MATH 352 - Homework 2

Twymun Safford

February 26th, 2024

Contents

1 Section 2.2 - Question 3 (a,b,c,d)

The following four methods are proposed to compute $21^{\frac{1}{3}}$. Rank them in order, based on their apparent speed of convergence, assuming $p_0 = 1$.

a)

$$p_n = \frac{20p_{n-1} + \frac{21}{p_{n-1}^2}}{21}$$

b)

$$p_n = p_{n-1} - \frac{p_{n-1}^3 - 21}{3p_{n-1}^2}$$

c)

$$p_n = p_{n-1} - \frac{p_{n-1}^4 - 21p_{n-1}}{p_{n-1}^2 - 21}$$

d)

$$p_n = (\frac{21}{p_{n-1}})^{\frac{1}{2}}$$

1.1 Solution

We first determine what $21^{\frac{1}{3}}$. That value is approximately 2.75892417638 and we are given that $p_0 = 1$. We can calculate the terms for each of the above methods using a program such as Excel. We can see this in the screenshot below (which comes from the Excel file included in the deliverable):

	a	b	c	d
0	1	1	1	1
1	1.952380952	7.666666667	0	4.582575695
2	2.121754274	5.230203739	0	2.140695143
3	2.242849692	3.742696919	0	3.132075595
4	2.334839673	2.994853568	0	2.589366527
5	2.40709338	2.777022226	0	2.847822274
6	2.465059288	2.759041866	0	2.715521253
7	2.512243463	2.758924181	0	2.780885095
8	2.551057096	2.758924176	0	2.748008838
9	2.583237767	2.758924176	0	2.764398093
P9-P8	0.047184175	-0.000117685	0	0.065363842

Figure 1: Excel worksheet showing convergence of methods

Sequences a, b, and d coverge but sequence b is converging fastest. Also sequence c is not converging at all. Therefore, based on their apparent speed of convergence the order of convergence is $b > a > d > c$.

2 Section 2.3 - Question 1

Let $f(x) = x^2 - 6$ and $p_0 = 1$. Use Newton's method to find p_2.

2.1 Solution

The derivative of the function is $f'(x)$ which is:

$$f'(x) = \frac{d}{dx}(x^2 - 6) = 2x$$

Therefore, from iterative formula 2.7 from this chapter we have:

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} = p_{n-1} - \frac{p_{n-1}^2 - 6}{2p_{n-1}}$$

Now, for n=1 and taking into account the initial value $p_0 = 1$ we have that:

$$p_1 = p_0 - \frac{p_0^2 - 6}{2p_0} = 1 - \frac{1^2 - 6}{2(1)} = \frac{7}{2}$$

Now, for n=2 and taking into account that $p_1 = \frac{7}{2}$ we have that:

$$p_2 = p_1 - \frac{p_1^2 - 6}{2p_1} = 1 - \frac{\frac{7}{2}^2 - 6}{2(\frac{7}{2})} = \frac{73}{28}$$

Therefore, $p_2 = \frac{73}{28}$

3 Section 2.3 - Question 4a

Let $f(x) = x_3 - cos(x)$. With $p_0 = 1$ and $p_1 = 0$, find p_3 using the Secant Method.

3.1 Solution

To apply the secant method we can use the formula 2.12. We can notice that for n=2 and taking into account the initial values p_0 and p_1:

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} = 0 - \frac{f(0)(0 - (-1))}{f(0) - f(-1)}$$

$$p_2 = \frac{[-(0)^3 - cos(0)](1)}{[-(0)^3 - cos(0)] - [-(-1)^3 - cos(-1)]}$$

$$p_2 = \frac{1}{-2 + cos(1)} = -0.685073$$

So that, p_2=0.685073 Now, for n=3 and using the result in the previous iteration:

$$p_3 = p_2 - \frac{f(p_2)(p_2 - p_1)}{f(p_2) - f(p_1)} = -0.685073 - \frac{f(-0.685073)(-0.685073 - 0)}{f(-0.685073) - f(0)} = -1.25208$$

4 Section 2.3 - Question 4b

Let $f(x) = x_3 - cos(x)$. With $p_0 = 1$ and $p_1 = 0$, find p_3 using the method of False Position.

4.1 Solution

We can notice that:

$$f(p_0)f(p_1) = f(1)f(0) = (1 - cos(1))(-1) < 0$$

Therefore, we can execute the first iteration of the method of False Position. This result has already been determined in part a) because the same iterative formula is used. Thus, as in part a) p_2=0.685073. Now, to carry the second iteration we have taken into account that:

$$f(p_2)f(p_0) = (-0.45285)(1 - cos(1)) < 0$$

So, we re-index the initial value p_1, $p2 = 0.45285$ and $p1 = 1$.
For these p-values we have that:

$$p_3 = p_2 - \frac{f(p_2)(p_2 - p_1)}{f(p_2) - f(p_1)} = -0.685073 - \frac{f(-0.685073)(-0.685073 - (-1)}{f(-0.685073) - f(-1)} = -0.841355$$

Thus, p_3 = -0.841355.

5 Section 2.4 - Question 13

The iterative method to solve $f(x) = 0$, given by the fixed-point method $g(x) = x$, where

$$p_n = g(p_{n-1}) = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} - \frac{f''(p_{n-1})}{2f'(p_{n-1})}\left[\frac{f(p_{n-1})}{f'(p_{n-1})}\right]^2, \quad \text{for } n = 1, 2, 3, \ldots,$$

has $g'(p) = g''(p) = 0$. This will generally yield cubic ($\alpha = 3$) convergence. Expand the analysis of Example 1 to compare quadratic and cubic convergence.

Example 1 Let $f(x) = e^x - x - 1$. **(a)** Show that f has a zero of multiplicity 2 at $x = 0$. **(b)** Show that Newton's method with $p_0 = 1$ converges to this zero but not quadratically.

Table 2.8

n	p_n
0	1.0
1	0.58198
2	0.31906
3	0.16800
4	0.08635
5	0.04380
6	0.02206
7	0.01107
8	0.005545
9	2.7750×10^{-3}
10	1.3881×10^{-3}
11	6.9411×10^{-4}
12	3.4703×10^{-4}
13	1.7416×10^{-4}
14	8.8041×10^{-5}
15	4.2610×10^{-5}
16	1.9142×10^{-6}

Solution **(a)** We have

$$f(x) = e^x - x - 1, \qquad f'(x) = e^x - 1 \qquad \text{and} \qquad f''(x) = e^x,$$

so

$$f(0) = e^0 - 0 - 1 = 0, \qquad f'(0) = e^0 - 1 = 0 \qquad \text{and} \qquad f''(0) = e^0 = 1.$$

Theorem 2.12 implies that f has a zero of multiplicity 2 at $x = 0$.

(b) The first two terms generated by Newton's method applied to f with $p_0 = 1$ are

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)} = 1 - \frac{e-2}{e-1} \approx 0.58198,$$

and

$$p_2 = p_1 - \frac{f(p_1)}{f'(p_1)} \approx 0.58198 - \frac{0.20760}{0.78957} \approx 0.31906.$$

The first sixteen terms of the sequence generated by Newton's method are shown in Table 2.8. The sequence is clearly converging to 0, but not quadratically. The graph of f is shown in Figure 2.12. ■

5.1 Solution

We observe for $f(x) = e^x - x - 1$ that f(0) = 0 and f'(0) = 0. The condition for the given method to have cubic convergence is that the zero must be simple such that $f'(p) \neq 0$. Creating a table of these values shows:

n	$f(x_n)$	$f'(x_n)$	$f''(x_n)$	x_{n+1}	ε_a
1	0.718282	1.71828	2.71828	0.443757	-1.25349
2	0.114795	0.558551	1.55855	0.179304	-1.47489
3	0.0170803	0.196384	1.19638	0.0692878	-1.58781
4	0.00245682	0.0717446	1.07174	0.0262853	-1.63599
5	0.000348505	0.0266338	1.02663	0.00990029	-1.655
6	$4.917 \cdot 10^{-5}$	0.00994946	1.00995	0.00371874	-1.66227
7	$6.9231 \cdot 10^{-6}$	0.00372567	1.00373	0.00139539	-1.66501
8	$9.7401 \cdot 10^{-7}$	0.00139637	1.0014	0.000523394	-1.66605
9	$1.3699 \cdot 10^{-7}$	0.000523531	1.00052	0.00019629	-1.66643
10	$1.9266 \cdot 10^{-8}$	0.000196309	1.0002	$7.3611 \cdot 10^{-5}$	-1.66658
11	$2.7094 \cdot 10^{-9}$	$7.3614 \cdot 10^{-5}$	1.00007	$2.7605 \cdot 10^{-5}$	-1.66663
12	$3.8101 \cdot 10^{-10}$	$2.7605 \cdot 10^{-5}$	1.00003	$1.0352 \cdot 10^{-5}$	-1.66665
13	$5.3579 \cdot 10^{-11}$	$1.0352 \cdot 10^{-5}$	1.00001	$3.8819 \cdot 10^{-6}$	-1.66666
14	$7.5346 \cdot 10^{-12}$	$3.8819 \cdot 10^{-6}$	1	$1.4557 \cdot 10^{-6}$	-1.66666
15	$1.0596 \cdot 10^{-12}$	$1.4557 \cdot 10^{-6}$	1	$5.4586 \cdot 10^{-7}$	-1.66683
16	$1.4899 \cdot 10^{-13}$	$5.4586 \cdot 10^{-7}$	1	$2.0468 \cdot 10^{-7}$	-1.66696

Based on this table, this is why the method fails.

6 Section 2.5 - Question 4

Let $g(x) = 1 + (sin(x))^2$ and $p_0^{(0)} = 1$ Use Steffensen's method to find $p_0^{(1)}$ and $p_0^{(2)}$.

6.1 Solution

Steffensen's method in algorithm form is:

ALGORITHM **2.6**

Steffensen's

To find a solution to $p = g(p)$ given an initial approximation p_0:

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 $i \leq N_0$ do Steps 3–6.

Step 3 Set $p_1 = g(p_0)$; (*Compute* $p_1^{(i-1)}$.)
$p_2 = g(p_1)$; (*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 $|p - p_0| < TOL$ 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.

Note that $\Delta^2 p_n$ might be 0, which would introduce a 0 in the denominator of the next iterate. If this occurs, we terminate the sequence and select $p_2^{(n-1)}$ as the best approximation.

Illustration To solve $x^3 + 4x^2 - 10 = 0$ using Steffensen's method, let $x^3 + 4x^2 = 10$, divide by $x + 4$, and solve for x. This procedure produces the fixed-point method

$$g(x) = \left(\frac{10}{x+4}\right)^{1/2}.$$

Following the depicted procedure we determine p_1 and p_2 as:

$$p_1^{(0)} = g(p_0) = g(1) = 1 + (sin(1))^2 = 1.708073$$

$$p_2^{(0)} = g(p_1) = g(1.708073) = 1 + (sin(1.708073))^2 = 1.981273$$

Now we find $p_0^{(1)}$ as the first term in the sequence $\hat{p_n}$ given by the Aitken's Δ^2 method:

$$p_0^{(1)} = p_0^{(0)} - \frac{(p_1^{(0)} - p_0^{(0)})^2}{p_2^{(0)} - 2p_1^{(0)} + p_0^{(0)}}$$

$$= 1 - \frac{(1.708703 - 1)^2}{1.981273 - 2 \cdot 1.708703 + 1}$$

$$= 2.152905$$

Perform one more iteration in order to determine $p_0^{(2)}$:

$$p_1^{(1)} = g(p_0^{(1)}) = g(2.152905) = 1 + (sin(2.152905))^2 = 1.697734$$

$$p_2^{(1)} = g(p_1^{(1)}) = g(1.697734) = 1 + (sin(1.697734))^2 = 1.983973$$

$$p_0^{(2)} = p_0^{(1)} - \frac{(p_1^{(1)} - p_0^{(1)})^2}{p_2^{(1)} - 2p_1^{(1)} + p_0^{(1)}}$$

$$= 2.152905 - \frac{(1.697734 - 2.152905)^2}{1.983973 - 2 \cdot 1.697734 + 2.152905}$$

$$= 1.873464$$

7 Section 2.6 - Question 3

Find the approximations to within 10^4 to all the real zeros of the following polynomial using Müller's method:

$$f(x) = x^3 - 2x^2 - 5$$

7.1 Solution

Müller's method is depicted like so:

ALGORITHM
2.8

Müller's

To find a solution to $f(x) = 0$ given three approximations, p_0, p_1, and p_2:

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 $i \le N_0$ 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 $|b - D| < |b + D|$ then set $E = b + D$
else set $E = b - D$.

Step 5 Set $h = -2f(p_2)/E$;
$p = p_2 + h$.

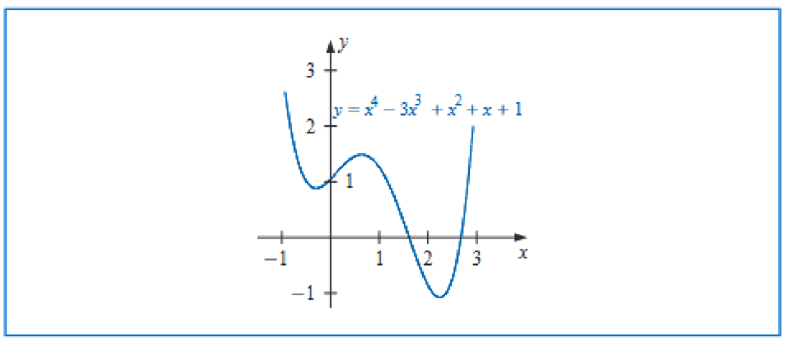
Step 6 If $|h| < TOL$ then
OUTPUT (p); (*The procedure was successful.*)
STOP.

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 =', N_0);
(*The procedure was unsuccessful.*)
STOP.

Illustration Consider the polynomial $f(x) = x^4 - 3x^3 + x^2 + x + 1$, part of whose graph is shown in Figure 2.14.

Figure 2.14



Three sets of three initial points will be used with Algorithm 2.8 and $TOL = 10^{-5}$ to approximate the zeros of f. The first set will use $p_0 = 0.5$, $p_1 = -0.5$, and $p_2 = 0$. The parabola passing through these points has complex roots because it does not intersect the x-axis. Table 2.12 gives approximations to the corresponding complex zeros of f.

We can implement Müller's method using Matlab, for instance. This is a nice implementation:

```
% MATLAB code goes here
% Name: Twymun K. Safford
% Class: MATH 352
% Assignment: Homework 2, Section 2.6, Problem 3a

%Implementation of Muller Method to aproximate roots
%within range of 10^-4 for f(x) = x^3 - 2x^2 - 5

```

```
function y=Muller()
  f=@(x) x^3-2*x^2-5;
  tol = 10^(-5);
  N0 = 10;
  p0=-1.5;
  p1=-1;
  p2=-0.5;
  h1=p1-p0;
  h2=p2-p1;
  delta1=(f(p1)-f(p0))/h1;
  delta2=(f(p2)-f(p1))/h2;
  d=(delta2-delta1)/(h2+h1);

  for i=3:N0
    b=delta2+h2*d;
    D=sqrt(b^2-4*f(p2)*d);

    if abs(b-D)<abs(b+D)
      E=b+D;
    else
      E=b-D;
    end

    h=-2*f(p2)/E;
    p=p2+h;
    fprintf("%d %10g%+g i %20g%+g i\n", i,real(p),imag(p),real(f(p)
        ),imag(f(p)));

    if abs(h)<tol
      y=p;
      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);
  end

end
```

From console, we can see that we have a complex root (which automatically yields another one, since complex roots come in conjugated pairs, according to Theorem 2.20), $x_{2,3}$=0.34532±1.31873i problem_2a.m | Muller.mlx

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Muller
3     -0.275+1.03652 i              -2.13694+0.261721 i
4  -0.283996+1.30664 i              -0.314983-0.430358 i
5  -0.346182+1.32145 i              0.0248391-0.00261464 i
6   -0.34532+1.31872 i           -4.87609e-05-1.81222e-05 i
7  -0.345324+1.31873 i           -6.80926e-10-8.17432e-10 i

ans =

  -0.3453 + 1.3187i

>> Muller
3     -0.275+1.03652 i              -2.13694+0.261721 i
4  -0.283996+1.30664 i              -0.314983-0.430358 i
5  -0.346182+1.32145 i              0.0248391-0.00261464 i
6   -0.34532+1.31872 i           -4.87609e-05-1.81222e-05 i
7  -0.345324+1.31873 i           -6.80926e-10-8.17432e-10 i

ans =

  -0.3453 + 1.3187i
```

And that there's a real root if we change p_0, p_1, and p_2 to be real and positive:

```
% Name: Twymun K. Safford
% Class: MATH 352
% Assignment: Homework 2, Section 2.6, Problem 3a

%Implementation of Muller Method to aproximate roots
%within range of 10^-4 for f(x) = x^3 - 2x^2 - 5

function y=Muller()
  f=@(x) x^3-2*x^2-5;
  tol = 10^(-5);
  NO = 10;
  p0=1.5;
  p1=2;
  p2=2.5;
  h1=p1-p0;
  h2=p2-p1;
```

```
  delta1=(f(p1)-f(p0))/h1;
  delta2=(f(p2)-f(p1))/h2;
  d=(delta2-delta1)/(h2+h1);

  for i=3:N0
    b=delta2+h2*d;
    D=sqrt(b^2-4*f(p2)*d);

    if abs(b-D)<abs(b+D)
      E=b+D;
    else
      E=b-D;
    end

    h=-2*f(p2)/E;
    p=p2+h;
    fprintf("%d %10g%+g i %20g%+g i\n", i,real(p),imag(p),real(f(p)),imag(f(p)));

    if abs(h)<tol
      y=p;
      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);
  end

end
```

```
>> Muller
3    2.70658+0 i            0.176121+0 i
4    2.69045+0 i         -0.00212076+0 i
5    2.69065+0 i        -5.87917e-07+0 i
6    2.69065+0 i        -1.65201e-13+0 i

ans =

    2.6906

>> Muller
3    2.70658+0 i            0.176121+0 i
4    2.69045+0 i         -0.00212076+0 i
5    2.69065+0 i        -5.87917e-07+0 i
6    2.69065+0 i        -1.65201e-13+0 i

ans =

    2.6906

>> 
```

With this real root being 2.6906.