03.11.2020

GEBZE
TEKNİK ÜNİVERSİTESİ

Numerical Analysis
Homework - II

Alican Bayındır
200102002087

Elektronik Mühendisliği

1. (P.84 Q.2c-4c) Find the solution of $\sin(3x)+3e^{-2x}\sin(x)-3e^{-x}\sin(2x)-e^{-3x}=0$ for $3 \le x \le 4$ accurate to within 10^{-5} using

$TOL = 0{,}00001$

   (a) Newton's Method,

   (b) Modified Newton's Method.

$$f(x) = \sin(3x) + 3e^{-2x}\sin(x) - 3e^{-x}\sin(2x) - e^{-3x}$$

a) Newton's method:

$$x_{n+1} = x_n - \frac{f(n)}{f'(n)}$$

$$f'(x) = 3\cos(3x) - 6e^{-2x}\sin(x) + \cos(x)\,3e^{-2x} + 3e^{-x}\sin(2x) + 2\cos(2x) - 3e^{x} + 3e^{-3x}$$

Init value $= 3$

I prepared a MATLAB code which calculates the root of the function automatically according to Newton's Method. Its output can be seen below:

```
COMMAND WINDOW
>> newton_methods
P(0) = 3
P(1) = 3.148086e+00 --> Error = 1.480861e-01
P(2) = 3.141564e+00 --> Error = 6.522472e-03
P(3) = 3.141568e+00 --> Error = 4.285324e-06
```

b) Modified Newton's method

$$x_{i+1} = x_i - \frac{f(x_i) \cdot f'(x_i)}{[f'(x_i)]^2 - f(x_i) \cdot f''(x_i)}$$

$$f(x) = \sin(3x) + 3e^{-2x}\sin(x) - 3e^{-x}\sin(2x) - e^{-3x}$$

$$f'(x) = 3\cos(3x) - 6e^{-2x}\sin(x) + \cos(x)\, 3e^{-2x} + 3e^{-x}\sin(2x) + 2\cos(2x) - 3e^{x} + 3e^{-3x}$$

$$f''(x) = 9e^{-2x}\sin(x) - 9\sin(3x) - 12e^{-2x}\cos(3x) - 9e^{-3x} + 12\cos(2x)e^{-x} + 9\sin(2x)e^{-x}$$

tolerance $= 10^{-5}$

The output of the MATLAB algorithm;

COMMAND WINDOW

```
>> modified_newton
P(0) = 3
P(1) = 3.128134e+00 --> Error = 1.281344e-01
P(2) = 3.141569e+00 --> Error = 1.343499e-02
P(3) = 3.141568e+00 --> Error = 1.434148e-06
```

2. (P.99 Q.2d-4d) Find approximations to within 10^{-5} to all the zeros of $f(x) = x^5 + 11x^4 - 21x^3 - 10x^2 - 21x - 5$ by

   (a) first finding the real zeros using Newton's Method and then reducing to polynomials of lower degree to determine any complex zeros,

   (b) Müller's Method.

$$f(x) = x^5 + 11x^4 - 21x^3 - 10x^2 - 21x - 5$$
$$f'(x) = 5x^4 + 44x^3 - 63x^2 - 20x - 21$$

a) Newton-Raphson formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

When initial value $P(0) = 1$;

COMMAND WINDOW

```
>> newton_question2
P(0) = 1
P(1) = 1.818182e-01 --> Error = 8.181818e-01
P(2) = -1.683929e-01 --> Error = 3.502111e-01
P(3) = -2.518760e-01 --> Error = 8.348307e-02
P(4) = -2.502382e-01 --> Error = 1.637752e-03
P(5) = -2.502369e-01 --> Error = 1.281120e-06
```

When initial value $P(0) = 2$;

COMMAND WINDOW

```
>> newton_question2
P(0) = 2
P(1) = 2.394958e+00 --> Error = 3.949580e-01
P(2) = 2.278151e+00 --> Error = 1.168065e-01
P(3) = 2.260466e+00 --> Error = 1.768521e-02
P(4) = 2.260086e+00 --> Error = 3.805255e-04
P(5) = 2.260086e+00 --> Error = 1.736398e-07
```

When initial value $P(0)=3$;

```
COMMAND WINDOW

>> newton_question2
P(0) = 3
P(1) = 2.567196e+00 --> Error = 4.328042e-01
P(2) = 2.336914e+00 --> Error = 2.302813e-01
P(3) = 2.266415e+00 --> Error = 7.049935e-02
P(4) = 2.260133e+00 --> Error = 6.282019e-03
P(5) = 2.260086e+00 --> Error = 4.756272e-05
P(6) = 2.260086e+00 --> Error = 2.712005e-09
```

So, we have found three initial value. Now, we can write the current version of the equation as;

$$f(x) = (x+0{,}250237)(x-2{,}260086)(x+12{,}612629)(x^2+0{,}3974x+0{,}7009)$$

$$g(x) = (x^2+0{,}3974x+0{,}7009)$$

to solve $g(x)$;

$$\Delta = b^2-4ac$$

$$\Delta = -2{,}6457$$

$$x_1 = \frac{-b+\sqrt{\Delta}}{2a}$$

$$x_1 = -0.1987+08133i$$

$$x_2 = -0.1987-08133i$$

Finally, we say that the equation has five roots and they are;

$$\begin{bmatrix} -0.1987+08133i & -0{,}2502 & \\ & & +12{,}6124 \\ -0.1987-08133i & +2{,}2600 & \end{bmatrix}$$

b) Muller's method;

To apply Muller's method we need three initial Points as x_0, x_1, x_2. We use them to find x_3 which we consider as intersection of x-axis of the parabola. $(x_n, f(x_n))$

$$P(x) = a(x-x_2)^2 + b(x-x_2) + c$$

$$f(x_0) = a(x_0-x_2)^2 + b(x_0-x_2) + c$$

$$f(x_1) = a(x_1-x_2)^2 + b(x_1-x_2) + c$$

$$f(x_2) = a\underbrace{(x_2-x_2)}_{0} + b\underbrace{(x_2-x_2)}_{0} + c = c$$

$$c = f(x_2)$$

so;

$$a = \frac{(x_0-x_2)^2[f(x_1)-f(x_2)] - (x_1-x_2)^2[f(x_0)-f(x_2)]}{(x_0-x_2)(x_1-x_2)(x_0-x_1)}$$

when $p0=1$, $p1=2$, $p2=0$ we find the root as;

COMMAND WINDOW

```
>> muller_method
P(0) = 1
P(1) = 2
P(2) = 0
P(3) = 4.473684e-01
P(4) = -1.532118e-01
P(5) = -2.699628e-01
P(6) = -2.500009e-01
P(7) = -2.502480e-01
P(8) = -2.502369e-01
P(9) = -2.502369e-01
```

$P(9) = -0,2502$

when initial values changed to $p0=1$, $p1=2$, $p2=4$ we find the root;

COMMAND WINDOW

```
>> muller_method
P(0) = 1
P(1) = 2
P(2) = 4
P(3) = 2.024691e+00
P(4) = 2.212326e+00
P(5) = 2.250167e+00
P(6) = 2.260141e+00
P(7) = 2.260086e+00
P(8) = 2.260086e+00
```

$p(7) = 2,2600$

when initial values changed to $p0=1$, $p1=2$, $p2=0,5 \pm 0,7i$

we find the root; $p(10) = -0,1987 \pm 0.8133i$

muller_method.m

COMMAND WINDOW

```
P(10) = -2.502382e-01
P(11) = -2.502369e-01

>> muller_method
P(0) = -1
P(1) = -2
P(2) = -5
P(3) = -1.905992e+00
P(4) = -1.458848e+00
P(5) = -1.172058e+00
P(6) = -7.334307e-01
P(7) = -4.602571e-01
P(8) = -2.874973e-01
P(9) = -2.107468e-01
P(10) = -1.989811e-01
P(11) = -1.987097e-01
P(12) = -1.987095e-01
```

WORKSPACE

Name	Value	Size
b	40.5768 +13.3252i	1×1
D	40.5768 +13.3252i	1×1
d	-34.3115 +77.4859i	1×1
delta1	41.2082 +12.4640i	1×1
delta2	40.6002 +13.3104i	1×1
E	81.1536 +26.6504i	1×1
f	@(x)(x^5)+(11*x^4)-(21*x^3)-(10*...	1×1
h	1.3072e-07 + 3.8887e-07i	1×1
h1	0.0118 + 0.0023i	1×1
h2	2.7141e-04 + 1.8186e-04i	1×1
i	12	1×1
N0	1000	1×1
p	-0.1987 - 0.8133i	1×1
p0	-0.2107 - 0.8158i	1×1
p1	-0.1990 - 0.8135i	1×1
p2	-0.1987 - 0.8133i	1×1
TOL	1.0000e-05	1×1

Well I changed values randomly but could not find the root 12.6124 but I am sure in some value of p0, p1, p2 we can find it as we did before in Newton's Method. So, roots are verified.

3. (P.64 Q.7) Use a fixed-point iteration method to determine a solution accurate to within 10^{-2} for $x^4 - 3x^2 - 3 = 0$ on $[1, 2]$. Use $p_0 = 1$.

$$g(x) = \sqrt[4]{3x^2+3}$$

While calculating fixed point iteration p_n becomes;

$$p_n = g(p_{n-1})$$

COMMAND WINDOW

```
>> fixed_point
0        1
1        1.565085e+00
2        1.793573e+00
3        1.885944e+00
4        1.922848e+00
5        1.937508e+00
6        1.943317e+00
```

APPENDIX

muller_method.m × | fixed_point.m × | newton_question2.m × | newton_question1.m × | +

```
syms f(x) x
f(x) = sin(3*x)+(3*exp(-2*x)) *sin(x) - (3*exp(-x)) *sin(2*x) - exp(-3*x);
g(x) = diff(f);
p0 = 3;
fprintf('P(0) = %d\n', p0);
for i=1:1000 %it should be stopped when tolerance is reached
        p = double(p0-(f(p0)/g(p0)));
        fprintf('P(%d) = %d --> Error = %d\n', i, double(p), abs(p-p0))
        if (abs(p-p0) <= 0.00001)
            fprintf('\n')
            return
        end
        p0=p;
end
```

The code that I use to calculate first question.

muller_method.m × | fixed_point.m × | newton_question2.m × | newton_question1.m × | +

```
syms f(x) x
f(x) = (x^5) + (11*x^4) - (21*x^3) - (10*x^2) - (21*x) - 5;
g(x) = diff(f);
p0 = -13;
fprintf('P(0) = %d\n', p0);
for i=1:1000 %it should be stopped when tolerance is reached
        p = double(p0-(f(p0)/g(p0)));
        fprintf('P(%d) = %d --> Error = %d\n', i, double(p), abs(p-p0))
        if (abs(p-p0) <= 0.00001)
            fprintf('\n')
            return
        end
        p0=p;
end
```

The code that I use to calculate second question muller_method.m × fixed_point.m × newton_question2.m × newton_question1.m ×

```
%Muller's Method
f =@(x) (x^5) + (11*x^4) - (21*x^3) - (10*x^2) - (21*x) - 5;
TOL = 10^-5;
N0 = 1000;
p0 = -1;
p1 = -2;
p2 = -5;
fprintf('P(0) = %d\n', p0)
fprintf('P(1) = %d\n', p1)
fprintf('P(2) = %d\n', p2)
h1 = p1 - p0;
h2 = p2 - p1;
delta1 = (f(p1) - f(p0))/h1;
delta2 = (f(p2) - f(p1))/h2;
d = (delta1 - delta2)/(h2 + h1);
i = 3;
while (i <= N0)
    b = delta2 + h2*d;
    D = (b.^2 - 4*f(p2)*d).^(1/2);
    if (abs(b-D) < abs(b+D))
        E = b + D;
    else
        E = b - D;
    end
    h = (-2)*f(p2)/E;
    p = p2+h;
    fprintf('P(%d) = %d\n', i, p)
    if (abs(h) < TOL)
        fprintf('\n')
        return
    end
    p0 = p1;
    p1 = p2;
    p2 = p;
    h1 = p1 - p0;
    h2 = p2 - p1;
    delta1 = (f(p1) - f(p0))/h1;
    delta2 = (f(p2) - f(p1))/h2;
    d = (delta2 - delta1)/(h2 + h1);
    i = i + 1;
end
fprintf('Method failed after %d iterations.', N0)
```

COMMAND WINDOW

The code that I use to calculate Muller's method muller_method.m × | fixed_point.m × | newton_question2.m × | newton_question1.m × | +

```
%Fixed Point Iteration
f =@(x) sqrt(sqrt((3*x^2)+3));
p0 = 1;
TOL = 10^-2;
N0 = 1000;
i = 1;
fprintf('0\t%d\n', p0)

while (i <= N0)
    p = double(f(p0));
    fprintf('%d\t%d\n', i, p)
    if (abs(p-p0) <= TOL)
        return
    end
    i = i + 1;
    p0 = p;
end
fprintf('Method failed after %d iterations.', N0)
```

The code that I use to calculate fixed point iteration.