Logo, company name

Description automatically generated

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING

ME 310 – NUMERICAL METHODS

SPRING 2022

HOMEWORK 2

Muhammed Rüstem ŞEVİK – 2446888

I understand that this is an individual assignment. I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own.

Table of Contents

[1. Introduction 3](#_Toc131872307)

[2. Body 3](#_Toc131872308)

[Question 1 3](#_Toc131872309)

[Question 7 5](#_Toc131872310)

[Question 9 9](#_Toc131872311)

[4. Discussion 11](#_Toc131872312)

[5. Conclusion 11](#_Toc131872313)

List of Figures

Figure 1: Scatter plot of iterations for false position method 6

Figure 2: Scatter plot of iterations for secant method 7

Figure 3: Surfaces and and the line of intersection 9

Figure 4: Line of intersection of surfaces 9

List of Tables

Table 1: Iteration details for false position method, = 0.5, = 5 5

Table 2: Iteration details for secant method, = 0.5, = 5 6

Table 3: Iteration details for false position method, = 5, = 0.5 6

# 1. Introduction

In this homework assignment, students are expected to enhance their MATLAB skills and learn various numerical root finding algorithms by constructing a hybrid root finding algorithm in question one, by using secant and false position method in question seven and roots of nonlinear systems in question nine.

# 2. Body

### Question 1

In this question students are expected to write a hybrid root finding algorithm similar to the “fzero” function of MATLAB. The question asks for creation of the root finding interval and after that usage of modified secant method and bisection method in a hybrid way.

Bisection Method formula.

|  |  |
| --- | --- |
|  | (1) |

Modified Secant Method formula.

|  |  |
| --- | --- |
|  | (2) |

Note that in the Phase 1 code works like the output of Run 4 not as described in the question. Also, since the original code is updated some of these test results are not same as the given output.

Code:

clc; clear;

f = @(x) sqrt(9.81\*x/0.25) \* tanh(sqrt(9.81\*0.25/x)\*4) - 35;

[root, iterations] = findRoot(f, 0, 1e-10, 1000);

function [root, iter] = findRoot(f, x0, tol, maxIter)

% Default value for maximum iteration

if nargin < 4

maxIter = 1000;

end

% Phase 1 (searching for a bracketing interval):

s\_iter = 1;

if isscalar(x0)

fprintf("<strong> Trial xL xU " + ...

" f(xL) f(xU) Method </strong> \n")

% initialize search width

delta = x0/50;

if delta == 0

delta = 1/50;

end

delta\_0 = delta;

% search for bracketing interval

sign\_change = false;

while ~sign\_change % && delta < abs(x0)

xL = x0 - delta;

xU = x0 + delta;

sign\_change = sign(f(xL)) ~= sign(f(xU));

fprintf(' %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', ...

s\_iter, xL, xU, f(xL), f(xU), "Search")

delta = delta\_0 + 2 \* delta;

s\_iter = s\_iter + 1;

if s\_iter == 1001

break

end

end

% check if bracketing interval was found

if ~sign\_change

error('Unable to find bracketing interval.');

end

else

% use provided interval as bracketing interval

xL = x0(1);

xU = x0(2);

% check if bracketing interval is valid

if sign(f(xL)) == sign(f(xU))

error('The function does not change sign in the provided interval.');

end

end

% Phase 2 (finding the root):

fprintf("<strong> Iter\t xL\t\t xU\t\t root " + ...

" f(root) Method </strong> \n")

iter = 1;

pert = (xU - xL) / 100;

xnew = (xL + xU) / 2; % initial guess being the mid-point of the interval

while iter < maxIter

% fist use modified secant method (MSM)

if abs(f(xnew)) < tol

root = xnew;

return

end

x\_candidate = xnew - (f(xnew) \* pert) / (f(xnew + pert) - f(xnew));

if xL <= x\_candidate && x\_candidate <= xU

xnew = x\_candidate;

fprintf(' %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n',...

iter, xL, xU, xnew, f(xnew), "Modified secant")

else % in here we have to swith to BM

fprintf(' \t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', xL,...

xU, x\_candidate, f(x\_candidate), "Modified secant (failed)")

xnew = (xU + xL) / 2;

fprintf(' %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', iter,...

xL, xU, xnew, f(xnew), "Bisection")

if f(xL) \* f(xnew) < 0

xU = xnew;

xnew = 0.5\*(xL+xnew);

elseif f(xnew) \* f(xU) < 0

xL = xnew;

xnew = 0.5\*(xnew+xU);

end

end

% increment iterations

iter = iter + 1;

end

% maxIter reached without finding root

error('Maximum iterations reached without finding root.');

end

### 

### Question 7

|  |  |
| --- | --- |
|  | (3) |

False Position Method formula.

|  |  |
| --- | --- |
|  | (4) |

Secant Method formula.

|  |  |
| --- | --- |
|  | (5) |
|  |  |

Error definition.

|  |  |
| --- | --- |
|  |  |
|  | (6) |
|  |  |

Calculation Details

False Position Method

Table : Iteration details for false position method, = 0.5, = 5

|  |  |  |  |
| --- | --- | --- | --- |
| Iterations | x | f(x) |  |
| 1 | 1.85463498 | 0.61768790 | - |
| 2 | 1.21630782 | 0.19581989 | 52.48072505 |
| 3 | 1.05852096 | 0.05687262 | 14.90635156 |
| 4 | 1.01616935 | 0.01604002 | 4.16777111 |

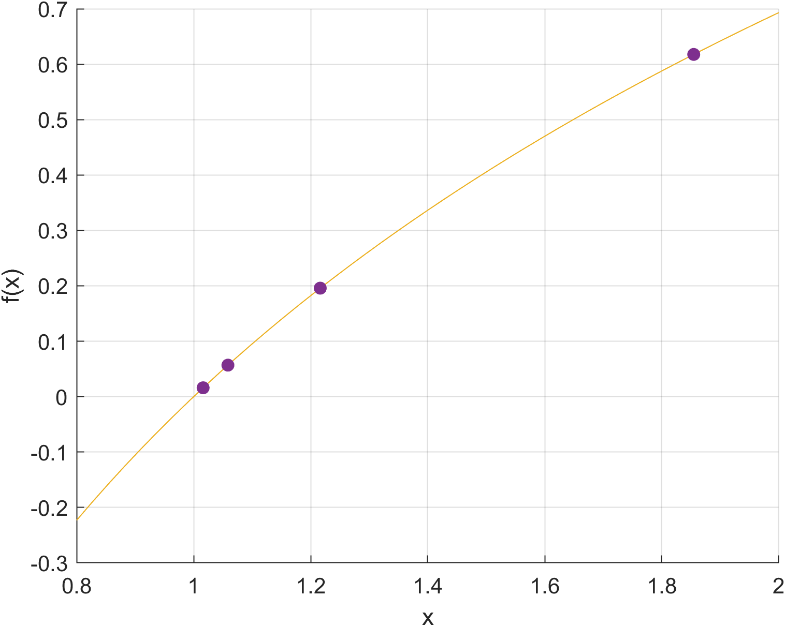


Figure : Scatter plot of iterations for false position method

Secant Method

Table : Iteration details for secant method, = 0.5, = 5

|  |  |  |  |
| --- | --- | --- | --- |
| Iterations | x | f(x) |  |
| 1 | 1.85463498 | 0.61768790 | - |

No plot here because the result is not successful. The method gave a negative number. The function ln(x) is not defined in the negative numbers. So, the process is terminated.

False Position Method

Table : Iteration details for false position method, = 5, = 0.5

|  |  |  |  |
| --- | --- | --- | --- |
| Iterations | x | f(x) |  |
| 1 | 1.85463498 | 0.61768790 | - |
| 2 | 1.21630782 | 0.19581989 | 52.48072505 |
| 3 | 0.92001335 | -0.08336709 | 32.20545256 |
| 4 | 1.00848885 | 0.00845303 | 8.77307671 |

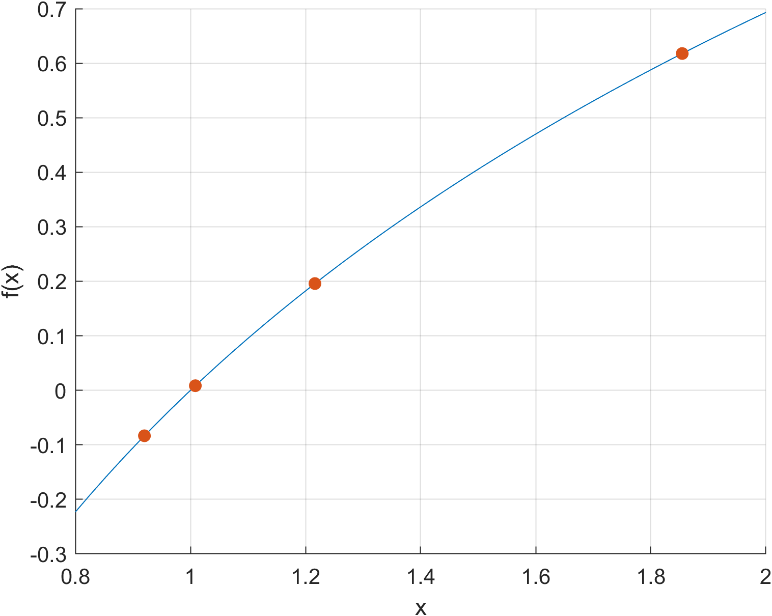


Figure : Scatter plot of iterations for secant method

Code:

clc; clear;

f = @(x) log(x);

false\_position(f, 0.5, 5, 0, 4)

secant(f, 0.5, 5, 0, 4)

secant(f, 5, 0.5, 0, 4)

function false\_position(f, x0, x1, e, i)

fprintf('\n\*\*\* FALSE POSITION METHOD \*\*\*\n');

step = 1;

condition = true;

previous\_xr = 0;

error = 100 \* e;

while condition

xm = (x0 \* f(x1) - x1 \* f(x0)) / (f(x1) - f(x0));

if xm ~= 0

error = abs((previous\_xr - xm) / xm \* 100);

condition = error > e;

end

if xm <= 0 || x0 <= 0 || x1 <= 0

fprintf("Negative Value in ln")

return

end

f\_xm = f(xm);

fprintf(['Iteration-%d, xm = %0.8f , f(xm) = %0.8f , ' ...

'error = %0.8f\n'], step, xm, f\_xm, error);

if f(x0) \* f(xm) < 0

x1 = xm;

elseif f(x0) \* f(xm) > 0

x0 = xm;

else

condition = false;

end

previous\_xr = xm;

step = step + 1;

if step - 1 == i

break;

end

end

fprintf(['Last Estimate is : %0.8f, Approximate Percent Relative ' ...

'Error : %.8f, Number of Iterations : %d , Function Value ' ...

': %0.8f\n'], xm, error, step-1, f\_xm);

end

function secant(f, xs\_old, x\_s, e, i)

fprintf('\n\*\*\* SECANT METHOD \*\*\*\n');

step = 1;

condition = true;

error = 1 + e;

while condition

x\_i = x\_s - (f(x\_s) \* (xs\_old - x\_s)) / (f(xs\_old) - f(x\_s));

if x\_i <= 0 || x\_s <= 0 || xs\_old <= 0

fprintf("Negative Value in ln")

return

end

if x\_i ~= 0

error = abs((x\_i - x\_s) / x\_i \* 100);

condition = error > e;

end

f\_x\_i = f(x\_i);

fprintf(['Iteration-%d, xm = %0.8f and f(xm) = %0.8f , ' ...

'error = %0.8f\n'], step, x\_i, f\_x\_i, error);

xs\_old = x\_s;

x\_s = x\_i;

step = step + 1;

if step -1 == i

break;

end

end

fprintf(['Last Estimate is : %0.8f, Approximate Percent Relative ' ...

'Error : %.8f, Number of Iterations : %d , Function Value : ' ...

'%0.8f\n'], x\_s, error, step-1, f\_x\_i);

end

x = [1.85463498, 1.21630782, 1.05852096, 1.01616935];

y = [0.61768790, 0.19581989, 0.05687262, 0.01604002];

x = [1.85463498, 1.21630782, 0.92001335, 1.00848885];

y = [0.61768790, 0.19581989, -0.08336709, 0.00845303];

f = @(x) log(x);

hold on

grid on

fplot(f, [0.8 2])

scatter(x, y, "filled")

ylabel("f(x)"); xlabel("x")

exportgraphics(gca, "7c.png", "Resolution",300)

### Question 9

Rearrange given parametric functions.

|  |  |
| --- | --- |
|  | (7) |
|  | (8) |
|  | (9) |
|  | (10) |
|  | (11) |
|  | (12) |

Plot the functions and to obtain the intersection.

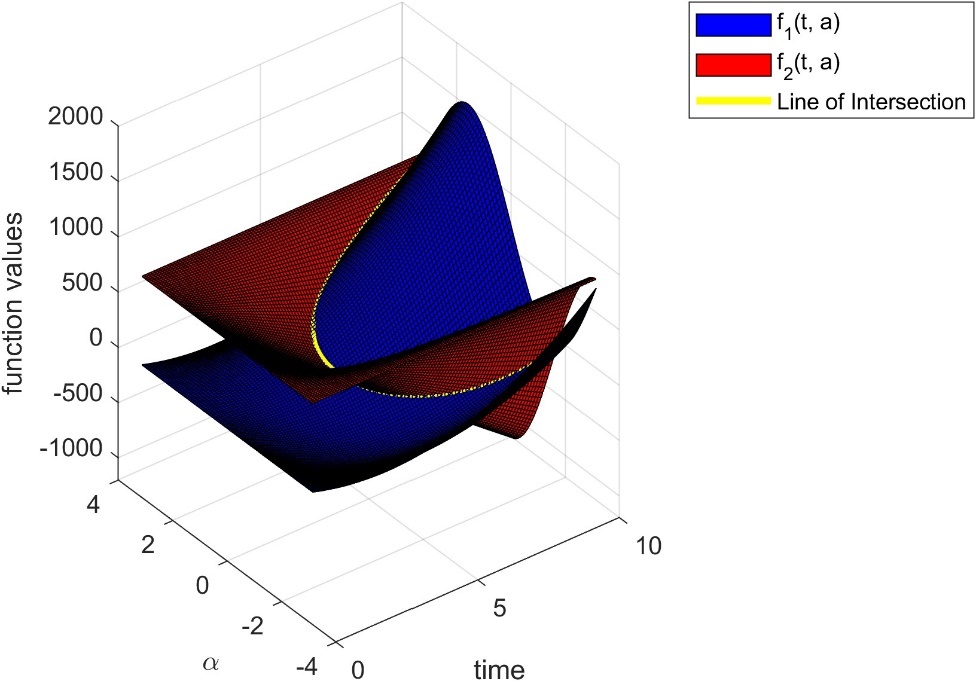


Figure : Surfaces and and the line of intersection

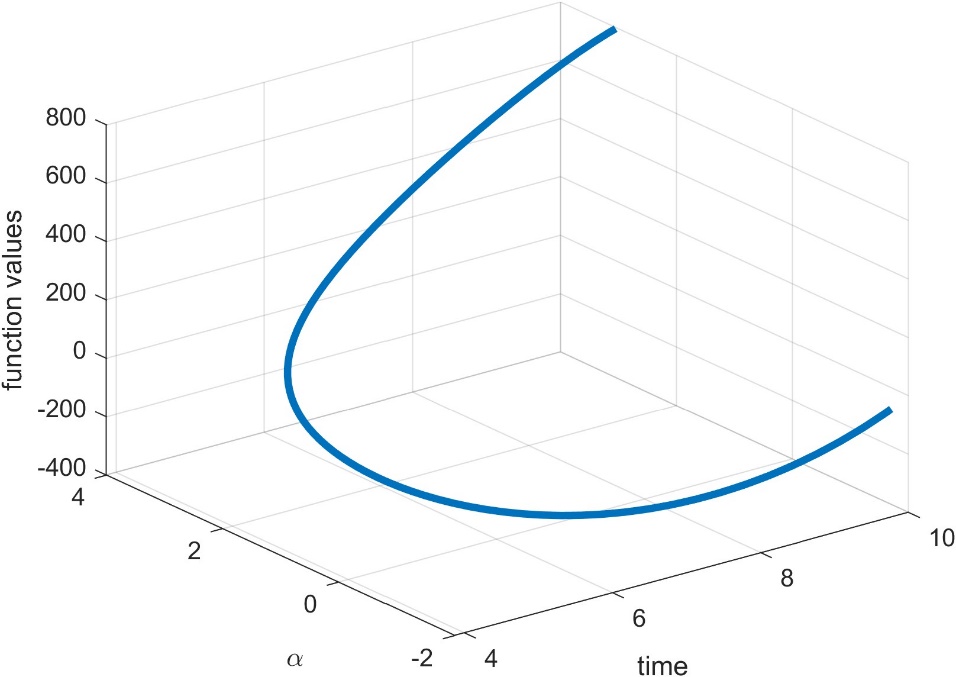


Figure : Line of intersection of surfaces

Since the intersection of these surfaces is not a single point it can be said that there are infinitely many solutions to the problem.

|  |  |
| --- | --- |
|  | (13) |
|  | (14) |
|  | (15) |
|  | (16) |

Solution of the system will give the new values for the next iteration.

|  |  |
| --- | --- |
|  | (17) |

For Newton Raphson method Jacobian matrix must be created as follows.

|  |  |
| --- | --- |
|  | (18) |
|  | (19) |

The result for the initial conditions

Code:

clc; clear;

f1 = @(t, a) 800 - 100 \* t \* (cos(a) + 1);

f2 = @(t, a) 15.2 \* t^2 + 80 \* t \* (sin(a) - 1);

f1t = @(t, a) - 100 \* (cos(a) + 1);

f1a = @(t, a) 100 \* t \* sin(a);

f2t = @(t, a) 30.4 \* t^2 + 80 \* (sin(a) - 1);

f2a = @(t, a) 80 \* t \* cos(a);

[t, a] = NR(f1, f2, f1t, f1a, f2t, f2a, 2.5, 0.5, 1e-6, 1000)

function [x, y] = NR(f1, f2, f1x, f1y, f2x, f2y, x0, y0, eps, max\_iter)

iter = 0; error = inf;

x = x0; y = y0;

while error > eps && iter < max\_iter

J = [f1x(x, y), f1y(x, y); f2x(x, y), f2y(x, y)]; % Jacobian matrix

F = [f1(x, y); f2(x, y)]; % function value vector

del = J \ F; % Newton-Raphson step

x = x - del(1); % update solutions

y = y - del(2);

error = norm(del); % compute error

iter = iter + 1;

end

end

clc; clear;

t = linspace(0, 10, 100);

a = linspace(-pi, pi, 100);

[t, a] = meshgrid(t, a);

func1 = 800 - 100 .\* t .\* (cos(a) + 1);

func2 = 15.2 .\* t.^2 + 80 .\* t .\* (sin(a) - 1);

h1 = surf(t, a, func2, "FaceColor",[0 0 1]);

hold on

h2 = surf(t, a, func1, "FaceColor",[1 0 0]);

ylabel("\alpha"); xlabel("time"); zlabel("function values");

fdiff = func1 - func2;

C = contours(t, a, fdiff, [0 0]);

tL = C(1, 2:end);

aL = C(2, 2:end);

fL = interp2(t, a, func1, tL, aL);

h3 = line(tL, aL, fL, 'Color', 'yellow', 'LineWidth', 3);

legend([h1,h2,h3], {'f\_{1}(t, a)', 'f\_{2}(t, a)', "Line of Intersection"});

%exportgraphics(gca, "Q9a1.jpeg", "Resolution", 300)

hold off

plot3(tL, aL, fL, LineWidth=3)

ylabel("\alpha"); xlabel("time"); zlabel("function values");

grid on

%exportgraphics(gca, "Q9a12.jpeg", "Resolution", 300)

# 4. Discussion

In this homework I grasp the concepts on how to use basic root finding numerical methods and how to construct hybrid numerical method algorithms. In question one, it is asked to write a hybrid root finding algorithm. In question seven it is asked to write false-position and secant methods in MATLAB. In this question we are also asked to compare the methods. This comparison has a meaning because the secant method gave a negative number in one of the iterations where the function is not defined for negative numbers. So, this demonstrated the weakness of the method. In question nine, we have been asked to plot some given function and solve the problem graphically. It came out that the system has infinitely many solutions. Also, in this question we are asked to write a Newton Raphson code for the system of nonlinear equations. In this problem, since the problem has infinitely many solutions different initial guesses resulted in different results.

# 5. Conclusion

In conclusion, in this homework, we developed our understanding on numerical root finding methods and how to construct simple numerical root finding algorithms. In this homework I have especially developed my MATLAB skills on plotting. I have learned have to create three- and two-dimensional plots. In my case this homework was very helpful in enhancing my MATLAB skills. I also developed my understanding of the various numerical methods but in this homework the emphasis was on the MATLAB side. To conclude, this homework was very helpful in learning MATLAB and numerical methods.