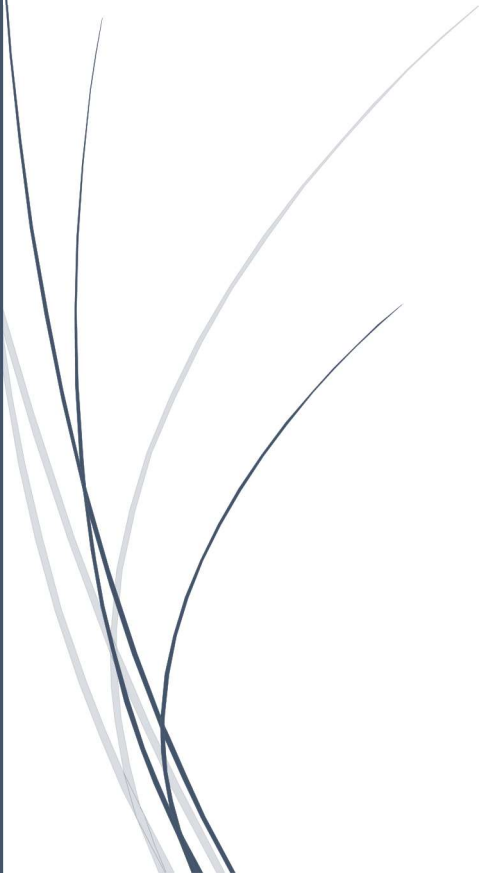


A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the word 'Report' in white text.

Report

# Numerical Project

## Part 1

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and curve upwards and to the right.

**AbdElmoneam elrawy. (26)**  
**Karim Ibrahim Mustafa. (30)**  
**Mahmoud hamed. (46)**  
**Mahmoud fathy. (47)**

## PSEDOCODE:

- **bisection method**

- 1- Get function, initial guesses.
- 2- Calculate y of each x.
- 3-  $Ys[] \leftarrow y$       append y in ys
- 4- Determine the x upper and the x lower
- 5- Check the converge of this function  
If  $Ys[1]*Ys[0]>0$ :  
Return false
- 6-  $Xus[] \leftarrow x_u$  &  $Xls[] \leftarrow x_l$       append  $x_l$  &  $x_u$  in their array
- 7-  $X_k = (x_u + x_l)/2$       calculate  $x_k$
- 8-  $Xks[] \leftarrow x_k$       append  $x_k$
- 9-  $F = f(x_k)$  &  $F_2 = f(x_l)$       calculate the function of  $x_k$  &  $x_l$
- 10- If  $f*f_2 > 0$ :  
 $X_l = x_k$   
Else:  
 $X_u = x_k$
- 11- Calculate the error
- 12- Repeat from step 2

- **False Position method:**

- 1- Get function, initial guesses.

- 2- Calculate y of each x.
  - 3-  $Ys[] \leftarrow y$       append y in ys
  - 4- Determine the x upper and the x lower
  - 5- Check the converge of this function  
     If  $Ys[1]*Ys[0]>0$ :      Return false
  - 6-  $Xus[] \leftarrow x_u$  &  $Xls[] \leftarrow x_l$     append  $x_l$  &  $x_u$  in their array
  - 7- Calculate the  $f(x_l)$  &  $f(x_u)$
  - 8-  $X_k = (x_l*f(x_u)-x_u*f(x_l))/((f(x_u)-f(x_l)))$       calculate  $x_k$
  - 9-  $Xks[] \leftarrow x_k$       append  $x_k$
  - 10- If  $f(x_k) < 0$ :     $X_l = x_k$   
     Else:       $X_u = x_k$
  - 11- Calculate the error
  - 12- Repeat from step 2
- 

- **Secent method:**

Given two points  $x(i)$ ,  $x(i-1)$ , eps, max\_iter :

- Error  $\leftarrow$  large value
- Iteration  $\leftarrow 0$
- While error > eps and iteration < max\_iter then do:
  - $X(i+1) \leftarrow x(i)-(F(x_0)(x(i-1)-x(i)))/F(x(i-1)-F(x(i)))$
  - Calculate absolute error
  - Error  $\leftarrow X(i+1) - X(i)$
  - Iteration  $\leftarrow$  iteration +1
- End algorithm

- **fixed point method:**

given the function ,initial point  $x_0$  , eps and max iteration:

- $g(x) \leftarrow$  function + x

- we assume  $g(x)$  take fixed form that is constructed by adding only  $x$  in both sides of  $f(x)$
  - $\text{error} \leftarrow \text{large value}$
  - $\text{iteration} \leftarrow 0$
  - $x_r \leftarrow x_0$
  - while  $\text{error} > \text{eps}$  and  $\text{iteration} < \text{max iteration}$  then:
    - $x_{r\_old} \leftarrow x_r$
    - $x_r \leftarrow g(x_{r\_old})$
    - $\text{iteration} \leftarrow \text{iteration} + 1$
  - end
- 

- **Bierge-Vieta:**

- given function ,initial point  $x_0$  , eps and max iteration:
- $x \leftarrow \text{initial } x_0$
- $\text{error} \leftarrow \text{large value}$
- get coefficients of polynomial and store them in list A
- $\text{iteration} \leftarrow 0$
- declare two lists B and C
- while  $\text{error} > \text{eps}$  and  $\text{iteration} < \text{max iteration}$  then:
  - $B[0] \text{ and } C[0] \leftarrow A[0]$
  - For  $i$  in range size of A:
    - $B[i] \leftarrow A[i] + x * B[i-1]$
    - If not last element in B:
      - $C[i] \leftarrow B[i] + x * C[i-1]$
  - $X_{old} \leftarrow x$
  - $x \leftarrow x - (B[\text{last}]/C[\text{last}])$
  - Calculate absolute error
  - $\text{Error} \leftarrow x - x_{old}$
- End

- 
- We implement the **Dekker's method** as general algorithm:

It search for the right interval  $[a,b]$  such that the root is between it .

We choose it because it fast as secent and accurate and not able to diverge like bisection .

The idea to combine the bisection method with the secant method goes

back to Dekker (1969).

Suppose that we want to solve the equation  $f(x) = 0$ . As with the bisection

method, we need to initialize Dekker's method with two points, say  $a_0$  and

$b_0$ , such that  $f(a_0)$  and  $f(b_0)$  have opposite signs. If  $f$  is continuous on  $[a_0,$

$b_0]$ , the intermediate value theorem guarantees the existence of a solution

between  $a_0$  and  $b_0$ .

Three points are involved in every iteration:

We first calculate the secant method if it'll diverge we will calculate

bisection method

- 1- Get function
- 2- Calculate the interval  $[x_a, x_b]$
- 3- If first iteration  $x_{b-1} = x_a$
- 4- Calculate  $f(x_b), f(x_{b-1})$  of each  $x$ .
- 5- Get  $s$  by secant and calculate  $m$  from bisection  
  
If  $x_r$  is between  $m$  and  $x_b$   
  
 $x_{b+1} = s$   
  
Else  
  
 $x_{b+1} = m$
- 6- If  $f(x_{b+1})$  and  $f(x_a)$  have same sign then  
  
 $x_{a+1} = x_b$
- 7- Calculate the error
- 8- Repeat from step 3

## **Data structure Used:**

- For all methods, we used :

- List to represent values of errors , roots in each iteration.
  - Table of lists that hold all information of each iteration needed in the method .
- 

## **Behavior Analysis:**

- For bisection and false position:

it will converge always if we choose the right interval (a,b).

But it will diverge if a and b are in the same side of the root

- Example 1:

If  $f(x)=x^2-2$  and given  $x_u=0, x_l=1$

The real root is 1.414 so it isn't in that interval so it'll diverge

○ Example 2:

If  $f(x) = x^2 - 2$  and given  $x_u = 1, x_l = 2$

The real root is 1.414 so it is in that interval so it'll converge

---

- **For newton:**

it will diverge if

$F'(x) = 0$  the root is maximum or minimum point

$F''(x) = 0$  there are coop point

But it's so fast of calculation and converge fast if it'll converge

Example1:

If  $f(x) = x^2 - 2$  and  $x$  guess  $= 0$  has max point

$F'(X) = 0$



Then that will diverge because of division by zero

Example2:

If  $f(x)=x^2-2$  and  $x_{\text{guess}}=1$ ,  $F'(1) \neq 0$  Then that will converge

---

- Fixed Point Iteration Method:
  - We include  $x$  to the function to get  $g(x)$  and it converges only when the derivative of  $g(x)$  is less than 1.
  - When the method converges, the error is roughly proportional to or less than the error of the previous step
  - **The main backward** of this implementation of this method is that function must contain term of order one and using small value of initial  $x$  to make sure that method is converge.so, method will diverge in most cases unless consider this assumptions.
- Secant method:
  - uses a succession of roots of secant lines to better approximate a root of a function  $f$ . The secant method

can be thought of as a finite-difference approximation of **Newton's method**.

- The iterations of the secant method converge to a root of , if the initial values are sufficiently close to the root.
- If the initial values are not close enough to the root, then there is no guarantee that the secant method converges

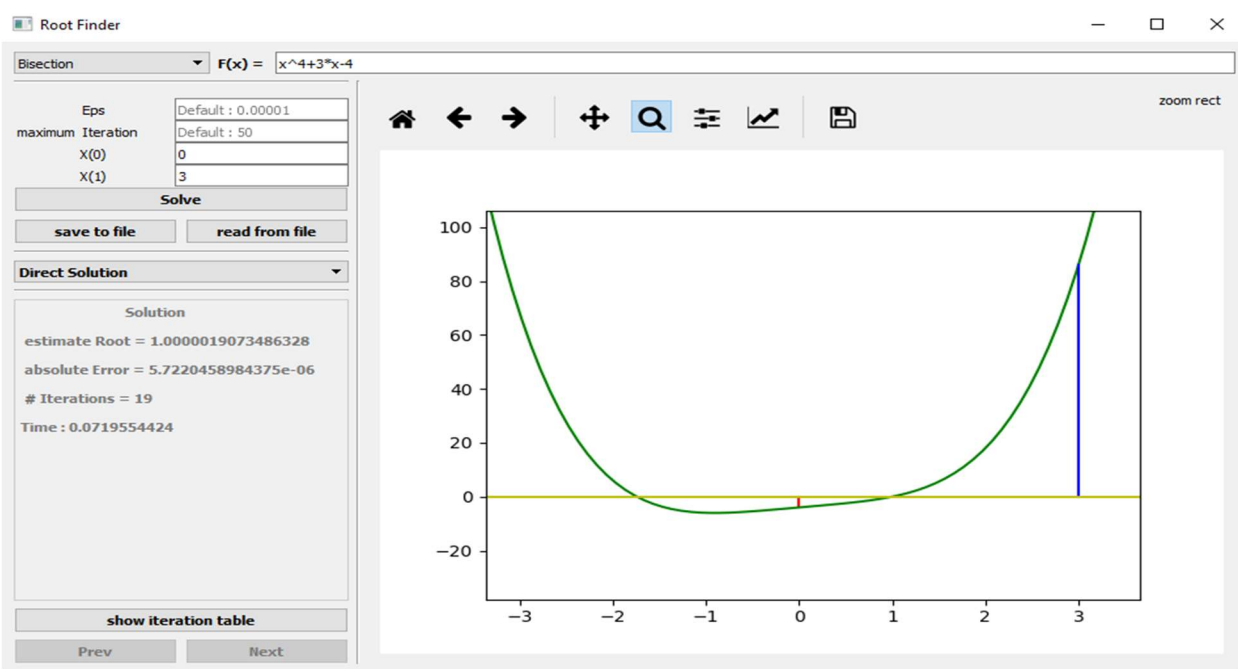
---

- Birge Vieta :

- Used to get root of polynomial equations using combinations of newton method and Horner method.
- Method don't use derivative as in newton method.
- It arrange coefficients of polynomial and use to compute corresponding values of B and C to estimate the root.

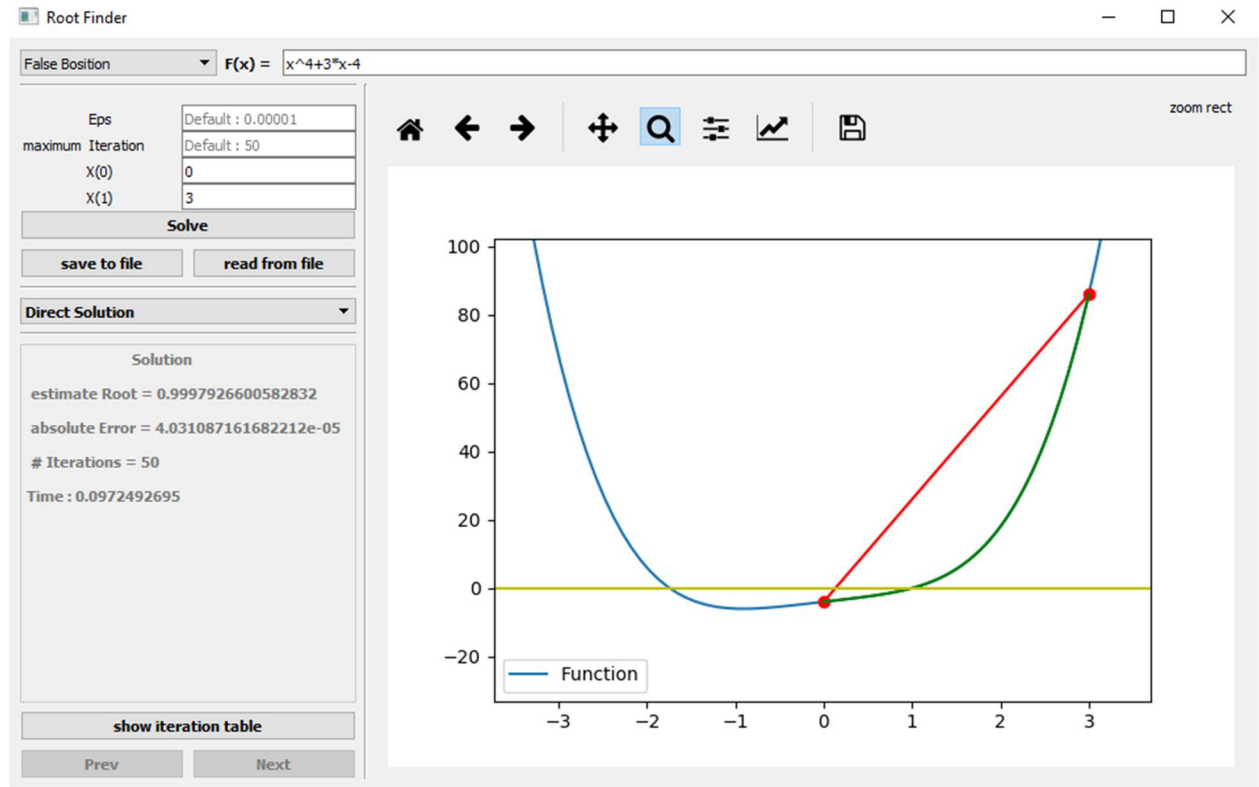
# Sample Runs:

- Bisection method:

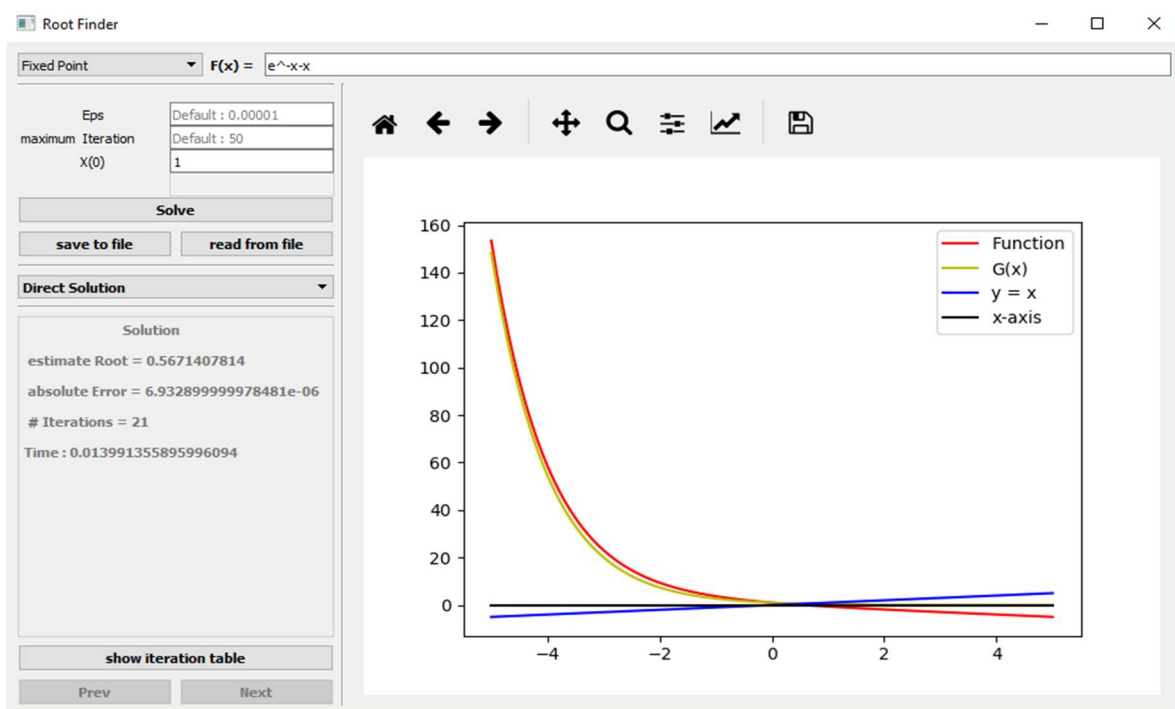


	xu	xl	Xr	F(Xr)	F(self.xl)	Error
1	3.0	0.0	1.5	5.5625	-4.0	1
2	1.5	0.0	0.75	-1.43359375	-4.0	0.75
3	1.5	0.75	1.125	0.9768066406	-1.43359375	0.375
4	1.125	0.75	0.9375	-0.4150238037	-1.43359375	0.1875
5	1.125	0.9375	1.03125	0.224732399	-0.4150238037	0.09375
6	1.03125	0.9375	0.984375	-0.1079253554	-0.4150238037	0.046875
7	1.03125	0.984375	1.0078125	0.055055622	-0.1079253554	0.0234375
8	1.0078125	0.984375	0.99609375	-0.0272524355	-0.1079253554	0.01171875
9	1.0078125	0.99609375	1.001953125	0.013694793	-0.0272524355	0.005859375
10	1.001953125	0.99609375	0.9990234375	-0.0068302192	-0.0272524355	0.0029296875
11	1.001953125	0.9990234375	1.00048828125	0.0034193997	-0.0068302192	0.00146484375
12	1.00048828125	0.9990234375	0.999755859375	-0.0017086268	-0.0068302192	0.000732421875
13	1.00048828125	0.999755859375	1.0001220703125	0.0008545816	-0.0017086268	0.0003662109375
14	1.0001220703125	0.999755859375	0.99993896484375	-0.0004272237	-0.0017086268	0.00018310546875
15	1.0001220703125	0.99993896484375	1.000030517578...	0.0002136286	-0.0004272237	9.1552734375e-05
16	1.000030517578...	0.99993896484375	0.999984741210...	-0.0001068101	-0.0004272237	4.57763671875e-05
17	1.000030517578...	0.999984741210...	1.000007629394...	5.34061e-05	-0.0001068101	2.288818359375e-05
18	1.000007629394...	0.999984741210...	0.999996185302...	-2.67028e-05	-0.0001068101	1.1444091796875e-05
19	1.000007629394...	0.999996185302...	1.000001907348...	1.33515e-05	-2.67028e-05	5.7220458984375e-06

- False Position:

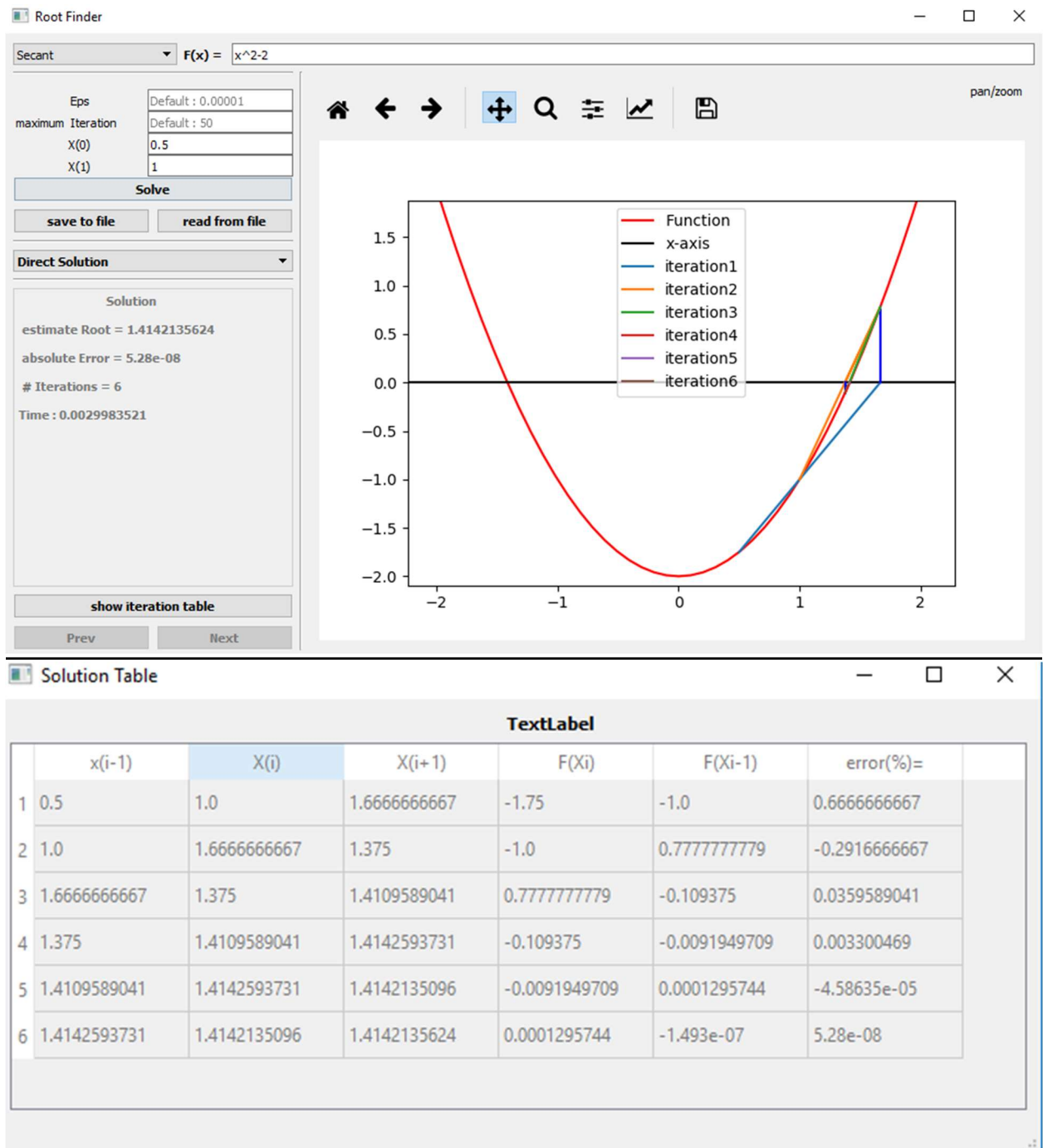


- Fixed Point:

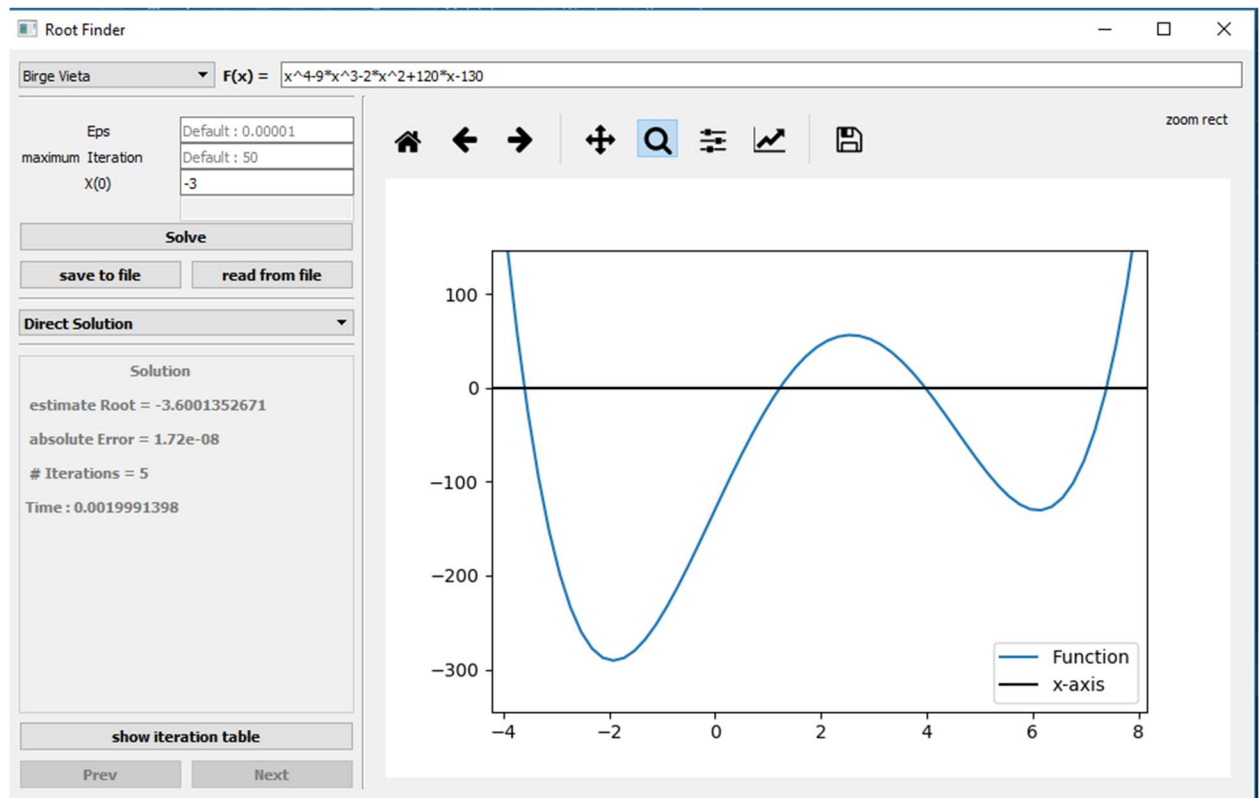


	xr_prev	xr	absolute error
1	1.0	0.3678794412	0.6321205588
2	0.3678794412	0.6922006275	0.3243211863
3	0.6922006275	0.5004735006	0.1917271269
4	0.5004735006	0.6062435351	0.1057700345
5	0.6062435351	0.545395786	0.0608477491
6	0.545395786	0.5796123355	0.0342165495
7	0.5796123355	0.5601154614	0.0194968741
8	0.5601154614	0.5711431151	0.0110276537
9	0.5711431151	0.5648793474	0.0062637677
10	0.5648793474	0.568428725	0.0035493776
11	0.568428725	0.5664147332	0.0020139918
12	0.5664147332	0.5675566373	0.0011419041
13	0.5675566373	0.5669089119	0.0006477254
14	0.5669089119	0.5672762322	0.0003673203
15	0.5672762322	0.5670678984	0.0002083338
16	0.5670678984	0.5671860501	0.0001181517
17	0.5671860501	0.5671190401	6.701e-05
18	0.5671190401	0.567157044	3.80039e-05
19	0.567157044	0.5671354902	2.15538e-05
20	0.5671354902	0.5671477143	1.22241e-05
21	0.5671477143	0.5671407814	6.9329e-06

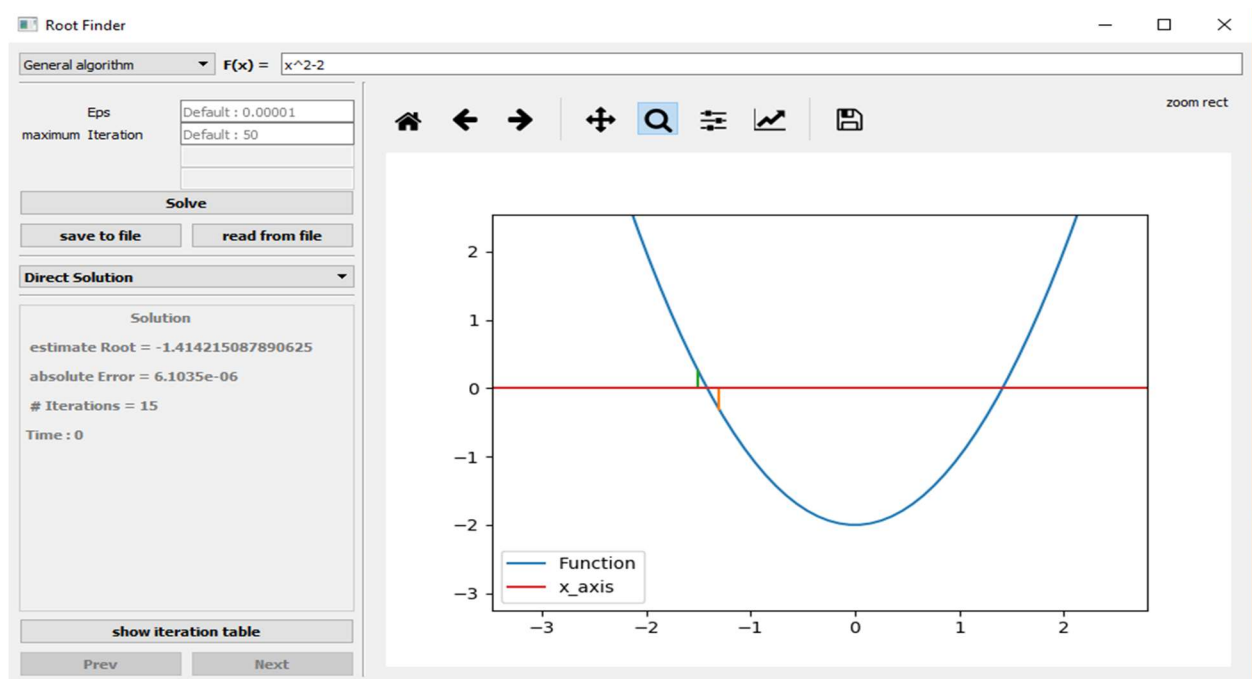
- Secant Method:



- Birge Vieta:



- General Algorithm:





Solution Table

	TextLabel						
	xa	xb	Xr	F(Xr)	F(xb1)	F(xb2)	Error
1	-1.5	-1.4	-1.45	0.1025	-0.04	0.25	0.05
2	-1.4	-1.45	-1.4249999999...	0.030625	0.1025	-0.04	0.025
3	-1.4	-1.4249999999...	-1.4124999999...	-0.00484375	0.030625	0.1025	0.0125
4	-1.4249999999...	-1.4124999999...	-1.4187499999...	0.0128515625	-0.00484375	0.030625	0.00625
5	-1.4124999999...	-1.4187499999...	-1.415625	0.0039941406	0.0128515625	-0.00484375	0.003125
6	-1.4124999999...	-1.415625	-1.4140625	-0.0004272461	0.0039941406	0.0128515625	0.0015625
7	-1.415625	-1.4140625	-1.41484375	0.0017828369	-0.0004272461	0.0039941406	0.00078125
8	-1.4140625	-1.41484375	-1.414453125	0.0006776428	0.0017828369	-0.0004272461	0.000390625
9	-1.4140625	-1.414453125	-1.4142578125	0.0001251602	0.0006776428	0.0017828369	0.0001953125
10	-1.4140625	-1.4142578125	-1.41416015625	-0.0001510525	0.0001251602	0.0006776428	9.76563e-05
11	-1.4142578125	-1.41416015625	-1.414208984375	-1.29485e-05	-0.0001510525	0.0001251602	4.88281e-05
12	-1.4142578125	-1.414208984375	-1.41423339843...	5.61053e-05	-1.29485e-05	-0.0001510525	2.44141e-05
13	-1.414208984375	-1.41423339843...	-1.41422119140...	2.15782e-05	5.61053e-05	-1.29485e-05	1.2207e-05
14	-1.414208984375	-1.41422119140...	-1.41421508789...	4.3148e-06	2.15782e-05	5.61053e-05	6.1035e-06

- All method Comparison:

