MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING

ME 310 – NUMERICAL METHODS

SPRING 2022

HOMEWORK 1

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
2. Body 3
    Question 1 3
    Question 9 4
    Question 10 6
4. Discussion 7
5. Conclusion 7

Table of Figures

Table 1: Numerical values of the calculation process 4

1. Introduction

In this homework assignment, students are expected to learn how to use MATLAB by constructing fundamental numerical methods such as incremental search in question one, various forms of errors and their uses in question nine, and also error propagation and how to approximate this error in question ten.

2. Body

Question 1

In this question students are expected to write an updated version of the incremental search algorithm. When a root is found the increment is reduced by a factor and started again in the interval. So that the root approximation is better than the classical algorithm and search can be done faster.

Code:

```
% METU ME310 Spring 2023
% Author: Muhammed Rustem Sevik 2446888

% This code finds the first n postive roots of a given function using
% nested incremental search

% Note that this code contains two functions one of them is modified
% version of the code on OdtuClass and other one is to make the original
% code perecise

f = @(x) sin(x.^2) + exp(-0.5*x);

IncrementalSearch(f, 0, 100, 0.01, 5, 3, 0.1)

function [roots] = Incremental(f, xMin, xMax, dx, nRoot)

foundRoots = 0;  % Counter for roots that are found
x = xMin;

while foundRoots < nRoot && x <= xMax
    f1 = f(x);
    f2 = f(x+dx);

    if f1*f2 < 0  % Sign change indicates that there is a root in [x,x+dx]
        foundRoots = foundRoots + 1;

        % To locate the root in [x,x+dx] we can perform linear
        % interpolation between points (x,f1) and (x+dx,f2).
        roots(foundRoots,1:2) = [x x+dx];
    end
    x = x + dx;
end

end  % of the function
```

```
function [final_roots] = IncrementalSearch(f, xMin, xMax, dx, nRoot, ...
    nSearch, fraction)
    roots = Incremental(f, xMin, xMax, dx, nRoot);
    a = size(roots);
    final_roots = zeros(a(1,1),1);

    for i = 1 : a(1,1)
        dx_local = dx;

        for j = 1 : (nSearch - 1)

            dx_local = dx_local * fraction;
            roots(i,1:2) = Incremental(f,roots(i,1),roots(i,2),dx_local,1);

        end % of j for loop

        final_roots(i,1) = (abs(f(roots(i,1)))*(roots(i,2))+ ...
            abs(f(roots(i,2)))*roots(i,1)) / ...
            (abs(f(roots(i,1)))+abs(f(roots(i,2))));

    end % of i for loop

end % of the function
```

Note that the code can work in the case where the function does not have more than one root in the interval smaller than the increment. Also when the function have double root, no sign change will occur hence the algorithm will fail to detect the root. These edge cases are not considered while writing the code. For such functions one needs to use a more suitable algorithm.

Question 9

$$f(x,y) = \sin(x) + 2\cos(y) \tag{1}$$

Second order Taylor Series expansion of $f(x,y)$ about points a and b yields to,

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b) + \frac{f_{xx}(a,b)}{2}(x-a)^2 + f_{xy}(a,b)(x-a)(y-b) + \frac{f_{yy}(a,b)}{2}(y-b)^2 \tag{2}$$

Substituting numerical values into the expression,

Table 1: Numerical values of the calculation process

Order	$f(0.51, 0.51)$	ϵ_t	E_t
0	2.234590662384949	4.138488485018952e-02	9.244002105384119e-04
1	2.233777977231768	5.001421172446531e-03	1.117150573581860e-04
2	2.233666247698649	6.480717947636987e-07	1.447576103430492e-08

Error definitions are,

$$E_t = |\text{True Value} - \text{Approximate Value}| \tag{3}$$

$$\epsilon_t = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \cdot 100\% \tag{4}$$

Code:

```
f = @(x,y) sin(x) + 2 * cos(y);
fx = @(x,y) cos(x);
fy = @(x,y) - 2 * sin(y);
fxx = @(x,y) -sin(x);
fyy = @(x,y) - 2 * cos(y);
fxy = @(x,y) 0;

fprintf(" order          value          relative_error      true_error
\n")
Taylor(f, fx, fy, fxx, fyy, fxy, 0.5, 0.51, 0.5, 0.51);

function error = Calculate_Relative_Error(xnew, xold)
    error = abs(xnew - xold) / xnew *100;
end

function error = Calculate_Error(xnew, xold)
    error = abs(xnew - xold);
end

function sum = Taylor(f, fx, fy, fxx, fyy, fxy, x_0, x, y_0, y)
    hx = x - x_0;
    hy = y - y_0;
    real_value = f(x, y);
    sum = 0;
    % 0th order calculation
    sum = sum + f(x_0, y_0);
    err_rel = Calculate_Relative_Error(real_value, sum);
    err_t = Calculate_Error(real_value, sum);
    fprintf("%4d  %25.15f  %20.15e  %20.15e \n", 0, sum, err_rel, err_t)
    % 1st order calculation
    sum = sum + fx(x_0, y_0) * hx + fy(x_0, y_0) * hy;
    err_rel = Calculate_Relative_Error(real_value, sum);
    err_t = Calculate_Error(real_value, sum);
    fprintf("%4d  %25.15f  %20.15e  %20.15e \n", 1, sum, err_rel, err_t)
    % 2nd order calculation
    sum = sum + fxx(x_0, y_0) * hx^2 / ...
        2 + fxy(x_0, y_0) * hx * hy + fyy(x_0, y_0) * hy^2 / 2;
    err_rel = Calculate_Relative_Error(real_value, sum);
    err_t = Calculate_Error(real_value, sum);
    fprintf("%4d  %25.15f  %20.15e  %20.15e \n", 2, sum, err_rel, err_t)
end
```

Question 10

In this question it is asked to variation on the function ϕ_0 while the input vary 2%.

$$\phi_0(\alpha, r) = \sin^{-1}\left((1+\alpha)\sqrt{1-\frac{\alpha r^2}{1+\alpha}}\right) \quad \text{where } \left(\frac{v_e}{v_0}\right) = r$$

$$\Delta\phi_0(\alpha, r) = \frac{\partial\phi_0}{\partial\alpha}\Delta\alpha + \frac{\partial\phi_0}{\partial r}\Delta r$$

Because $\Delta r = 0$ no need to calculate $\frac{\partial\phi_0}{\partial r}$ term.

$$\frac{\partial}{\partial\alpha}\left(\sin^{-1}\left((1+\alpha)\sqrt{1-\frac{\alpha r^2}{1+\alpha}}\right)\right) = \frac{-2\alpha(r^2-1)-r^2+2}{2(\alpha+1)\sqrt{\frac{-\alpha r^2+\alpha+1}{\alpha+1}}\sqrt{\alpha(\alpha(r^2-1)+r^2-2)}}$$

$$\Delta\phi_0(\alpha, r) = \frac{-2\alpha(r^2-1)-r^2+2}{2(\alpha+1)\sqrt{\frac{-\alpha r^2+\alpha+1}{\alpha+1}}\sqrt{\alpha(\alpha(r^2-1)+r^2-2)}}\Delta\alpha$$

Substituting numerical values into the expression,

$$\Delta\phi_0(\alpha, r) = -0.018081157905095$$

Calculating the exact value,

$$\phi_0(0.25, 2) = 0.593199776149629$$

The range of values for ϕ_0,

$$\phi_0 = 0.593199776149629 \mp 0.018081157905095$$

or it can be written as,

$$0.574322180000658 \leq \phi_0 \leq 0.612077372298599$$

The real values are,

$$\phi_0 = 0.574088900132879 \text{ and } \phi_0 = 0.611859961194538$$

As we can see the approximation and the real value of the function is fairly close. The approximations can be used considering the required precision.

Code:

```
val = Compute_Angle(0.25, 2, 0.02)
low = Compute_Angle(0.25 * 0.98, 2, 0.02)
up = Compute_Angle(0.25 * 1.02, 2, 0.02)

function data = Compute_Angle(a, r, percent)
    angle =  asin((1 + a) * sqrt(1 - a / (1 + a) * r^2));
    d_by_da = (-2 * a * (r^2 - 1) - r^2 + 2) / (2 * (a + 1) * ...
        sqrt((-a * r^2 + a + 1)/(a + 1))*sqrt(a*(a*(r^2 - 1) + r^2 - 2)));
```

```
    del_y = d_by_da * percent * a;
    lower_bound = angle - del_y;
    upper_bound = angle + del_y;
    data = [angle, d_by_da, del_y, lower_bound, upper_bound];

end
```

4. Discussion

In this homework I grasp the concepts on how to use basic numerical methods and how to construct simple numerical method algorithm. In question one, it is asked to write a nested incremental search algorithm. I have written a simple incremental search algorithm which has some weaknesses. The algorithm is not able to detect the roots of some functions where the roots are placed in a smaller interval than the increment. Also the code cannot detect the double root cases since the sign does not change. In the question nine, it is asked to write a multivariable taylor series expansion in the order two. And also asked to calculate the relative and true errors. In this question I grasped the concepts of the error definitions. In the question ten, we are asked to calculate variation of the output when the input is changing. It is important in the case of error propagation. I learned how the small changes effect the output.

5. Conclusion

In conclusion, in this homework, we developed our understanding on numerical methods, how to construct simple numerical methods. I have grasped the concept of error, what are the definitions of errors and importance of different error definitions. I have learned what the error propagation is. To conclude this homework was a very helpful in learning numerical methods.