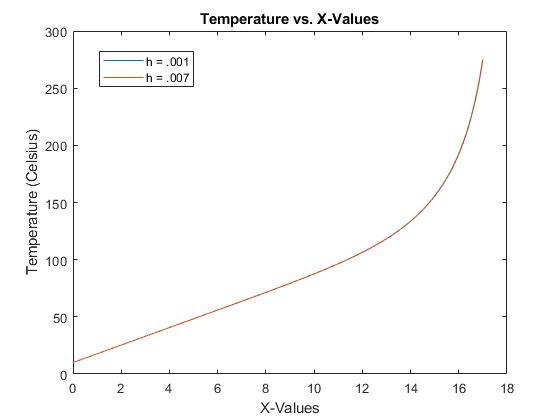
grid on

**Task 2**

1. My final initial z-guess value was 7.6382 and my code performed 13 iterations of the bisection method.



1. The maximum allowable step size was 0.007, which achieved and error of 0.4999%.



As you can see, the plot seems to perfectly overlap the finer step size plot. However, when you zoom in, you actually see that there is in fact a difference where the red line lies slightly above the blue line. This is because the coarser step size is giving us a value for temperature that is slightly larger than the finer step size.

1. Could not get the Runge Kutta to perform. My values kept approaching infinity and I could not figure out why.

*Matlab Code*

close all

clear

clc

%%%%%%%%%%Euler's Method and Bisection Method (Part A)

%Step 1: Define Boundary Conditions

To = 10; %Units of degrees celsius

Tf = 275; %T(17) in units of degress celsius

h\_prime2 = 5.5\*10^-8;

Ta = 30; %Units of degrees celsius

h = 0.001; %Step size for Euler's Method

Es = 0.01; %Units in percent (Stopping Criterion)

Ea = 100; %Units in percent (Initial approximate error)

iter = 17/h; %Number of iterations needed to evaluate over entire interval

x0 = 0; %Starting point on x-axis

zl = 6; %Lower initial guess for Bisection

zu = 8; %Upper initial guess for Bisection

k = 0; %Counter for Bisection Iterations

%Step 2: Define anonymous functions for Euler's Method

Ti = @(T,z) T + z\*h; %Equation 1

zi = @(T,z) z + (h\_prime2\*(T-Ta)^4)\*(h); %Equation 2

%Step 3: Create Loop for Euler's Method

while Ea > Es

%Perform Bisection Method

zr = (zl + zu)/2;

%Perform Euler's Method

for i = 1:iter

if i == 1

x\_a(i,1) = x0;

else

x\_a(i,1) = x\_a(i-1,1) + h;

end

if i == 1

T\_a(i) = (Ti(To,zr));

z\_a(i) = (zi(To,zr));

else

T\_a(i) = (Ti(T\_a(i-1),z\_a(i-1)));

z\_a(i) = (zi(T\_a(i-1),z\_a(i-1)));

end

end

%Continue with Bisection Method

Ea = abs((Tf - T\_a(end))/(Tf))\*100;

g = Tf - T\_a(end);

if g > 0

zl = zr;

elseif g < 0

zu = zr;

else

zroot = zr;

end

k = k+1;

end

figure (1)

plot (x\_a,T\_a);

title 'Temperature vs. X-Values'

xlabel 'X-Values'

ylabel 'Temperature (Celsius)'

hold on

%%%%%%%%%%Begin Part B

zl = 6;

zu = 8;

Ea = 100;

Es = .01;

hnew = input('Coarser Step Size =')

iter1 = 17/hnew;

p = 0;

%Define new anonymous functions

Tii = @(T,z) T + z\*hnew; %Equation 1

zii = @(T,z) z + (h\_prime2\*(T-Ta)^4)\*(hnew); %Equation 2

while Ea > Es

%Perform Bisection Method

zr = (zl + zu)/2;

%Perform Euler's Method

for i = 1:iter1

if i == 1

x\_b(i,1) = x0;

else

x\_b(i,1) = x\_b(i-1,1) + hnew;

end

if i == 1

T\_b(i) = (Tii(To,zr));

z\_b(i) = (zii(To,zr));

else

T\_b(i) = (Tii(T\_b(i-1),z\_b(i-1)));

z\_b(i) = (zii(T\_b(i-1),z\_b(i-1)));

end

end

%Continue with Bisection Method

Ea = abs((Tf - T\_b(end))/(Tf))\*100;

g = Tf - T\_b(end);

if g > 0

zl = zr;

elseif g < 0

zu = zr;

else

zroot = zr;

end

p = p+1;

end

[n,m] = size(T\_b);

[x\_val,indx] = intersect(roundn(x\_a,-5),roundn(x\_b,-5));

T\_d = T\_a(indx);

for s = 1:m

E(s) = abs((T\_d(s) - T\_b(s))/(T\_d(s)))\*100;

end

w = max(E);

disp('Maximum Error =');

disp(w);

if w > 0.5

disp('Use smaller step size')

elseif w < 0.5

disp('Use greater step size')

else

disp('Step size achieved error of 0.5% exactly')

end

figure (1)

plot (x\_b,T\_b);

%%%%%%%%%%Begin Part C

%Step 2: Define New Constants

%Step 1: Define K-functions

Ea\_c = 100; %Units in percent

Es\_c = .01; %Units in percent

z(1) = 6;

while Ea\_c > Es\_c

for i = 1:iter

if i == 1

x\_c(i,1) = x0;

else

x\_c(i,1) = x\_c(i-1,1) + h;

end

if i == 1

k1\_T(i) = Ti(To,z);%K11

k1\_z(i) = zi(To,z);%K12

a = To + ((k1\_T(i))\*(h/2));

b = z + ((k1\_z(i))\*(h/2));

k2\_T(i) = Ti(a,b);

k2\_z(i) = zi(a,b);

c = To + ((k2\_T(i))\*(h/2));

d = z + ((k2\_z(i))\*(h/2));

k3\_T(i) = Ti(c,d);

k3\_z(i) = zi(c,d);

e = To + ((k3\_T(i))\*h);

f = z + ((k3\_z(i))\*h);

k4\_T(i) = Ti(e,f);

k4\_z(i) = zi(e,f);

T\_c (i) = To + ((1/6) \* (k1\_T(i) + 2\*k2\_T(i) + 2\*k3\_T(i) + k4\_T(i)))\*h;

z\_c(i) = z + ((1/6) \* (k1\_z(i) + (2\*k2\_z(i)) + (2\*k3\_z(i)) +k4\_z(i)))\*h;

else

k1\_T(i) = Ti(T\_c(i-1),z\_c(i-1));%K11

k1\_z(i) = zi(T\_c(i-1),z\_c(i-1));%K12

a(i) = T\_c(i-1) + ((k1\_T(i))\*(h/2));

b(i) = z\_c(i-1) + ((k1\_z(i))\*(h/2));

k2\_T(i) = Ti(a(i),b(i));

k2\_z(i) = zi(a(i),b(i));

c(i) = T\_c(i-1) + ((k2\_T(i))\*(h/2));

d(i) = z\_c(i-1) + ((k2\_z(i))\*(h/2));

k3\_T(i) = Ti(c(i),d(i));

k3\_z(i) = zi(c(i),d(i));

e(i) = T\_c(i-1) + ((k3\_T(i))\*h);

f(i) = z\_c(i-1) + ((k3\_z(i))\*h);

k4\_T(i) = Ti(e(i),f(i));

k4\_z(i) = zi(e(i),f(i));

T\_c (i) = T\_c(i-1) + ((1/6) \* (k1\_T(i) + (2\*k2\_T(i)) + (2\*k3\_T(i)) +k4\_T(i)))\*h;

z\_c(i) = z\_c(i-1) + ((1/6) \* (k1\_z(i) + (2\*k2\_z(i)) + (2\*k3\_z(i)) +k4\_z(i)))\*h;

end

end

Ea\_c(i) = abs((Tf - T\_c(end))/(Tf))\*100;

g = Tf - T\_c(end);

end

Ea\_c = 100; %Units in percent

Es\_c = .01; %Units in percent

z(1) = 8;

while Ea\_c > Es\_c

for i = 1:iter

if i == 1

x\_c(i,1) = x0;

else

x\_c(i,1) = x\_c(i-1,1) + h;

end

if i == 1

k1\_T(i) = Ti(To,z);%K11

k1\_z(i) = zi(To,z);%K12

a = To + ((k1\_T(i))\*(h/2));

b = z + ((k1\_z(i))\*(h/2));

k2\_T(i) = Ti(a,b);

k2\_z(i) = zi(a,b);

c = To + ((k2\_T(i))\*(h/2));

d = z + ((k2\_z(i))\*(h/2));

k3\_T(i) = Ti(c,d);

k3\_z(i) = zi(c,d);

e = To + ((k3\_T(i))\*h);

f = z + ((k3\_z(i))\*h);

k4\_T(i) = Ti(e,f);

k4\_z(i) = zi(e,f);

T\_c (i) = To + ((1/6) \* (k1\_T(i) + 2\*k2\_T(i) + 2\*k3\_T(i) + k4\_T(i)))\*h;

z\_c(i) = z + ((1/6) \* (k1\_z(i) + (2\*k2\_z(i)) + (2\*k3\_z(i)) +k4\_z(i)))\*h;

else

k1\_T(i) = Ti(T\_c(i-1),z\_c(i-1));%K11

k1\_z(i) = zi(T\_c(i-1),z\_c(i-1));%K12

a(i) = T\_c(i-1) + ((k1\_T(i))\*(h/2));

b(i) = z\_c(i-1) + ((k1\_z(i))\*(h/2));

k2\_T(i) = Ti(a(i),b(i));

k2\_z(i) = zi(a(i),b(i));

c(i) = T\_c(i-1) + ((k2\_T(i))\*(h/2));

d(i) = z\_c(i-1) + ((k2\_z(i))\*(h/2));

k3\_T(i) = Ti(c(i),d(i));

k3\_z(i) = zi(c(i),d(i));

e(i) = T\_c(i-1) + ((k3\_T(i))\*h);

f(i) = z\_c(i-1) + ((k3\_z(i))\*h);

k4\_T(i) = Ti(e(i),f(i));

k4\_z(i) = zi(e(i),f(i));

T\_c (i) = T\_c(i-1) + ((1/6) \* (k1\_T(i) + (2\*k2\_T(i)) + (2\*k3\_T(i)) +k4\_T(i)))\*h;

z\_c(i) = z\_c(i-1) + ((1/6) \* (k1\_z(i) + (2\*k2\_z(i)) + (2\*k3\_z(i)) +k4\_z(i)))\*h;

end

end

Ea\_c(i) = abs((Tf - T\_c(end))/(Tf))\*100;

g = Tf - T\_c(end);

end