1. Initially when I answered this question, I set my for loop to 300. This gave me such a small error that I had to back track to find an error that wasn’t 0. Eventually I found that 16 iterations gave me 10-6 accuracy so 17 was used

x1 = zeros(1, 301);

x2 = zeros(1, 301);

a = 0

for i = 1:17

J = [2\*exp(2\*x1(i)-x2(i))-1 -exp(2\*x1(i)-x2(i)) ; 2\*x1(i) -1];

A = [exp(2\*x1(i)-x2(i))-x1(i) ; x1(i)^2-x2(i)];

%equivalent to inv(J) \* A

J2 = J\A;

x1(i + 1) = x1(i) - J2(1);

x2(i + 1) = x2(i) - J2(2);

a = i + 1;

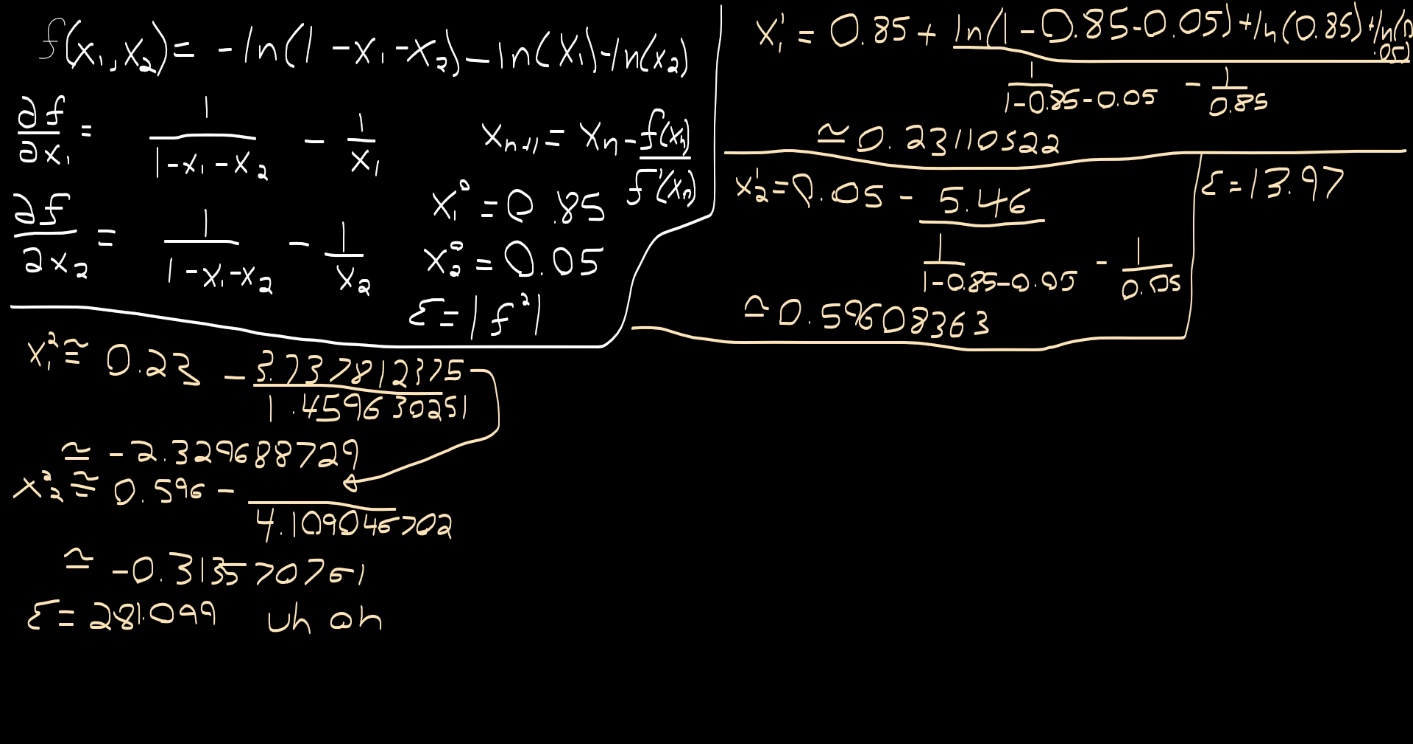
end

x1(a)

x2(a)

A(1)^2 + A(2)^2

1. This series diverges, it becomes apparent as the first error is 13 and the second is 180. I preformed Newtons method until I encountered said problem, I then coded the problem to ensure my results. In matlab I hit x1 = infinity + infinity i.



x1 = zeros(1, 301);

x2 = zeros(1, 301);

a = 0;

x1(1) = 0.85;

x2(1) = 0.05;

for i = 1:100

f = -log(1 - x1(i) - x2(i)) - log(x1(i)) - log(x2(i));

fx1 = 1/(1-x1(i) - x2(i)) - 1/x1(i);

fx2 = 1/(1-x1(i) - x2(i)) - 1/x2(i);

x1(i + 1) = x1(i) - (f/fx1);

x2(i + 1) = x2(i) - (f/fx2);

x1(i + 1)

x2(i + 1)

end

final\_answer = f^2

I later redid this problem by hand. x1 and x2 must be greater than 0 as ln(0) = error. x1 and x2 must be smaller than 1 when combine as ln(1 – 0.5 - 0.5) = error. With these bounds in mind I found that when x1 and x2 were close to 0 the equation spit out a larger number than when x1 and x2 were closer to 1, therefore the answer to the equation by hand is (x1 + x2)/2 = 0.9 giving a value greater than the minimum of 1.6936

Therefore the minimum must be greater than 1 and less than 1.6936



function answer = runge\_kutta(start\_point, end\_point, initial\_value ...

, number\_of\_steps)

t = start\_point;

h = end\_point/number\_of\_steps;

w = initial\_value;

n = number\_of\_steps;

for i = 1:n

k1 = h\*f(t,w);

k2 = h\*f(t+h/2, w+k1/2);

k3 = h\*f(t+h/2, w+k2/2);

k4 = h\*f(t+h, w+k3);

w = w + (k1+2\*k2+2\*k3+k4)/6;

t = t + h;

end %for

answer = w;

end %Runge\_Kutta

function answer = f(t, y)

answer = y^2 + sin(t);

end

The closest I could get to the correct answer was with 100000 steps, a step size of 2.000000000000000e-05 and a change in y(2) of 5.080665869172662e-05, y(2) was: 1.712482504716393e+04

1. t is never used in f’(t) = ycos(y) however I don’t believe this matters as h and t can be kept track of throughout the function.

My final answer was: 1.399002690485886

My code follows:

eulers(0,1,1,10000)

function answer = eulers(start\_point, end\_point, initial\_value ...

,number\_of\_itterations)

t = start\_point;

y = initial\_value;

h = end\_point/number\_of\_itterations;

for i = 1:number\_of\_itterations

y = y + h\*f(t,y);

t = t + h;

end %for

answer = y;

end %eulers

function answer = f(x, y)

answer = y\*cos(y);

end %f(x,y)