```
Bounding_Search();
function fun_val = Objective_Fun(x)
%fun_val = (x^2-1)^3-(2*x-5)^4 ;       %(a,b) =(-10,0) %

%fun_val =  2*exp(x) +2*x - x^3-8          %(a,b) =(-2,1) %

%fun_val = -4*x*sin(x)                     %(a,b) =(.5,pi) %

%fun_val = 2*(x-3)^2+ exp(.5*x^2);         %(a,b) =(-2,3) %

% fun_val = x^2-10*exp(.1*x);             %(a,b) =(-6,6) %

%fun_val = 15*x^2-20*sin(x);               %(a,b) =(-4,4) %

end
function Bounding_Search()
 fprintf('\nBounding Search Method');
 x0 = input('\n Choose intial guess of x between (a,b) = ');
delta = input('set delta value =');
 feval = 0;

 x2 = x0; x1 = x2 - abs(delta); x3 = x2 + abs(delta);
 f1 = Objective_Fun(x1);
 f2 = Objective_Fun(x2);
 f3 = Objective_Fun(x3);
 i = 1;
 feval= feval+3;
 if(f1<f2 && f2<f3)
     delta = -abs(delta);

 elseif(f3<f2 && f2<f1)
     delta = abs(delta);
     x1 = x3;
     f1 = f3;

 else
 fprintf('run programe with different value of intial guess')

 end


out = fopen('Bounding_search_iterations.out', 'w');

fprintf(out,'#It\tx1\tx2\tx3\tf(x1)\tf(x2)\tf(x3)\n');


while (f1<f2)
     fprintf(out,
'%d\t%4.2f\t%4.2f\t%4.2f\t%4.2f\t%4.2f\t%4.2f\n',i,x1,x2,x3,f1,f2,f3);
       f2 = f1;
       x3 = x2;
       x2 = x1;

       x1 = x2 + (2^i)*delta;
       f1 = Objective_Fun(x1);
```

```
       i = i+1;
        feval = feval + 1;
        if(f1>f2)
         break;
     end
 end
  fprintf('\n************************\n');
   fprintf('The minimum point lies between (%8.3f, %8.3f)', x1, x3);

 fprintf('\nTotal number of function evaluations:  %d\n', feval);

 fprintf(out, '\nTotal number of function evaluations: %d', feval);

     fprintf(out, '\nThe minimum point lies between (%8.3f, %8.3f)', x1,
x3);

end
```

Initial Guess	△=Value of increment	No. of iteration	No. of function evaluation	Bracketed Interval
Q.1 f(x) = (x^2-1)^3-(2*x-5)^4 (Since question is given for maximization so final answer of f(x) must be multiplied with minus likewise we will apply in other problem also)				
-8	**.1**	**5**	**8**	**(-6.500,-1.700)**
	.2	**4**	**7**	**(-6.600,-1.800)**
	.3	**3**	**6**	**(-7.100,-3.500)**
	.4	**3**	**6**	**(-6.800,-2.000)**
	.5	**2**	**5**	**(-7.500,-4.500)**
Q.2 f(X) = 2*exp(x) +2*x - x^3-8				
0	**.1**	**3**	**6**	**(-1.500,-.300)**
	.2	**2**	**5**	**(-1.400,-.200)**
	.3	**2**	**5**	**(-2.100,-.300)**
	.4	**2**	**5**	**(-2.800,-.400)**
	.5	**1**	**4**	**(-1.500,.000)**
Q.3 f(X) = -4*x*sin(x)				
3	**.1**	**3**	**6**	**(1.500,2.700)**
	.2	**2**	**5**	**(1.600,2.800)**
	.3	**2**	**5**	**(.900,2.700)**
	.4	**2**	**5**	**(0.200,2.600)**
	.5	**2**	**5**	**(-.500,2.500)**
Q.4 f(X) = 2*(x-3)^2+ exp(.5*x^2)				
1	**.1**	**3**	**6**	**(1.300,2.500)**
	.2	**2**	**5**	**(1.200,2.400)**
	.3	**1**	**4**	**(1.000,1.900)**
	.4	**1**	**4**	**(1.000,2.200)**
	.5	**1**	**4**	**(1.000,2.500)**
Q.5 f(X) = x^2-10*exp(.1*x)				
5	**.1**	**5**	**8**	**(-1.300,3.500)**
	.2	**4**	**7**	**(-1.200,3.600)**
	.3	**4**	**7**	**(2.900,4.300)**
	.4	**4**	**7**	**(2.200,7.400)**
	.5	**3**	**6**	**(-2.500,3.500)**
Q.6 f(X) = x^2-10*exp(.1*x)				
-3	**.1**	**5**	**8**	**(-1.500,3.300)**
	.2	**4**	**7**	**(-1.600,3.200)**
	.3	**4**	**7**	**(0.900,6.300)**
	.4	**3**	**6**	**(-1.800,3.000)**
	.5	**3**	**6**	**(-1.500,4.500)**

Conclusion of bounding phase method from programme and table

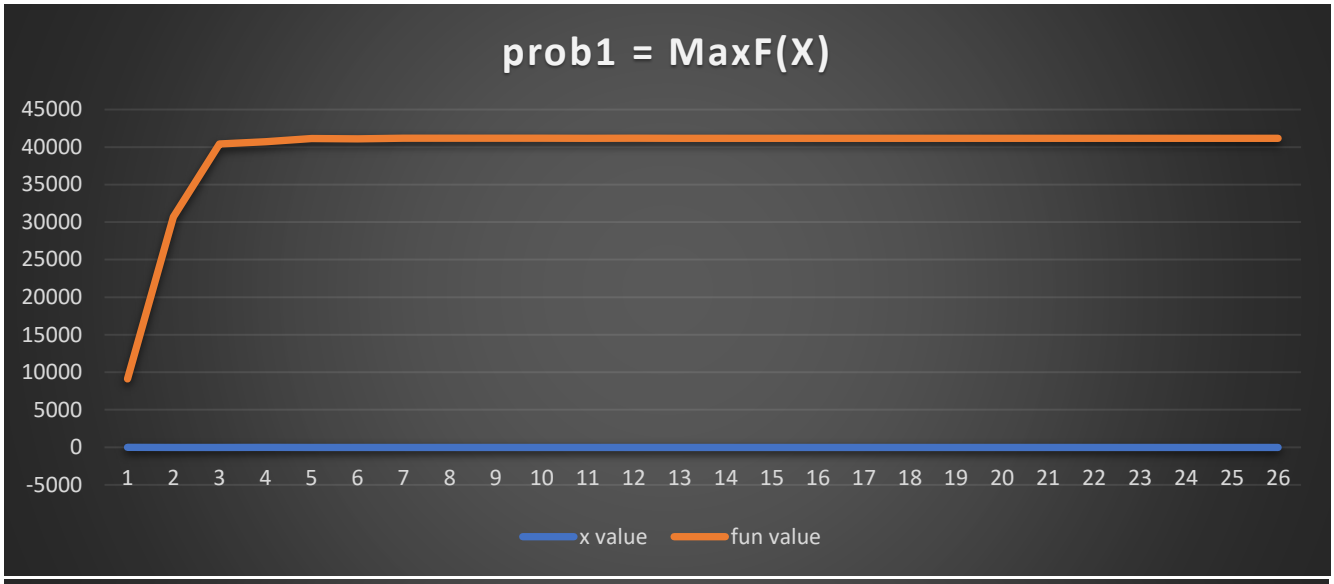
As the value of increment increases the no of function evaluation decreases but the bracketed interval increases. That means convergence rate is decreases. So this method is helpful for bracketing further we used any gradient method to get convergence of this function.

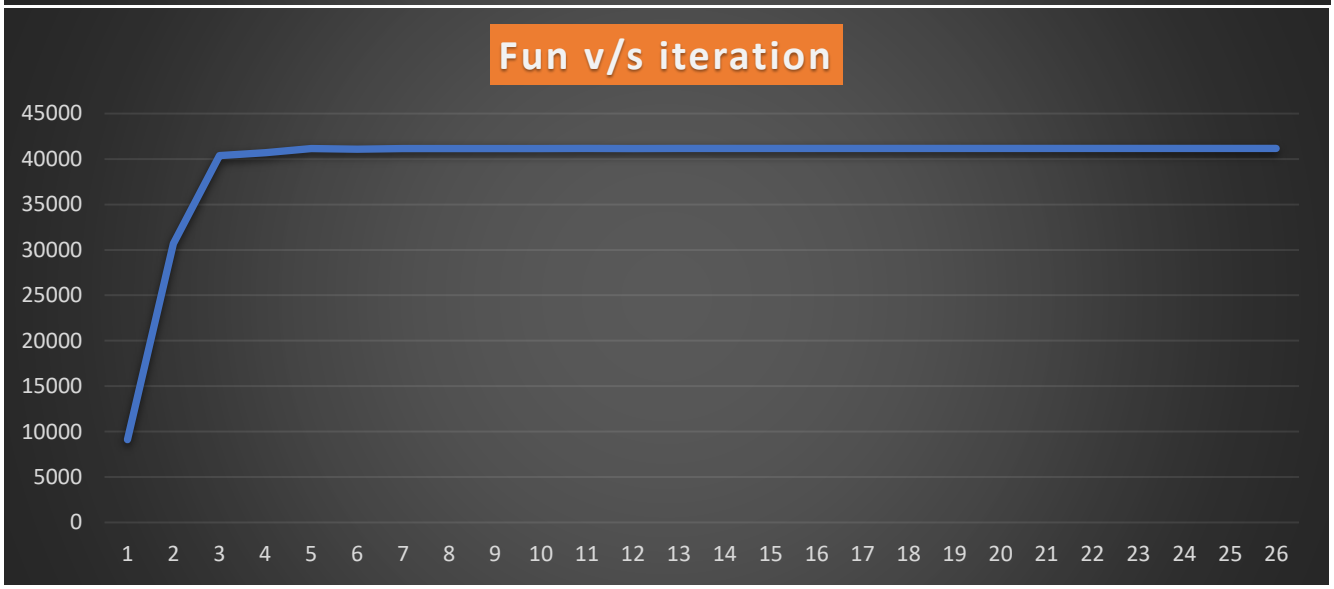
Bisection Method

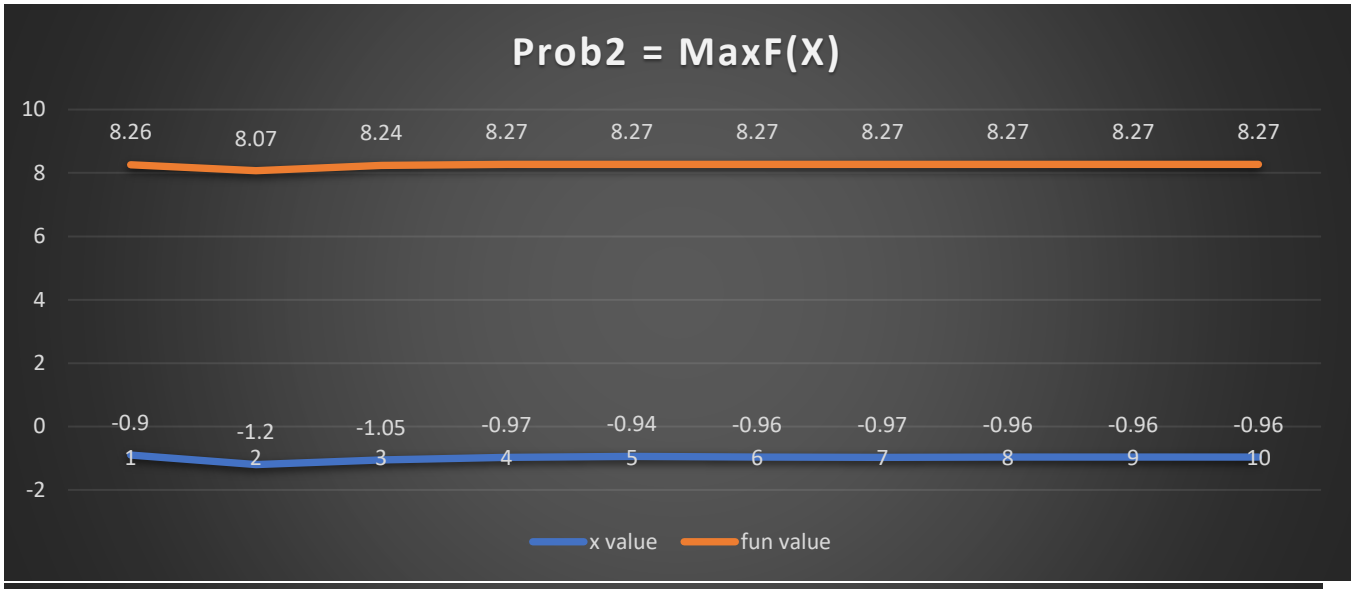
```
x1 = input('Enter x1 = lower limit of x =');
x3 = input('Enter x3 = upper limit of x =');
Bisection_search(x1,x3);

function fun_val = Objective_Fun(x) %test
fun_val = (x^2-1)^3-(2*x-5)^4 ;      %(a,b) =(-6.5,-1.7) %

%fun_val = 2*exp(x) +2*x - x^3-8;     %(a,b) =(-1.5,-.3) %

%fun_val = -4*x*sin(x)                %(a,b) =(1.5,2.7) %

%fun_val = 2*(x-3)^2+ exp(.5*x^2);     %(a,b) =(1.3,2.5) %

 %fun_val = x^2-10*exp(.1*x);          %(a,b) =(-1.3,3.5) %

%fun_val = 15*x^2-20*sin(x);            %(a,b) =(-1.5,3.3) %

end


function Bisection_search(x1,x3);
 fprintf('Bisection Search Method');

 delta_x = .0001; e = 10^-3;
 x1  = x1 + delta_x; x2 = x1 -delta_x;
 x3 = x3+delta_x; x4 = x3 - delta_x;
 z = (x1 + x3)/2; z1 = z + delta_x; z2 = z - delta_x;

 f1 = Objective_Fun(x1);

 f2 = Objective_Fun(x2);


 f3 = Objective_Fun(x3);


 f4 = Objective_Fun(x4);
 f5 =  Objective_Fun(z1);
 f6 =  Objective_Fun(z2);

 p = (f1-f2)/(2*delta_x);
 q = (f3-f4)/(2*delta_x);
 r = (f5-f6)/(2*delta_x);
 feval = 6;
 out = fopen('Bisection_search_iterations.out', 'w');

fprintf(out,'x1\tz\tx3\tf(x5)\tf(x6)\tr\n');

 while(abs(r)>=e)

fprintf(out,'%4.2f\t%4.2f\t%4.2f\t%4.2f\t%4.2f\t%4.2f\n',x1,z,x3,f5,f6,r);

 if(r>0)
     x1 = x1;
     x3 = z;
```

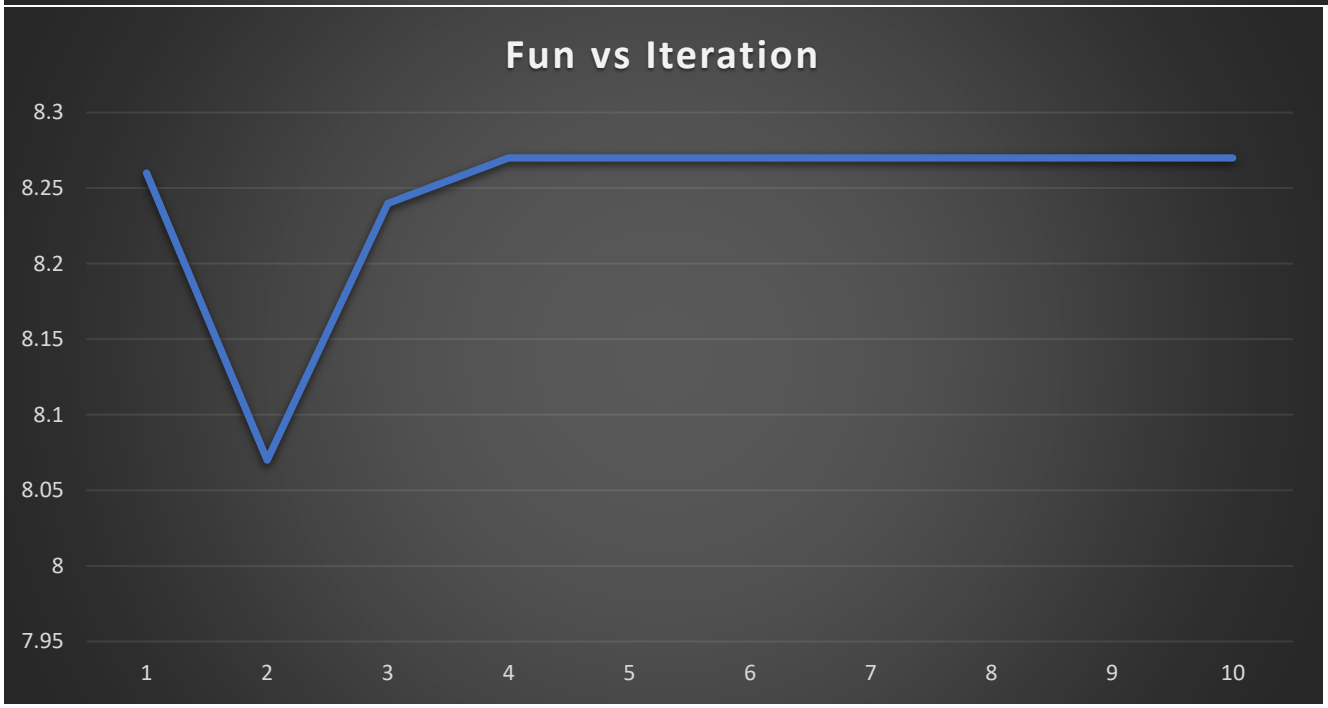
```
 else(r<0)
     x1 = z;
     x3 =x3;

 end

  z = (x1+x3)/2;
  z1 = z + delta_x; z2 = z - delta_x;
  f5 =  Objective_Fun(z1);
  f6 =  Objective_Fun(z2);
  r = (f5-f6)/(2*delta_x);

  feval= feval+2;
 end

  fprintf('\n************************\n');
   fprintf('The minimum point lies at (%8.3f)', z);

 fprintf('\nTotal number of function evaluations:  %d\n', feval);

 fprintf(out, '\nTotal number of function evaluations: %d', feval);

     fprintf(out, '\nThe minimum point lies at (%8.3f)',z);



end
```

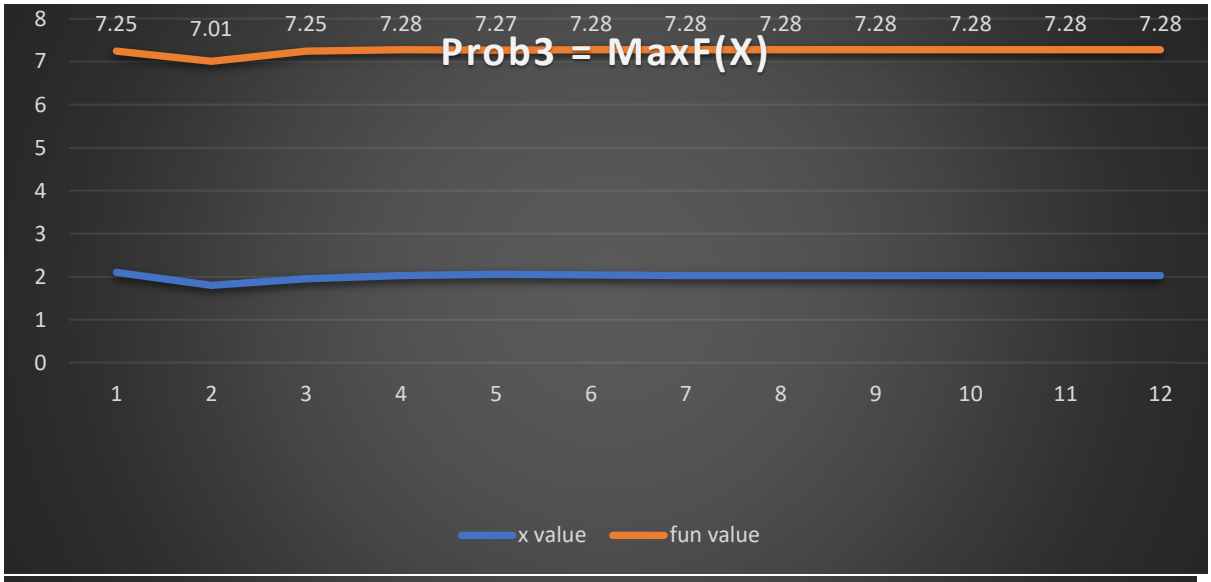
Initial Guess	△=Value of increment	No. of iteration	No. of function evaluation	Bracketed Interval	Bisection Method	
Q.1 f(x) = (x^2-1)^3-(2*x-5)^4 (Since question is given for maximization so final answer of f(x) must be multiplied with minus(likewise in other question)					**Feval no**	**Minima point**
-8	.1	5	8	(-6.500,-1.700)	52	-5.775
	.2	4	7	(-6.600,-1.800)	54	-5.775
	.3	3	6	(-7.100,-3.500)	54	-5.775
	.4	3	6	(-6.800,-2.000)	52	-5.775
	.5	2	5	(-7.500,-4.500)	50	-5.775
Q.2 f(X) = 2*exp(x) +2*x - x^3-8						
0	.1	3	6	(-1.500,-.300)	26	-0.960
	.2	2	5	(-1.400,-.200)	28	-0.960
	.3	2	5	(-2.100,-.300)	30	-0.960
	.4	2	5	(-2.800,-.400)	24	-0.960
	.5	1	4	(-1.500,.000)	30	-0.960
Q.3 f(X) = -4*x*sin(x)						
3	.1	3	6	(1.500,2.700)	28	2.029
	.2	2	5	(1.600,2.800)	28	2.029
	.3	2	5	(.900,2.700)	30	2.029
	.4	2	5	(0.200,2.600)	28	2.029
	.5	2	5	(-.500,2.500)	30	2.029
Q.4 f(X) = 2*(x-3)^2+ exp(.5*x^2)						
1	.1	3	6	(1.300,2.500)	18	1.591
	.2	2	5	(1.200,2.400)	30	1.591
	.3	1	4	(1.000,1.900)	14	1.591
	.4	1	4	(1.000,2.200)	18	1.591
	.5	1	4	(1.000,2.500)	32	1.591
Q.5 f(X) = x^2-10*exp(.1*x)						
5	.1	5	8	(-1.300,3.500)	28	0.527
	.2	4	7	(-1.200,3.600)	26	0.527
	.3	4	7	(-2.900,4.300)	30	0.527
	.4	4	7	(-2.200,7.400)	30	0.527
	.5	3	6	(-2.500,3.500)	30	0.527
Q.6 f(X) = 15*x^2-20*sin(x)						
-3	.1	5	8	(-1.500,3.300)	36	0.564
	.2	4	7	(-1.600,3.200)	38	0.564
	.3	4	7	(0.900,6.300)	36	0.564
	.4	3	6	(-1.800,3.000)	36	0.564
	.5	3	6	(-1.500,4.500)	38	0.564

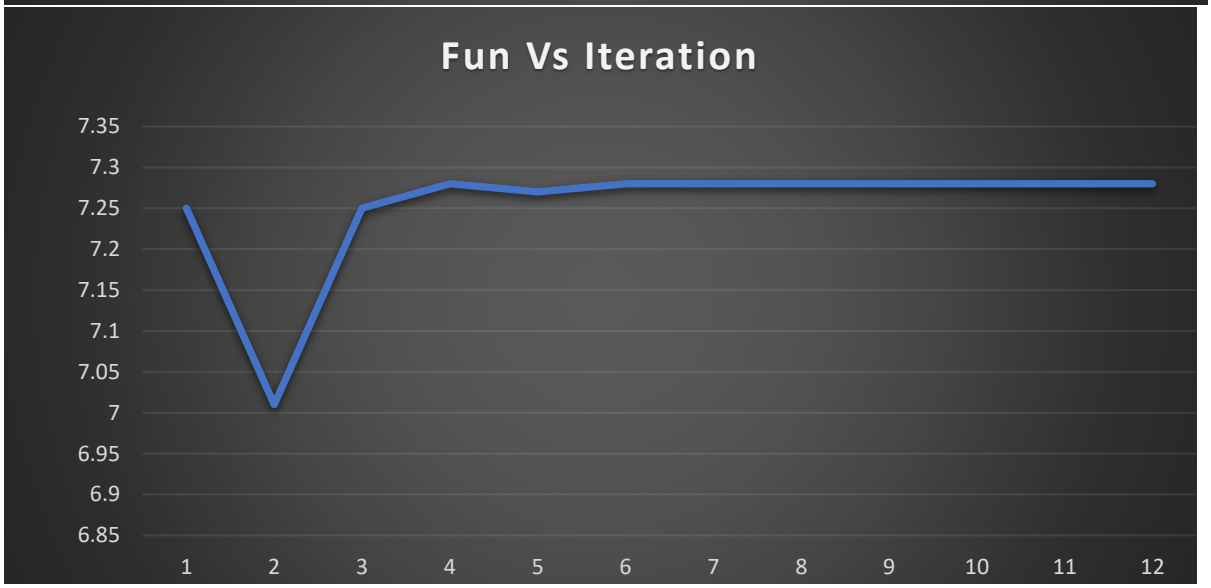

prob1 = MaxF(X)
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
-5000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
x value
fun value

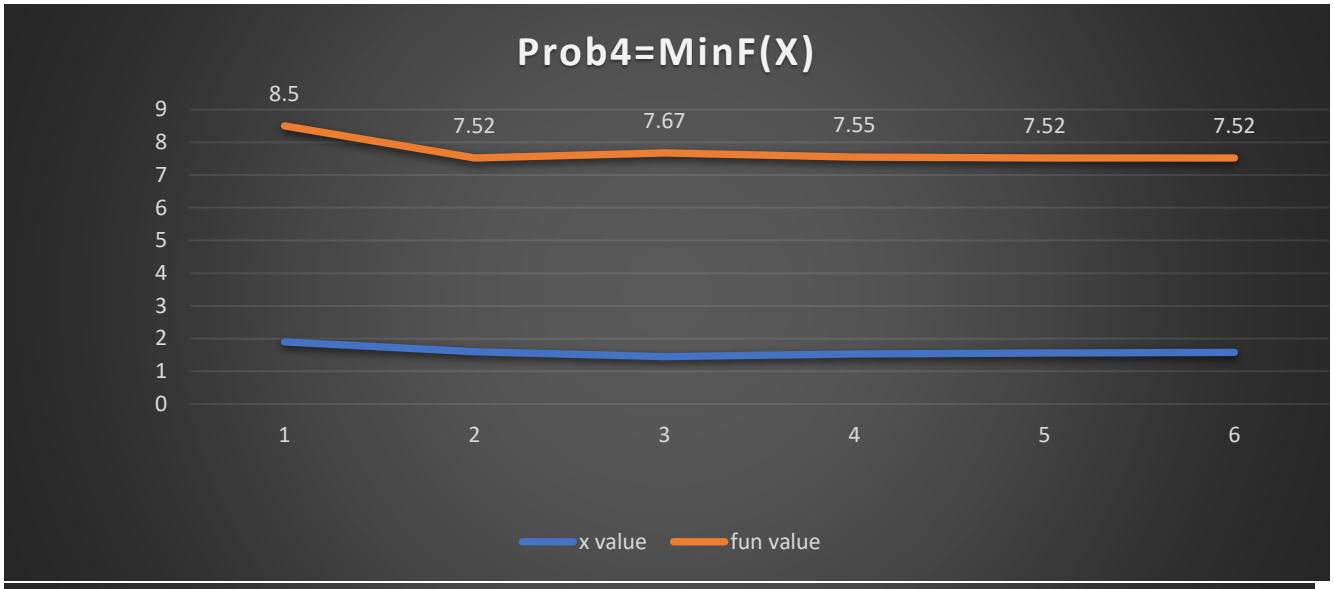

Fun v/s iteration
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

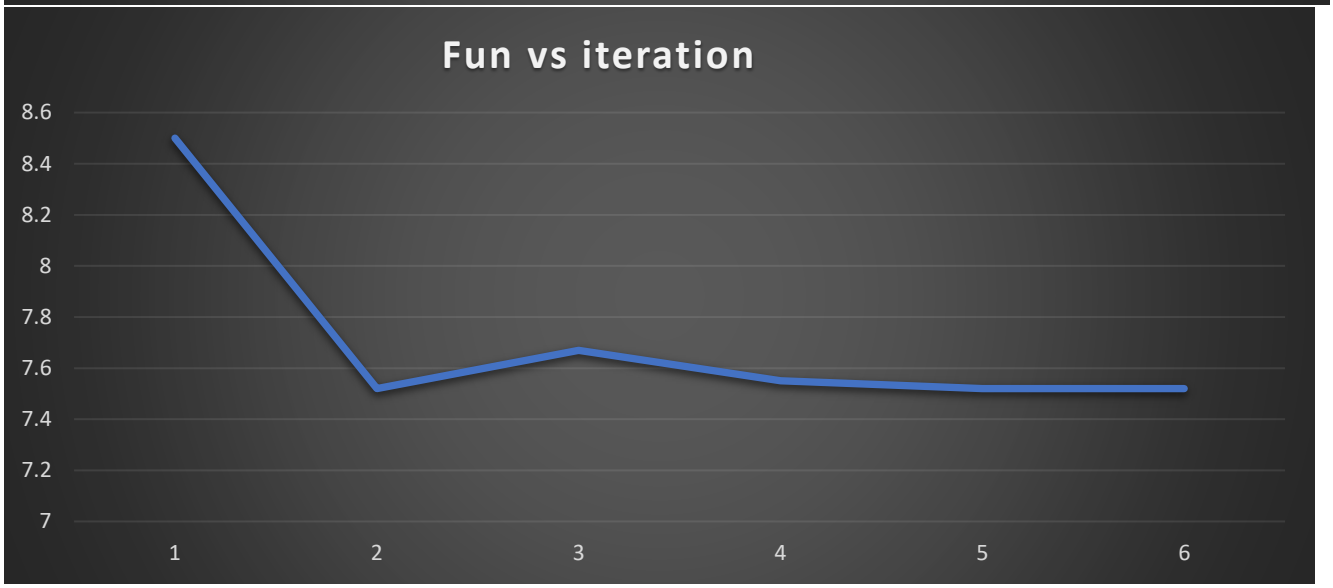

Prob2 = MaxF(X)
10
8
6
4
2
0
-2
8.26
8.07
8.24
8.27
8.27
8.27
8.27
8.27
8.27
8.27
-0.9
-1.2
-1.05
-0.97
-0.94
-0.96
-0.97
-0.96
-0.96
-0.96
1
2
3
4
5
6
7
8
9
10
x value
fun value

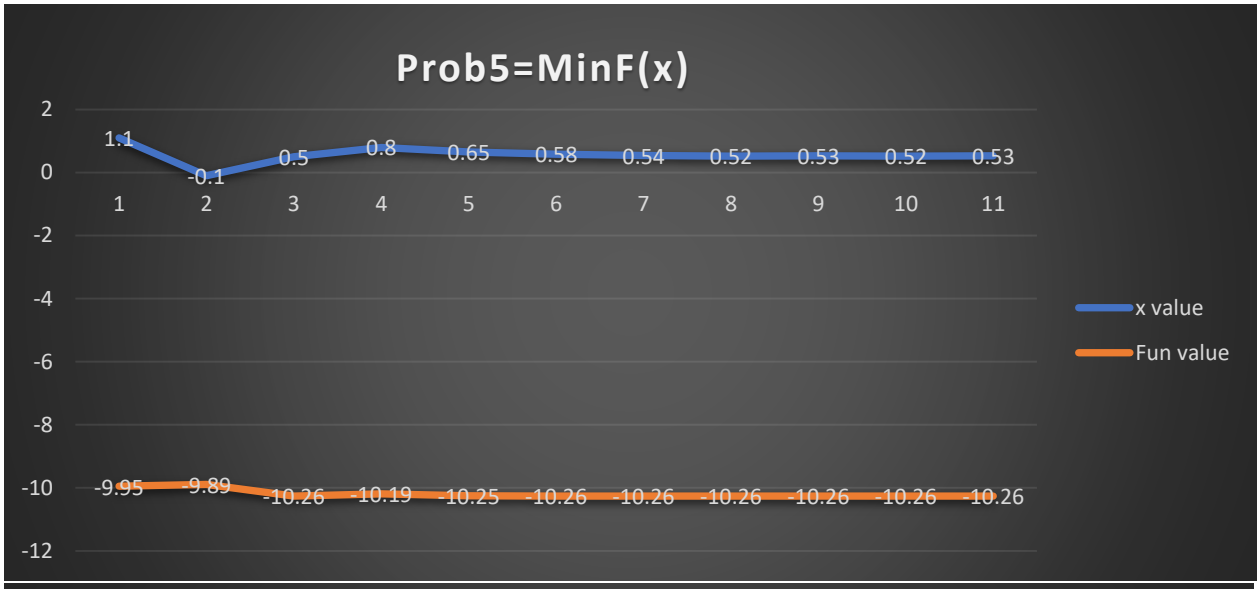

Fun vs Iteration
8.3
8.25
8.2
8.15
8.1
8.05
8
7.95
1
2
3
4
5
6
7
8
9
10

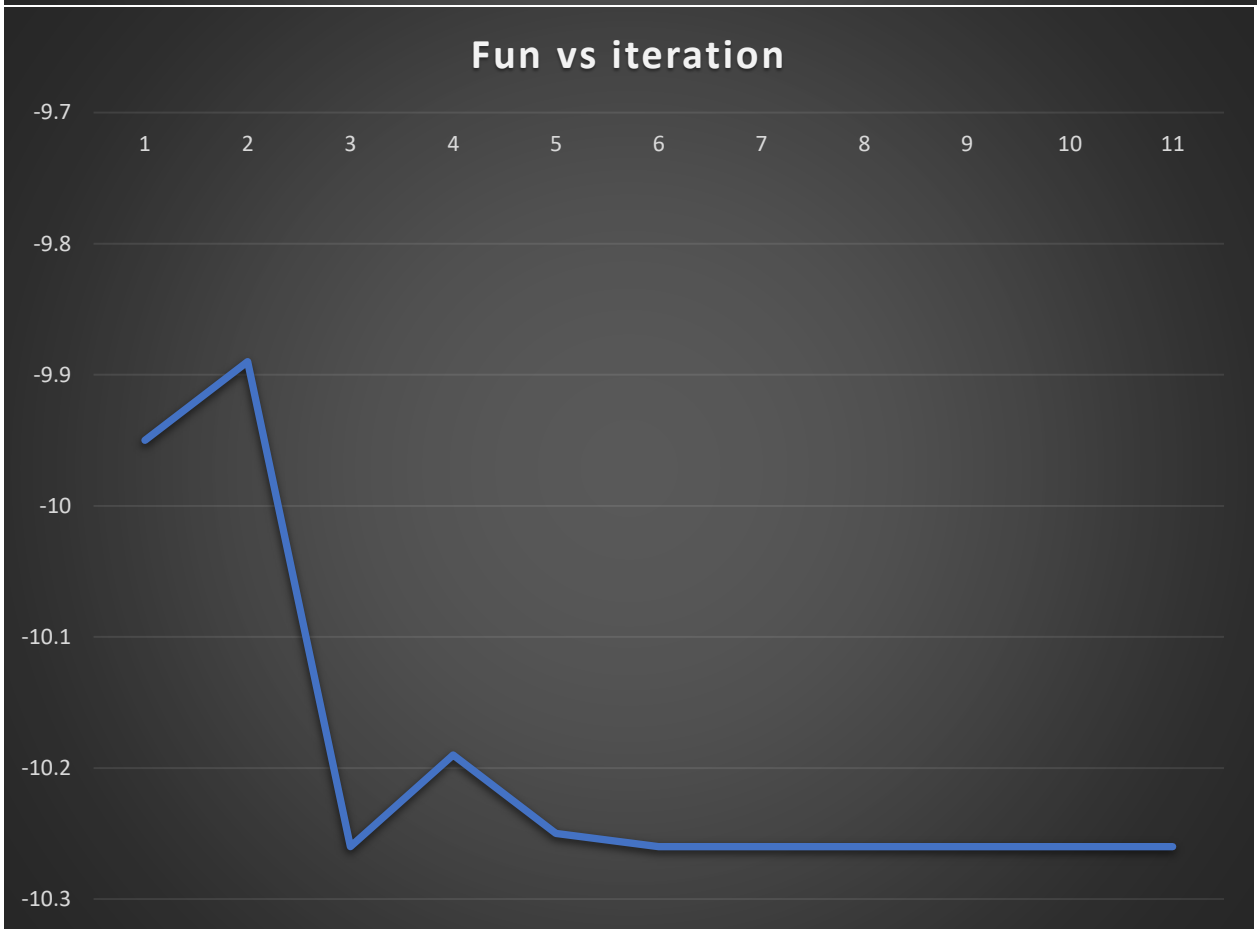

Prob3 = MaxF(X)
7.25
7.01
7.25
7.28
7.27
7.28
7.28
7.28
7.28
7.28
7.28
7.28
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
12
x value
fun value

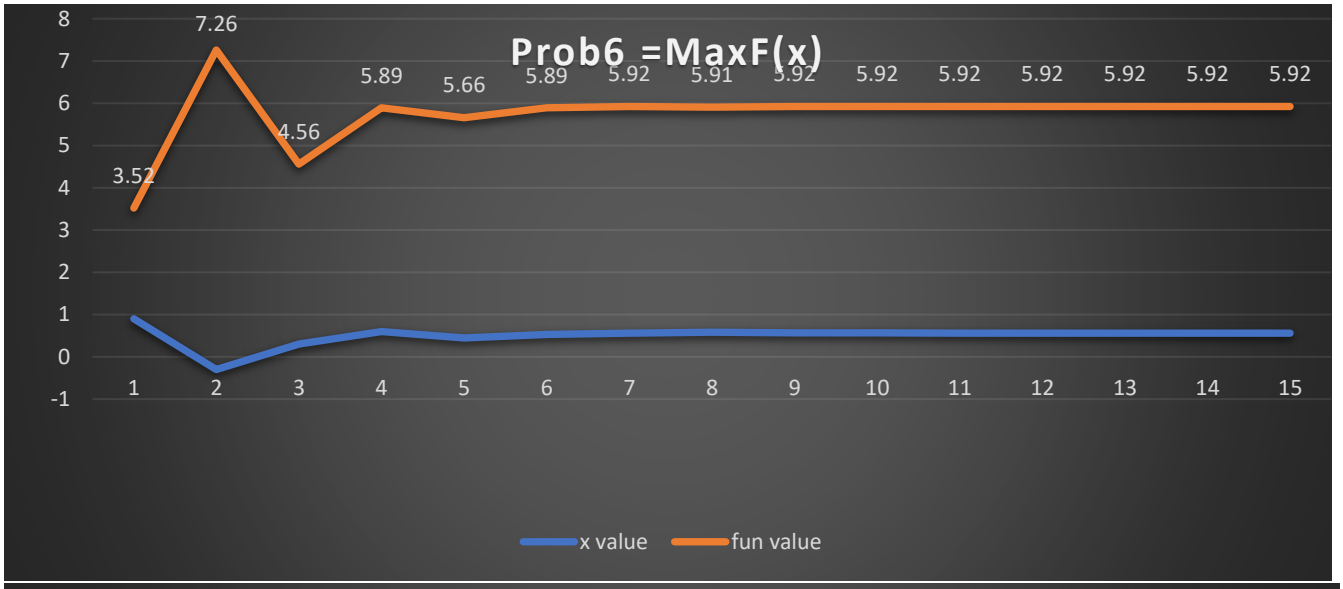

Fun Vs Iteration
7.35
7.3
7.25
7.2
7.15
7.1
7.05
7
6.95
6.9
6.85
1
2
3
4
5
6
7
8
9
10
11
12

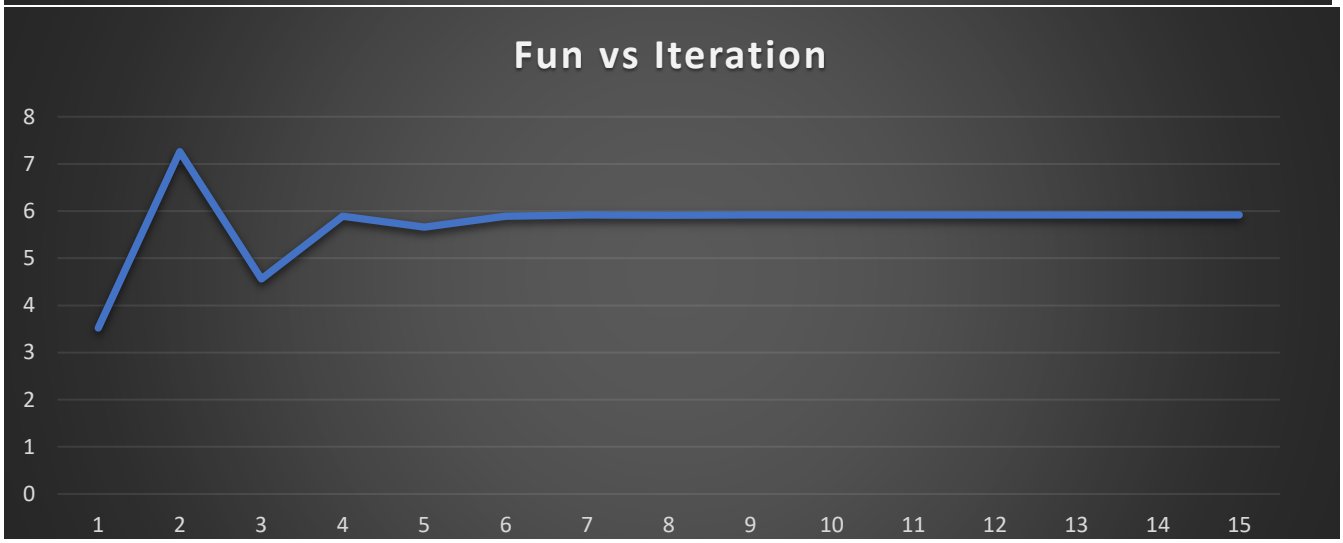

Prob4=MinF(X)
8.5
7.52
7.67
7.55
7.52
7.52
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
x value
fun value


Fun vs iteration
8.6
8.4
8.2
8
7.8
7.6
7.4
7.2
7
1
2
3
4
5
6


Prob5=MinF(x)
2
0
-2
-4
-6
-8
-10
-12
1
2
3
4
5
6
7
8
9
10
11
1.1
-0.1
0.5
0.8
0.65
0.58
0.54
0.52
0.53
0.52
0.53
-9.95
-9.89
-10.26
-10.19
-10.25
-10.26
-10.26
-10.26
-10.26
-10.26
-10.26
x value
Fun value


Fun vs iteration
-9.7
-9.8
-9.9
-10
-10.1
-10.2
-10.3
1
2
3
4
5
6
7
8
9
10
11


Prob6 =MaxF(x)
8
7
6
5
4
3
2
1
0
-1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
7.26
3.52
4.56
5.89
5.66
5.89
5.92
5.91
5.92
5.92
5.92
5.92
5.92
5.92
x value
fun value


Fun vs Iteration
8
7
6
5
4
3
2
1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15