UNIVERSITI TUNKU ABDUL RAHMAN

Assignment

Faculty : Faculty of Science

Unit Code : UDPS2013

Unit Title : Numerical Methods

Lecturer : Yeoh Hong Beng

Semester : 2020/01

Student Name	Student ID	Course
NGU YI HUI	18ADB01438	SC
LIM CHIEN AI	17ADB04072	SC

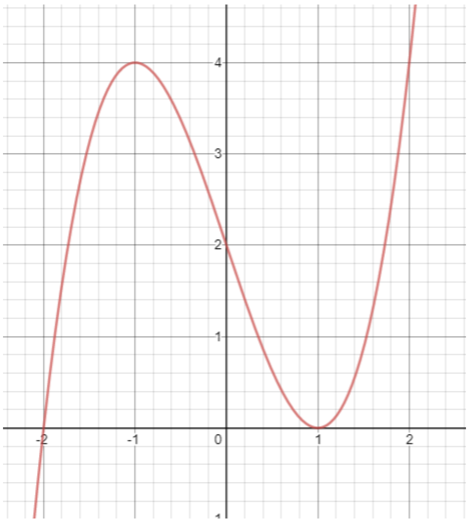
Q1. (a)

Method	Condition
(i) Bisection Method	1. f(x) is continuous at [a,b]. 2. f(a) and f(b) have opposite sign. 3. The rate of convergence is slow. 4. Bisection Method is not efficient because it might take many iterations to obtain a more accurate solution to f(x). 5. A good intermediate approximation can be unintentionally discarded or by-passed.
(ii) False-Position Method	1. f(x) is continuous at [a,b]. 2. Two initial estimations p_0 and p_1 are close to the root p. 3. $f(p_0)*f(p_1) < 0$, $f(p_0) \neq f(p_1)$. 4. The rate of convergence is usually faster than Bisection Method. 5. False- Position Method will converge faster if f(x) is close to a straight line.
(iii)Newton's Method	1. $f \in C^2[a,b]$, both f(x) and f'(x) are continuous at [a,b]. 2. The initial approximation, p0 is closed to the root p. 3. $f'(p_0) \neq 0$, $\|p-p_0\|$ is small. 4. $f'(n) \neq 0$, n=1,2,3, … 5. f(x) is differentiable. 6. $p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$ shows that when the magnitude of $f'(p_{n-1})$ is large, it will converge faster.
(iv)Steffensen's Method with Fixed Point Iteration method	1. For x=g(x), f(x) is continuous at [a,b]. 2. $g(x) \in [a,b]$. 3. $\|g'(x)\| \leq k < 1$. 4. Three initial estimations p_0, p_1, p_2 are close to the root p. 5. If the fixed point iteration converges linearly, Steffensen's Method will help increase the rate of convergence. 6. If g(x) is obtained by Newton Method, Steffensen's Method will not help much on increasing efficiency. 7. This method applies a modification of Aitken's Method to a linearly convergent sequence obtained from a fixed-point iteration.
(v) Secant Method	1. f(x) is continuous at [a,b]. 2. Two initial estimations p_0 and p_1 are close to the root p.

	3. $f(p_0)$ and $f(p_1)$ cannot be too close to each other, because it makes the secant line become flat. 4. $f(x)$ is well-fitted by the straight line.
(vi)Muller's Method	1. $f(x)$ is continuous at [a,b]. 2. Three initial estimations p_0, p_1, p_2 are close to the root p. 3. $p_1 - p_0 \neq 0$. 4. $p_1 - p_2 \neq 0$. 5. $p_0 - p_2 \neq 0$. 6. $f(x)$ is well-fitted by a parabola or curve. 7. Muller's Method will reduce estimation error for its next iteration.

Q1. (b)

f(x) = x^3 - 3x + 2



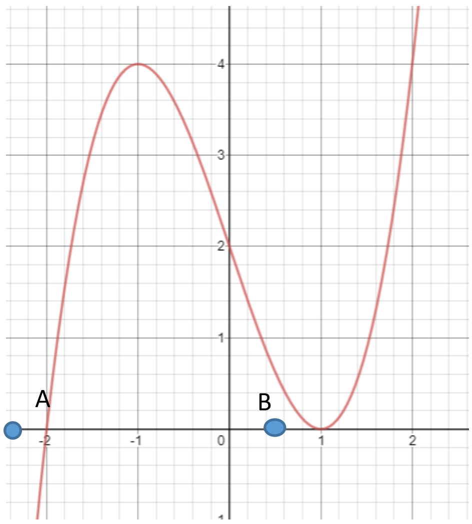
I. Newton Method

f(x) = x^3 - 3x + 2

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

$$P_n = P_{n-1} - \frac{f(P_{n-1})}{f'(P_{n-1})}, n = 1,2,3\ldots$$

$$g(P_n) = P_n - \frac{P_n^{\ 3} - 3P_n + 2}{3P_n^{\ 2} - 3}, n = 0,1,2\ldots.$$



For the simple root p = -2, we use point A= p_0 = -2.4,

n	p_n	p_{n+1}	Stop? 1 for 'Yes' and 0 for 'No'
0	-2.400000000	-2.076190476	-
1	-2.076190476	-2.003596011	0
2	-2.003596011	-2.000008590	0
3	-2.000008590	-2.000000000	0
4	-2.000000000	-2.000000000	1

p* = -2.00000000 (up to 8 decimal places)

number of iterations = 5

For the double root p = 1, we use point B= p_0 = 0.5,

n	p_n	p_{n+1}	Stop? 1 for 'Yes' and 0 for 'No'
0	0.500000000	0.777777778	-
1	0.777777778	0.893518519	0
2	0.893518519	0.947757252	0
3	0.947757252	0.974112168	0
4	0.974112168	0.987112665	0
5	0.987112665	0.993570263	0
6	0.993570263	0.996788587	0
7	0.996788587	0.998395155	0
8	0.998395155	0.999197792	0
9	0.999197792	0.999598950	0
10	0.999598950	0.999799488	0
11	0.999799488	0.999899747	0
12	0.999899747	0.999949875	0
13	0.999949875	0.999974937	0
14	0.999974937	0.999987469	0
15	0.999987469	0.999993734	0
16	0.999993734	0.999996867	0
17	0.999996867	0.999998434	0
18	0.999998434	0.999999217	0
19	0.999999217	0.999999608	0
20	0.999999608	0.999999804	0
21	0.999999804	0.999999902	0
22	0.999999902	0.999999951	0
23	0.999999951	0.999999976	0
24	0.999999976	0.999999987	0
25	0.999999987	0.999999998	0
26	0.999999998	0.999999998	1

p* =1.00000000   (up to 8 decimal places)

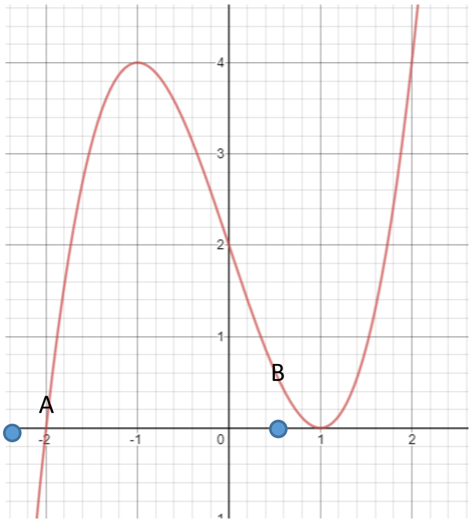
number of iterations = 27

II. Steffensen’s Method with Fixed Point Iteration Method

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

$$\hat{p}_n = p - \frac{(p_{n+1}-p_n)^2}{p_{n+2}-2p_{n+1}+p_n} \quad ,n = 0,1,2,\ldots$$

$$g(p_n) = p_n - \frac{{p_n}^3-3p_n+2}{3{p_n}^2-3} \quad ,n=0,1,2,\ldots$$



For the simple root p = -2, we use point A= p_0 = -2.4,

p_n	$g(p_n)$	p_n(cap) - Aitken Method	Stop? 1 - 'Yes"; 0 - 'No'
-2.400000000	-2.076190476	-	-
-2.076190476	-2.003596011	-	0
-2.003596011	-	-1.982618142	0
-1.982618142	-2.000204982	-	0
-2.000204982	-2.000000028	-	0
-2.000000028	-	-2.000002389	0
-2.000002389	-2.000000000	-	0
-2.000000000	-2.000000000	-	0
-2.000000000			1

p^* = -2.00000000 (up to 8 decimal places)

number of iterations = 2

For the double root p = 1, we use point B= p_0 = 0.5,

p_n	$g(p_n)$	p_n (cap) - Aitken Method	Stop? 1 - 'Yes"; 0 - 'No'
0.500000000	0.777777778	-	-
0.777777778	0.893518519	-	0
0.893518519	-	0.976190476	0
0.976190476	0.988143048	-	0
0.988143048	0.994083310	-	0
0.994083310	-	0.999952385	0
0.999952385	0.999976193	-	0

0.999976193	0.999988096	-	0
0.999988096	-	1.000000000	0
1.000000000	1.000000000	-	0
1.000000000			1

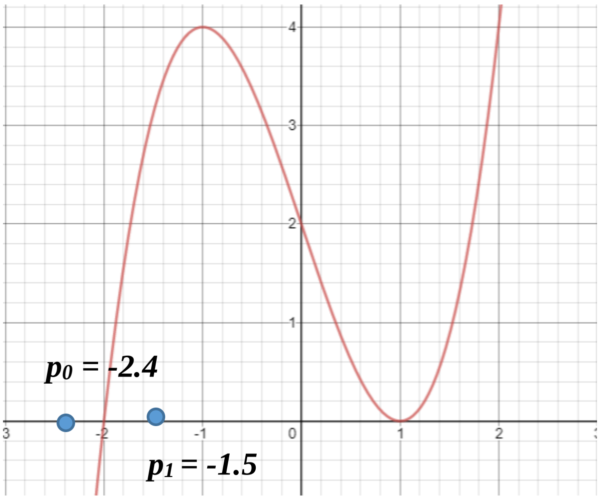
p* = 1.00000000 (up to 8 decimal places)

number of iterations = 3

III. Secant Method

f(x) = x^3 - 3x + 2

$$\boxed{p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1}-p_{n-2})}{f(p_{n-1})-f(p_{n-2})} \quad , n = 2,3, \dots}$$

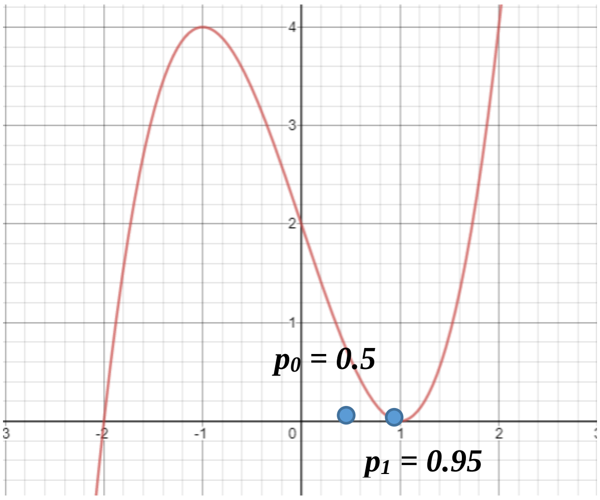


For the simple root p = -2, we use p_0 = -2.4 & p_1 = -1.5,

n	p_{n-2}	p_{n-1}	p_n	Stop? 1 for 'Yes' and 0 for 'No'
2	-2.400000000	-1.500000000	-1.862950058	0
3	-1.500000000	-1.862950058	-2.066635623	0
4	-1.862950058	-2.066635623	-1.993697173	0
5	-2.066635623	-1.993697173	-1.999728243	0
6	-1.993697173	-1.999728243	-2.000001146	0
7	-1.999728243	-2.000001146	-2.000000000	0
8	-2.000001146	-2.000000000	-2.000000000	1

p* = -2.00000000 (up to 8 decimal places)

number of iterations = 7



For the double root p = 1, we use $p_0 = 0.5$ & $p_1 = 0.95$,

n	p_{n-2}	p_{n-1}	p_n	Stop? 1 for 'Yes' and 0 for 'No'
2	0.500000000	0.950000000	0.955373406	0
3	0.950000000	0.955373406	0.976609475	0
4	0.955373406	0.976609475	0.984733193	0
5	0.976609475	0.984733193	0.990791186	0
6	0.984733193	0.990791186	0.994267017	0
7	0.990791186	0.994267017	0.996470869	0
8	0.994267017	0.996470869	0.997817164	0
9	0.996470869	0.997817164	0.998651945	0
10	0.997817164	0.998651945	0.999166849	0
11	0.998651945	0.999166849	0.999485174	0
12	0.999166849	0.999485174	0.999681833	0
13	0.999485174	0.999681833	0.999803372	0
14	0.999681833	0.999803372	0.999878480	0
15	0.999803372	0.999878480	0.999924898	0
16	0.999878480	0.999924898	0.999953585	0
17	0.999924898	0.999953585	0.999971314	0
18	0.999953585	0.999971314	0.999982271	0
19	0.999971314	0.999982271	0.999989043	0
20	0.999982271	0.999989043	0.999993228	0
21	0.999989043	0.999993228	0.999995815	0
22	0.999993228	0.999995815	0.999997413	0
23	0.999995815	0.999997413	0.999998401	0
24	0.999997413	0.999998401	0.999999012	0
25	0.999998401	0.999999012	0.999999389	0
26	0.999999012	0.999999389	0.999999623	0
27	0.999999389	0.999999623	0.999999767	0
28	0.999999623	0.999999767	0.999999856	0
29	0.999999767	0.999999856	0.999999911	0
30	0.999999856	0.999999911	0.999999945	0
31	0.999999911	0.999999945	0.999999966	0
32	0.999999945	0.999999966	0.999999979	0

33	0.999999966	0.999999979	0.999999987	0
34	0.999999979	0.999999987	0.999999995	0
35	0.999999987	0.999999995	0.999999995	1

p* = 1.00000000   (up to 8 decimal places)

number of iterations = 34

IV. Muller Methods

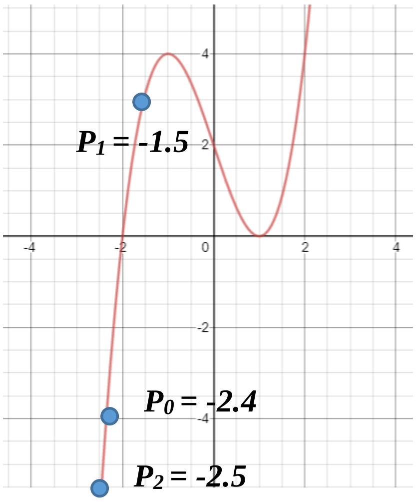
f(x) = $x^3 - 3x + 2$

$$c = f(P_n), n = 2,3,4,\ldots$$

$$b = \frac{(P_{n-2} - P_n)^2[f(P_{n-1}) - f(P_n)] - (P_{n-1} - P_n)^2[f(P_{n-2}) - f(P_n)]}{(P_{n-2} - P_n)(P_{n-1} - P_n)(P_{n-2} - P_{n-1})}, n = 2,3,4,\ldots$$

$$a = \frac{(P_{n-1} - P_n)\ [f(P_{n-2}) - f(P_n)] - (P_{n-2} - P_n)\ [f(P_{n-1}) - f(P_n)]}{(P_{n-2} - P_n)(P_{n-1} - P_n)(P_{n-2} - P_{n-1})}, n = 2,3,4,\ldots$$

$$P_{n+1} = P_n - \frac{2c}{b + (sign(b))\sqrt{b^2 - 4ac}}, n = 2,3,4,\ldots$$



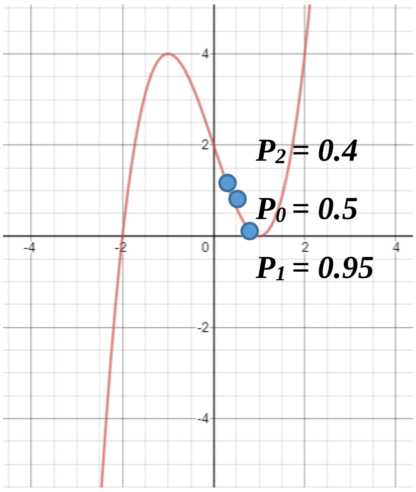
For double root p = -2, we use P_0 = -2.4, P_1 = -1.5, P_2 = -2.5,

n	p_{n-2}	p_{n-1}	p_n	a	b
2	-2.400000000	-1.500000000	-2.500000000	-6.400000000	15.650000000
3	-1.500000000	-2.500000000	-2.010731134	-6.010731134	9.379003928
4	-2.500000000	-2.010731134	-2.000289976	-6.511021110	8.998262413
5	-2.010731134	-2.000289976	-1.999999827	-6.011020937	8.999994811
6	-2.000289976	-1.999999827	-2.000000000	-6.000278459	9.000000003

n	c	p_{n+1}	Stop the iteration?
2	-6.125000000	-2.010731134	-
3	-0.097272389	-2.000289976	0
4	-0.002610289	-1.999999827	0
5	0.000001557	-2.000000000	0
6	0.000000000	-2.000000000	1

p* = -2.00000000 (up to 8 decimal places)

number of iterations = 5



For double root p = 1, we use $P_0 = 0.5$, $P_1 = 0.95$, $P_2 = 0.4$,

n	p_{n-2}	p_{n-1}	p_n	a	b
2	0.500000000	0.950000000	0.400000000	1.850000000	-2.575000000
3	0.950000000	0.400000000	0.964364288	2.314364288	-0.218111250
4	0.400000000	0.964364288	0.987116550	2.351480838	-0.090160978
5	0.964364288	0.987116550	0.993770444	2.945251282	-0.037456580
6	0.987116550	0.993770444	0.999135381	2.980022376	-0.005249949
7	0.993770444	0.999135381	0.999862780	2.992768605	-0.000827697
8	0.999135381	0.999862780	0.999985312	2.998983475	-0.000088234
9	0.999862780	0.999985312	0.999999260	2.999847219	-0.000004444
10	0.999985312	0.999999260	0.999999977	2.999973783	-0.000000136

n	c	p_{n+1}	Stop the iteration?
2	0.864000000	0.964364288	-
3	0.003764458	0.987116550	0
4	0.000495811	0.993770444	0
5	0.000116180	0.999135381	0
6	0.000002242	0.999862780	0
7	0.000000056	0.999985312	0
8	0.000000001	0.999999260	0
9	0.000000000	0.999999977	0
10	0.000000000	0.999999977	1

$p^* = 0.\ 99999998$ (up to 8 decimal places)

number of iterations = 9

Q1. (c)

Comment on efficiency:

Root Finding Method	Number of iteration to find simple root p = -2	Number of iteration to find double root p = 1
I. Newton Method	5	27
III. Secant Method	7	34
IV. Muller Methods	5	9

For finding the simple root p = -2, Muller Methods and Newton Method are both having the least number of iteration to obtain the approximate solution. Secant Method has gone through 7 iterations which is slightly more than Newton Method.

For finding the double root p = 1, Muller Methods is the most efficient method which only uses 9 iterations to obtain the solution. It is followed by Newton Method and Secant Method the last.

For the overall performance, Muller Method is considered as the most efficient method in this example because it is suitable to determine the approximated solution for a parabola or a curve.

Root Finding Method	Number of iteration to find simple root p = -2	Number of iteration to find double root p = 1
I. Newton Method	5	27
II. Steffensen's Method (Newton Method)	6	7

For finding the simple root p = -2, Steffensen's Method does not help to increase the rate of convergence as it uses 6 iterations compared with Newton Method which only uses 5 iterations.

For finding the double root p = 1, Steffensen's Method successfully increases the rate of convergence by using 7 iterations to obtain the solution rather than using 27 iterations.

Q2. (a)

Lagrange Polynomial	**Newton Divided-Difference Polynomial**	**Hermite Polynomial**	**Cubic Spline Polynomial**
It passes through all the data points available over an interval.	It passes through all the data points available over an interval.	It passes through all the data points available over an interval.	It passes through all the data points available over an interval.
All the data points are not required to be sorted out with respect to coordinate x.	All the data points are required to be sorted out with respect to coordinate x.	All the data points are not required to be sorted out with respect to coordinate x.	All the data points are required to be sorted out with respect to coordinate x.
-	-	First derivative of Hermite polynomial is the same as the first derivative of f(x) at all data points given.	It involves the piecewise-polynomial approximation where the cubic polynomial is used for each successive pair of nodes.

Q2. (b)

(i) Divded Difference Table:

x	f(x)	First Divided Differences	Second Divided Differences	Third Divided Differences	Forth Divided Differences	Fifth Divided Differences
0.00000	1.00000					
		1.10700				
0.20000	1.22140		0.61275			
		1.35210		0.22625		
0.40000	1.49182		0.74850		0.06198	
		1.65150		0.27583		2.46927
0.60000	1.82212		0.91400		2.53125	
		2.01710		2.30083		
0.80000	2.22554		2.29450			
		2.93490				
1.00000	2.81252					

(ii)

1. Newton's Forward-Difference

$f(0.1) = P_5(0 + 0.1)$

$h = 0.2, sh = s(0.2) = 0.1 \qquad \Rightarrow \qquad s = 0.5$

$$\begin{aligned} P_5(0 + 0.1) &= 1.00000 + 0.1(1.10700) + 0.2^2(0.5)(-0.5)(0.61275) \\ &\quad +0.2^3(0.5)(-0.5)(-1.5)(0.22625) \\ &\quad +0.2^4(0.5)(-0.5)(-1.5)(-2.5)(0.06198) \\ &\quad +0.2^5(0.5)(-0.5)(-1.5)(-2.5)(-3.5)(2.46927) \\ &= 1.10775 \end{aligned}$$

2. Newton's Centrered-Difference

$f(0.45) = P_5(0.4 + 0.05)$

$h = 0.2, sh = s(0.2) = 0.05 \qquad \Rightarrow \qquad s = 0.25$

$$\begin{aligned} P_n(0.4 + 0.05) &= 1.49182 + 0.05\left(\frac{1.35210+1.65150}{2}\right) + 0.25^2(0.2^2)(0.74850) \\ &\quad +0.25(0.25^2 - 1)(0.2^3)\left(\frac{0.22625+0.27583}{2}\right) \\ &\quad +0.25^2(0.2^4)(0.25^2 - 1)(0.06198) \end{aligned}$$

$$= 1.56830$$

3. Newton's Backward-Difference

$f(0.9) = P_5(1 - 0.1)$

$h = 0.2, sh = s(0.2) = -0.1 \quad \Rightarrow \quad s = -0.5$

$$\begin{aligned}
P_5(1 - 0.1) &= 2.81252 - 0.1(2.93490) + 0.2^2(-0.5)(0.5)(2.29450) \\
&\quad +0.2^3(-0.5)(0.5)(1.5)(2.30083) \\
&\quad +0.2^4(-0.5)(0.5)(1.5)(2.5)(2.53125) \\
&\quad +0.2^5(-0.5)(0.5)(1.5)(2.5)(3.5)(2.46927) \\
&= 2.48279
\end{aligned}$$

Q3.

$7x_1 - 2x_2 + x_3 + 2x_4 = 3$

$2x_1 + 8x_2 + 3x_3 + x_4 = -2$

$-x_1 + 5x_3 + 2x_4 = 5$

$2x_2 - x_3 + 4x_4 = 4$

(a) **Jacobi Method**

$X_1^{(k)} = \frac{1}{7}\ [2X_2^{(k-1)} - X_3^{(k-1)} - 2X_4^{(k-1)} + 3]$

$X_2^{(k)} = \frac{1}{8}\ [-2X_1^{(k-1)} - 3X_3^{(k-1)} - X_4^{(k-1)} - 2]$

$X_3^{(k)} = \frac{1}{5}\ [X_1^{(k-1)} - 2X_4^{(k-1)} + 5]$

$X_4^{(k)} = \frac{1}{4}\ [-2X_2^{(k-1)} + X_3^{(k-1)} + 4]$ where k = 1, 2, 3, …

n	$X_1^{(k)}$	$X_2^{(k)}$	$X_3^{(k)}$	$X_4^{(k)}$	Stop iteration? '1' for Yes, '0' for No.
0	0.00000	0.00000	0.00000	0.00000	-
1	0.42857	-0.25000	1.00000	1.00000	0
2	-0.07143	-0.85714	0.68571	1.37500	0
3	-0.30714	-0.66116	0.43571	1.60000	0
4	-0.27972	-0.53661	0.29857	1.43951	0
5	-0.17869	-0.47197	0.36825	1.34295	0
6	-0.14258	-0.51129	0.42708	1.32805	0
7	-0.15797	-0.54052	0.44026	1.36242	0
8	-0.17802	-0.54591	0.42344	1.38032	0
9	-0.18227	-0.53683	0.41227	1.37881	0
10	-0.17765	-0.53138	0.41202	1.37148	0
11	-0.17396	-0.53153	0.41588	1.36870	0
12	-0.17376	-0.53355	0.41773	1.36973	0
13	-0.17490	-0.53442	0.41735	1.37121	0
14	-0.17552	-0.53418	0.41654	1.37155	0
15	-0.17543	-0.53377	0.41628	1.37123	0
16	-0.17518	-0.53365	0.41642	1.37095	0
17	-0.17509	-0.53373	0.41658	1.37093	0
18	-0.17513	-0.53381	0.41661	1.37101	0
19	-0.17518	-0.53382	0.41657	1.37106	1
20	-0.17519	-0.53380	0.41654	1.37105	1
21	-0.17518	-0.53379	0.41654	1.37104	1
22	-0.17517	-0.53379	0.41655	1.37103	1
23	-0.17517	-0.53379	0.41655	1.37103	1

$X_1 = -0.1752, X_2 = -0.5338, X_3 = 0.4166, X_4 = 1.3710$

Number of Iterations = 19

(b) **Gauss Seidel Method**

$$X_1^{(k)} = \frac{1}{7}\ [2X_2^{(k-1)} - X_3^{(k-1)} - 2X_4^{(k-1)} + 3]$$
$$X_2^{(k)} = \frac{1}{8}\ [-2X_1^{(k)} - 3X_3^{(k-1)} - X_4^{(k-1)} - 2]$$
$$X_3^{(k)} = \frac{1}{5}\ [X_1^{(k)} - 2X_4^{(k-1)} + 5]$$
$$X_4^{(k)} = \frac{1}{4}\ [-2X_2^{(k)} + X_3^{(k)} + 4]$$

where k = 1, 2, 3, ….

n	$X_1^{(k)}$	$X_2^{(k)}$	$X_3^{(k)}$	$X_4^{(k)}$	Stop iteration? '1' for Yes, '0' for No.
0	0.00000	0.00000	0.00000	0.00000	-
1	0.42857	-0.35714	1.08571	1.45000	0
2	-0.24286	-0.77768	0.37143	1.48170	0
3	-0.27003	-0.50699	0.35332	1.34182	0
4	-0.15014	-0.51269	0.43324	1.36465	0
5	-0.16970	-0.54062	0.42020	1.37536	0
6	-0.17888	-0.53477	0.41408	1.37091	0
7	-0.17506	-0.53288	0.41662	1.37059	0
8	-0.17480	-0.53386	0.41680	1.37113	0
9	-0.17525	-0.53388	0.41650	1.37106	0
10	-0.17520	-0.53377	0.41654	1.37102	0
11	-0.17516	-0.53379	0.41656	1.37103	1
12	-0.17517	-0.53380	0.41655	1.37104	1
13	-0.17517	-0.53379	0.41655	1.37103	1
14	-0.17517	-0.53379	0.41655	1.37103	1

$X_1 = -0.1752, X_2 = -0.5338, X_3 = 0.4166, X_4 = 1.3710$

Number of Iterations = 11

(c) **Successive Over-Relaxation Method (SOR)**

$X_1^{(k)} = 0.8[\ \frac{1}{7}\ (2X_2^{(k-1)} - X_3^{(k-1)} - 2X_4^{(k-1)} + 3)] + 0.2\ x_1^{(k-1)}$

$X_2^{(k)} = 0.8[\ \frac{1}{8}\ (-2X_1^{(k)} - 3X_3^{(k-1)} - X_4^{(k-1)} - 2)] + 0.2\ x_2^{(k-1)}$

$X_3^{(k)} = 0.8[\ \frac{1}{5}\ (X_1^{(k)} - 2X_4^{(k-1)} + 5)] + 0.2\ x_3^{(k-1)}$

$X_4^{(k)} = 0.8[\ \frac{1}{4}\ (-2X_2^{(k)} + X_3^{(k)} + 4)] + 0.2\ x_4^{(k-1)}$ where k = 1, 2, 3,…

n	$X_1^{(k)}$	$X_2^{(k)}$	$X_3^{(k)}$	$X_4^{(k)}$	Stop iteration? '1' for Yes, '0' for No.
0	0.00000	0.00000	0.00000	0.00000	-
1	0.34286	-0.26857	0.85486	1.07840	0
2	0.00585	-0.61918	0.62682	1.38872	0
3	-0.18656	-0.61344	0.45113	1.41335	0
4	-0.20928	-0.55751	0.40447	1.38657	0
5	-0.18958	-0.53358	0.40686	1.37212	0
6	-0.17715	-0.53056	0.41395	1.36944	0
7	-0.17417	-0.53241	0.41670	1.37019	0
8	-0.17448	-0.53362	0.41696	1.37088	0
9	-0.17500	-0.53390	0.41671	1.37108	0
10	-0.17519	-0.53386	0.41657	1.37107	0
11	-0.17520	-0.53381	0.41654	1.37105	0
12	-0.17518	-0.53379	0.41654	1.37103	1
13	-0.17517	-0.53379	0.41655	1.37103	1
14	-0.17517	-0.53379	0.41655	1.37103	1

$X_1 = -0.1752, X_2 = -0.5338, X_3 = 0.4165, X_4 = 1.3710$

Number of Iterations = 12

Method	**No. of iteration required**
Jacobi method	19
Gauss-Seidel method	11
Successive Over-Relaxation method	12

From the table above, we conclude that the Gauss-Seidel method is the most efficient method among the three methods. This is because the number of iteration required for Gauss-Seidel method is the least compare with others. In another word, Gauss-Seidel method converges faster compared with Jacobi method and the Successive Over-Relaxation method. Furthermore, the Successive Over-Relaxation method is more efficient than the Jacobi method as it requires lesser iteration to obtain the solution.

Moreover, we also conclude that the efficiency for Gauss-Seidel method and Successive Over-Relaxation method is almost the same. It is because the ω that used to calculate the SOR method is close to 1.