Homework 1 – A Report on Solutions of Equations in One Variable

Xiaolong Liu, 201900170025

10th.Oct.2021

Shandong University, Taishan College

Abstract

In this report, we will introduce three method of finding solutions of equations in one variable. And then use these three methods to calculate roots of three functions.

Contents

1 Introductions

In this report, we will introduce three method of finding solutions of equations in one variable. And then use these three methods to calculate roots of three functions.

2 Algorithm Description & Mathematical Derivation

2.1 Newton's Method

Let $f \in C^2[a,b]$ and let $p_0 \in [a,b]$ be an approximation to p where p is a solution of $f(x)=0$ in (a,b) such that $f'(p_0) \neq 0$ and $|p-p_0|$ is sufficiently small. Consider the Taylor formula, we have

$$0 = f(p) = f(p_0) + (p-p_0)f'(p_0) + \frac{(p-p_0)^2}{2} f''(\xi(p)),$$

where $\xi(p)$ between p and p_0. Then we can write

$$0 \approx f(p_0) + (p-p_0)f'(p_0) \Rightarrow p \approx p_0 - \frac{f(p_0)}{f'(p_0)} := p_1.$$

Then consider the sequence $\{p_n\}_{n=0}^{\infty}$ such that

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, n \geq 1.$$

We only need the following steps:

```
INPUT initial approximation p_0; tolerance TOL, maximum number of
    iterations N_0.
OUTPUT approximate solution p or message of failure.
Step 1 Set i = 1.
Step 2 While i <= N_0 do Steps 3-6.
Step 3 Set p = p_0 - f(p_0)/f'(p_0). {Compute p_i.)
Step 4 If abs(p - p_0) < TOL then
        OUTPUT (p); (The procedure was successful.)
        STOP.
Step 5 Set i = i+1.
Step 6 Set p_0 = p. (Update p_0.)
Step 7 OUTPUT ('The method failed after N_0 iterations, N_0 =', N_0);
        (The procedure was unsuccessful.)
        STOP.
```

But now we should choose the initial value p_0 appropriately to make sure the iterative sequence convergent. Actually we have the following theorem.

Theorem 2.1.1. *Let $f \in C^2[a,b]$ and let $p \in (a,b)$ such that $f(p) = 0, f'(p) \neq 0$, then there exists a $\delta > 0$ such that the previous $\{p_n\}$ convergent to p for any $p_0 \in [p - \delta, p + \delta]$.*

But this theorem just tell us the existence of t_0 making $\{p_n\}$ converges. So in a practical application, an initial approximation is selected, and successive approximations are generated by Newton's method. Either these will generally converge quickly to the root or it will be clear that convergence is unlikely.

2.2 Steffensen's Method

Let $\Delta^k p_n$ means k-th forward difference, then we denote $\hat{p}_n = \{\Delta^2\}p_n := p_n - \dfrac{(\Delta p_n)^2}{\Delta^2 p_n}$. If we consider the solution of $g(x) = x$, then Steffensen's Method generate the following sequence:

$$p_0^{(0)}, p_1^{(0)} = g(p_0^{(0)}), p_2^{(0)} = g(p_0^{(0)}), p_0^{(1)} = \{\Delta^2\}(p_0^{(0)}), p_1^{(1)} = g(p_0^{(1)}), \cdots$$

The process is described in the following algorithm:

```
INPUT initial approximation p_0; tolerance TOL; maximum number of
    iterations N_0.
OUTPUT approximate solution p or message of failure.
Step 1 Set i = 1.
Step 2 While i <= N_0 do Steps 3-6.
Step 3 Set p_1 = g(p_0); (Compute P_1^{(i-1)}.)
            Pi = g(p_1); (Compute P_2^{(i-1)}.)
            P = P_0 - (p_1 - P_0)^2/(P_2 -2p_1 + p_0). (Compute p_0^{(
                i)}.)
Step 4 If abs(p - p_0) < TOL then
         OUTPUT (p); (Procedure completed successfully.)
```

```
         STOP.
Step 5 Set i = i + 1.
Step 6 Set p_0 = p. {Update p_0.)
Step 7 OUTPUT ('Method failed after N_0 iterations. N_0 =', N_0)',
        (Procedure completed unsuccessfully.)
        STOP.
```

As follows:

k	$p_0^{(k)}$	$p_1^{(k)}$	$p_2^{(k)}$
0	$p_0^{(0)}$	$p_1^{(0)} = g(p_0^{(0)})$	$p_2^{(0)} = g(p_1^{(0)})$
1	$p_0^{(1)} = p_0^{(0)} - \dfrac{(p_1^{(0)} - p_0^{(0)})^2}{p_2^{(0)} - 2p_1^{(0)} + p_0^{(0)}}$	$p_1^{(1)} = g(p_0^{(1)})$	$p_2^{(1)} = g(p_1^{(1)})$
2	$p_0^{(2)} = p_0^{(1)} - \dfrac{(p_1^{(1)} - p_0^{(1)})^2}{p_2^{(1)} - 2p_1^{(1)} + p_0^{(1)}}$	$p_1^{(2)} = g(p_0^{(2)})$	$p_2^{(2)} = g(p_1^{(2)})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Moreover, we have the following theorem.

Theorem 2.2.1. *Let $x = g(x)$ has solution p with $g'(p) \neq 1$. If there exists $\delta > 0$ such that $g \in C^3[p-\delta, p+\delta]$, then Steffensen's method gives quadratic convergence for any $p_0 \in [p-\delta, p+\delta]$.*

From the illustration, it appears that Steffensen's method gives quadratic convergence without evaluating a derivative.

2.3 Müller's Method

The fundamental theorem as follows

Theorem 2.3.1. *If $z = a+bi$ is a complex root of multiplicity m of the polynomial $P(x) \in \mathbb{R}[x]$, then $\bar{z} = a - bi$ is also a root of multiplicity m of $P(x)$ and $(x^2 - 2ax + a^2 + b^2)^m | P(x)$.*

Let's begin. Try to find solution of $f(x) = 0$. Consider the quadratic polynomial $P(x) = a(x-p_2)^2 + b(x-p_2) + c$ passing through $(p_i, f(p_i)), i = 0, 1, 2$, then we can calculate a, b, c. Then consider another zero of P as

$$p_3 = p_2 - \frac{2c}{b + \operatorname{sgn}(b)\sqrt{b^2 - 4ac}},$$

Then we replace p_0 by p_3 and fix p_1, p_2, then calculate p_4. Then we repeat this process.

The process is described in the following algorithm:

```
INPUT p_0, p_1, p_2, tolerance TOL, maximum number of iterations N_0.
OUTPUT approximate solution p or message of failure.
Step 1 Set h_1 = p_1 - p_0;
           h_2 - p_2 - p_1,;
           \delta_1=(f(p_1)-f(P_0))/h_1;
```

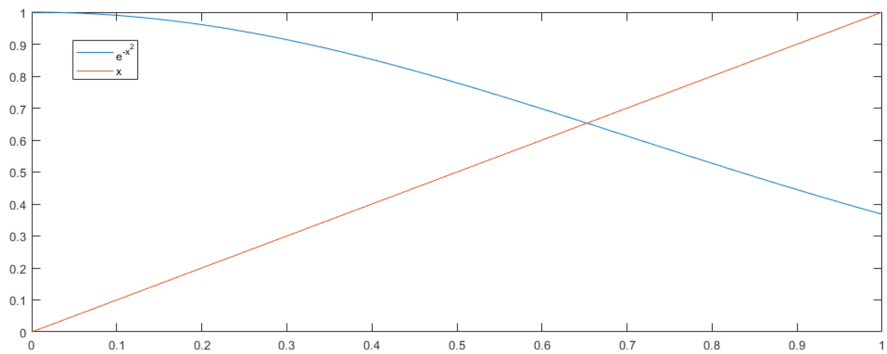
```
            \delta_2=(f(p_2)-f(P_1))/h_2;
            d = (\delta_2-\delta_1)/(h_2 + h_1);
            i = 3.
Step 2 While i < N_0 do Steps 3-7.
Step 3 b = \delta_2+h_2*d;
       D = (b^2 - 4f(p_2)d)^{1/2}. (Note: May require complex
           arithmetic.)
Step 4 If abs(b - D) < abs(b + D) then set E = b + D
                                   else set E = b - D.
Step 5 Set h = - 2f(p_2)/E;
            p = P_2 + h.
Step 6 If abs(h) < TOL then
           OUTPUT (p); (The procedure was successful.)
           STOP.
Step 7 Set p_0 = p_1; (Prepare for next iteration.)
            P_1 = P_2;
            P_2 = P;
            h_1=p_1-p_0;
            h_2=p_2-p_1;
            \delta_1=(f(p_1)-f(p_0))/h_1;
            \delta_2=(f(p_2)-f(p_1))/h_2;
            d = (\delta_2-\delta_1)/(h_2 + h_1);
            i=i+ 1.
Step 8 OUTPUT ('Method failed after N_0 iterations. N_0 =', M_0);
       (The procedure was unsuccessful.)
       STOP.
```

3 Codes and Numerical Experiments

3.1 Newton's Method

Problem 3.1.1. *Consider the function $f(x) = e^{-x^2} - x$ and approximate the root of f by using* Newton's Method.

Solution. First we consider the following diagram:



Then we may choose the initial point $p_0 = 0.60$. We have $f'(x) = -2xe^{-x^2} - 1$. We use Matlab to find the solution:

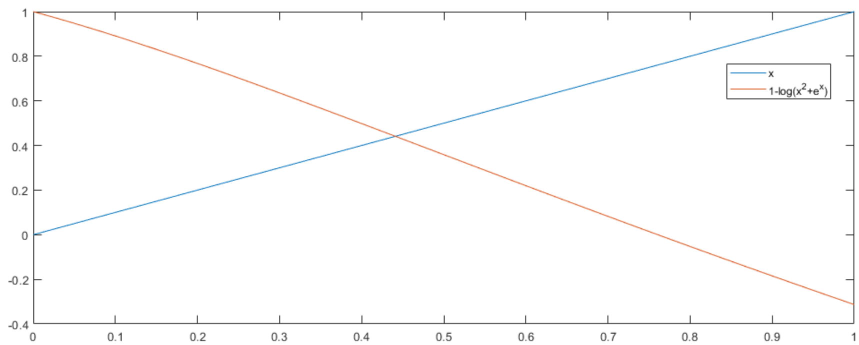
```
>> format long
>> p0=0.60;
p1 = p0+(exp(-p0*p0)-p0)/(2*p0*exp(-p0*p0)+1)
p1 =
    0.653165528964849
>>  p2 = p1+(exp(-p1*p1)-p1)/(2*p1*exp(-p1*p1)+1)
p2 =
    0.652918643575563
>>  p3 = p2+(exp(-p2*p2)-p2)/(2*p2*exp(-p2*p2)+1)
p3 =
    0.652918640419205
>> p4 = p3+(exp(-p3*p3)-p3)/(2*p3*exp(-p3*p3)+1)
p4 =
    0.652918640419205
```

Then we have the approximate root 0.652918640419205. ☐

3.2 Steffensen's Method

Problem 3.2.1. *Consider the function $f(x) = \log(x^2 + e^x) + x - 1$ and approximate the root of f by using* Steffensen's Method.

Solution. First we let $g(x) = 1 - \log(x^2 + e^x)$ and consider the following diagram:



Then we use the following code:

```
>> format long
p00=0.4;
p10=1-log(p00*p00+exp(p00)),p20=1-log(p10*p10+exp(p10));
p01=p00-((p10-p00)^2)/(p20-2*p10+p00);
p11=1-log(p01*p01+exp(p01)),p21=1-log(p11*p11+exp(p11));
p02=p01-((p11-p01)^2)/(p21-2*p11+p01);
p12=1-log(p02*p02+exp(p02)),p22=1-log(p12*p12+exp(p12));
p03=p02-((p12-p02)^2)/(p22-2*p12+p02);
p13=1-log(p03*p03+exp(p03)),p23=1-log(p13*p13+exp(p13));
p04=p03-((p13-p03)^2)/(p23-2*p13+p03)
```

Then we have the following results:

k	$p_0^{(k)}$	$p_1^{(k)}$	$p_2^{(k)}$
0	0.40	0.498119445761088	0.361442775174023
1	0.441003342756265	0.441098840957910	0.440965797126198
2	0.441043247479188	0.441043247580673	0.441043247439289
3	0.441043247521595	0.441043247521595	0.441043247521595
4	0.441043247521595		
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Then the root of $f(x) = \log(x^2 + e^x) + x - 1$ about 0.441043247521595. □

3.3 Müller's Method

Problem 3.3.1. *Consider the function* $f(x) = x^5 - x^4 + 2x^3 - 3x^2 + x - 4$ *and approximate the root of* f *by using* Müller's Method.

Solution. Consider the following codes:

```
>> format long;
p0=(-);p1=(-);p2=(-);
d1=p0-p2;d2=p1-p2;d3=p0-p1;
fp0=p0^5-p0^4+2*p0^3-3*p0^2+p0-4;fp1=p1^5-p1^4+2*p1^3-3*p1^2+p1-4;fp2
    =p2^5-p2^4+2*p2^3-3*p2^2+p2-4;
a=(d2*(fp0-fp2)-d1*(fp1-fp2))/(d1*d2*d3),
b=((d1^2)*(fp1-fp2)-(d2^2)*(fp0-fp2))/(d1*d2*d3),
c=fp2,
p3=p2-(2*c)/(b+(abs(b)/b)*(b^2-4*a*c)^(1/2)),
f=p3^5-p3^4+2*p3^3-3*p3^2+p3-4,
p0=p1;p1=p2;p2=p3;
```

► First we have the following results about the real root with $p_0 = 1.2$,$p_1 = 1.5$ and $p_2 = 1.7$:

i	p_i	$f(p_i)$
3	1.498296574175876	0.001835517596346
4	1.498189939893628	-7.718962482528013e-07
5	1.498189984726409	1.910116509407089e-11

So the first root is about 1.49818998.

► Then we have the following results about first two complex roots:

$p_0 = -0.7$	$p_1 = -0.3$	$p_2 = 0$
i	p_i	$f(p_i)$
3	0.016020408163265 - 0.792266894176845i	-2.523506209578482 - 0.065465541675960i
4	-0.470413316909407 - 1.057976557811231i	-0.445654226884208 - 0.256667459692234i
5	-0.538941161906186 - 1.128603805703716i	0.306190136268743 + 0.390807671993063i
6	-0.512010333510205 - 1.090420779711699i	-0.017911509025860 - 0.009840947774387i
7	-0.513693201430710 - 1.091466437854219i	0.000496378495271 - 0.001049828059143i
8	-0.513639604731048 - 1.091565404686881i	5.778364521535906e-05 + 2.717918063055436e-05i
9	-0.513634156123568 - 1.091562467843629i	-1.488296068608008e-06 + 3.164174593006663e-06i
10	-0.513634316942909 - 1.091562169479793i	-1.732752541450111e-07 - 8.150121066918814e-08i

So the first conjugate roots are about $-0.513634 \pm 1.091562i$.

▶ Finally we have the following results about the second two complex roots:

$p_0 = 0.1$	$p_1 = 0.2$	$p_2 = 0.3$
i	p_i	$f(p_i)$
3	0.208188775510204 + 1.411532054143292i	0.084080577956281 + 1.079480207905581i
4	0.344144312633665 + 1.233650211820825i	-0.385195493760319 - 1.088373138986842i
5	0.335421919618369 + 1.374570738136923i	0.910443736643821 - 0.308951200987823i
6	0.237657273752753 + 1.359629319679118i	0.024168980972318 + 0.419487647096113i
7	0.252742116488426 + 1.320135727833907i	-0.136660632701914 + 0.030305488089555i
8	0.268137779920826 + 1.323129590199704i	-0.015419230757869 - 0.061141085363400i
9	0.266952059567213 + 1.330052625293138i	0.028613082478504 - 0.006777067470872i
10	0.263785811882965 + 1.329448177736219i	0.002901134182951 + 0.012745532671000i
11	0.264064332912165 + 1.328042674407968i	-0.005618897592586 + 0.001299632959213i
12	0.264687098919871 + 1.328163294425883i	-0.000583235719683 - 0.002502776441912i
13	0.264633717800785 + 1.328440858412159i	0.001117023917614 - 0.000259516747265i
14	0.264509959327432 + 1.328416949861474i	1.154180412523687e-04 + 4.975345356668104e-04i
15	0.264520622214368 + 1.328361833393245i	-2.215187903211735e-04 + 5.141918052364858e-05i
16	0.264545166707100 + 1.328366577551762i	-2.290954742845130e-05 - 9.866719215567699e-05i
17	0.264543054140941 + 1.328377510240549i	4.395105825683032e-05 - 1.020377266569561e-05i
18	0.264538603619907 + 1.328374400038764i	-8.719393883804827e-06 + 2.024242219844652e-06i

So the second conjugate roots are about $0.26454 \pm 1.32837i$. □

4 Discussions and Conclusions

In our experiments we find that the number of times Newton's method and Steffensen's method consumed less than 5 times os they are really fast. But in Müller's method, the real roots are rapid convergence, but the conjugate roots are not so fast. Besides, we should take initial value very carefully unless it may not be able to converge, especially in the case of Müller's method.

References

[1] Richard L. Burden, J. Douglas Faires, Annette M. Burden, *Numerical Analysis (Tenth Edition)*, Cengage Learning, 2014.