

Scilab Manual for
Numerical techniques lab
by Prof Kanika Gupta
Others
ABES Engineering College, Ghaziabad¹

Solutions provided by
Prof Kanika Gupta
Others
ABES Engineering College, Ghaziabad

June 20, 2026

¹Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website <http://scilab.in>

Contents

List of Scilab Solutions	3
1 To find out the root of the Algebraic and Transcendental equations using Bisection method	5
2 To find out the root of the Algebraic and Transcendental equations using Regula-falsi method	9
3 To find out the root of the Algebraic and Transcendental equations using Newton Raphson	13
4 To find out the root of the Algebraic and Transcendental equations using Method of Iteration.	17
5 To implement Newtons Forward and Backward difference table.	19
6 To implement Langranges Interpolation formula	23
7 To implement Numerical Integration using Trapezoidal, Simpson 1/3 and Simpson 3/8 rule.	25
8 To implement Least Square Method for curve fitting	30
9 To estimate regression equation from sampled data and regression coefficient	33

List of Experiments

Solution 1.1	Bisection Method	5
Solution 2.2	REGULA FALSI	9
Solution 3.3	NewtonRaphson	13
Solution 4.4	ITERATION	17
Solution 5.5	NewtonFwdAndBckwd	19
Solution 6.6	Lagrange Interpolation	23
Solution 7.7	Trapezoidal Simpsons	25
Solution 8.8	Curve Fitting	30
Solution 9.9	REGRESSION	33

List of Figures

1.1	Bisection Method	7
1.2	Bisection Method	8
2.1	REGULA FALSI	10
2.2	REGULA FALSI	10
3.1	NewtonRaphson	14
3.2	NewtonRaphson	15
4.1	ITERATION	18
5.1	NewtonFwdAndBckwd	20
5.2	NewtonFwdAndBckwd	20
6.1	Lagrange Interpolation	24
7.1	Trapezoidal Simpsons	26
7.2	Trapezoidal Simpsons	26
8.1	Curve Fitting	32
8.2	Curve Fitting	32
9.1	REGRESSION	34

Experiment: 1

To find out the root of the Algebraic and Transcendental equations using Bisection method

Scilab code Solution 1.1 Bisection Method

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.1
4 //Objective–Root of algebraic and transcendental
   equations using bisection
5 //for example for solution of given equation
6 //ax4+bx3+cx2+dx+e=0
7 //For Example Enter the value of a,b,c,d and e for
   the polynomial as
8 //a=1
9 //b=0
10 //c=0
11 //d=-1
12 //e=-10
13 //Enter the range p and q for checking the function
```

```

14 //p= 1.8
15 //q= 1.9
16 //such that f(p) is positive and f(q) is negative
17 //then x=(a+b)/2
18 //the graph will be plotted
19 clc
20 clear
21 disp("Enter value of a,b,c,d,e ax^4+bx^3+cx^2+dx+e=0
      ")
22 a=input("Enter a: ");
23 b=input("Enter b: ");
24 c=input("Enter c: ");
25 d=input("Enter d: ");
26 e=input("Enter e: ");
27 function y=fun(x)
28     y=a*x.^4+b*x.^3+c*x.^2+d*x+e;
29 endfunction
30 //deff["fun(int x)=(a*x^4)+(b*x^3)+(c*x^2)+(d*x)+e",
      x];
31 p=input("Enter 1st initial approximation: ");
32 q=input("Enter 2nd initial approximation: ");
33 while(fun(p)*fun(q)>0)
34     disp("Enter correct range of approximations");
35     p=input("Enter 1st initial approximation: ");
36     q=input("Enter 2nd initial approximation: ");
37 end
38 p1=[p:.1:q]
39 x=(p+q)/2;
40 i=0;
41 z=0;
42 while(abs(fun(x))>0.0000001)
43     x=(p+q)/2;
44 if(fun(x)*fun(q)<0) then
45     p=x;
46 else
47     q=x;
48 end
49 i=i+1;

```

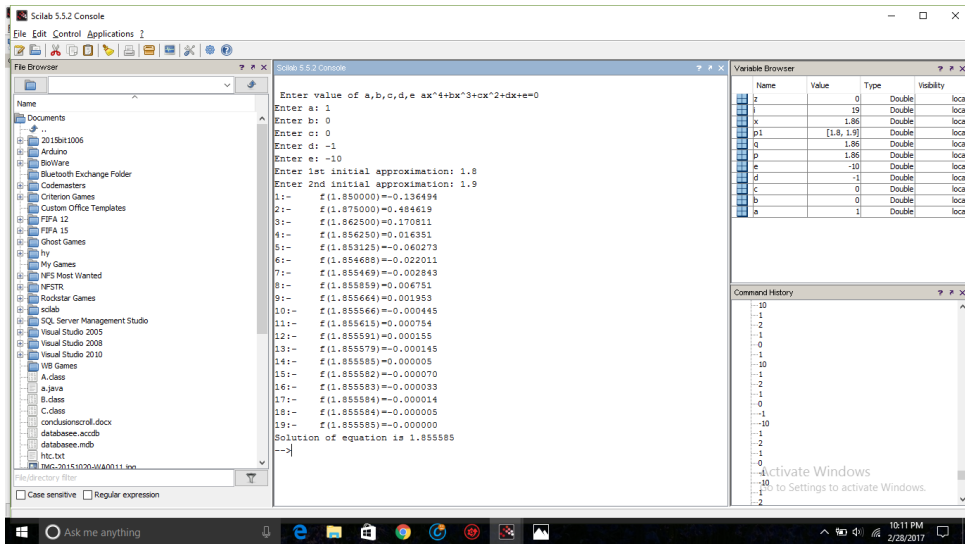


Figure 1.1: Bisection Method

```

50 mprintf( '%d:-\ t f (%f)=%f\n' , i , x , fun ( x ) )
51 end
52 mprintf(" Solution of equation is %f" , x)
53 plot ( p1 , fun ( p1 ) )

```

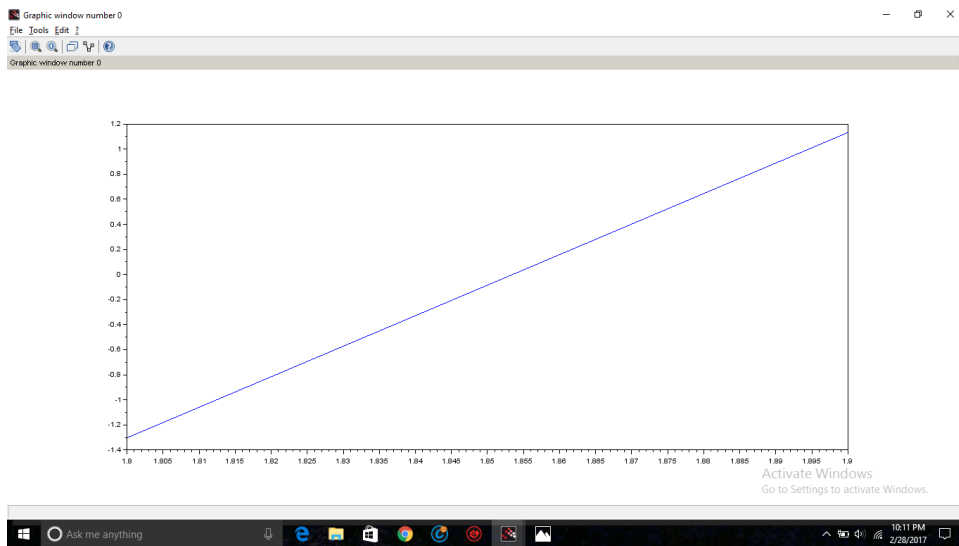


Figure 1.2: Bisection Method

Experiment: 2

To find out the root of the Algebraic and Transcendental equations using Regula-falsi method

Scilab code Solution 2.2 REGULA FALSI

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.2
4 //Objective–Root of algebraic and transcendental
   equations using Regula–falsi
5 //for example for solution of given equation
6 //ax4+bx3+cx2+dx+e=0
7 //For Example Enter the value of a,b,c,d and e for
   the polynomial as
8 //a=1
9 //b=0
```

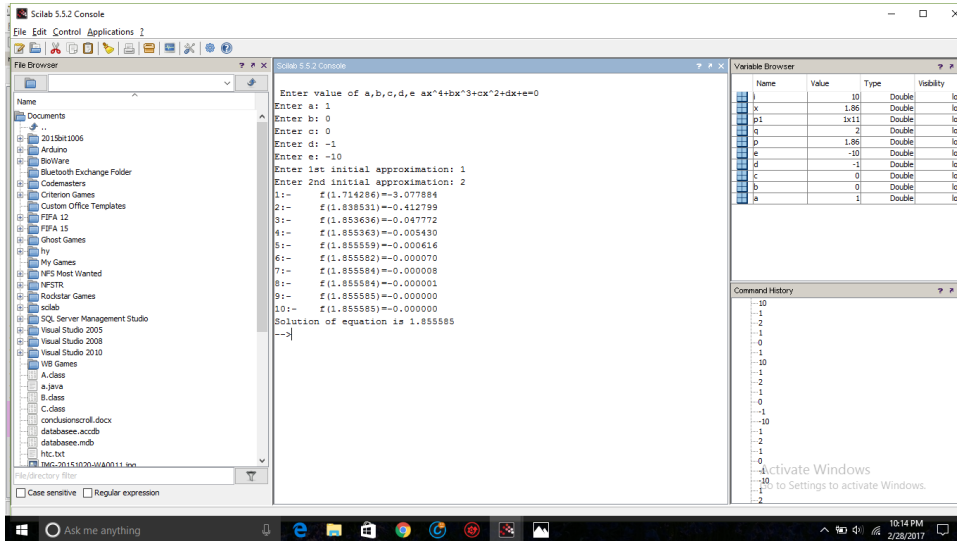


Figure 2.1: REGULA FALSI

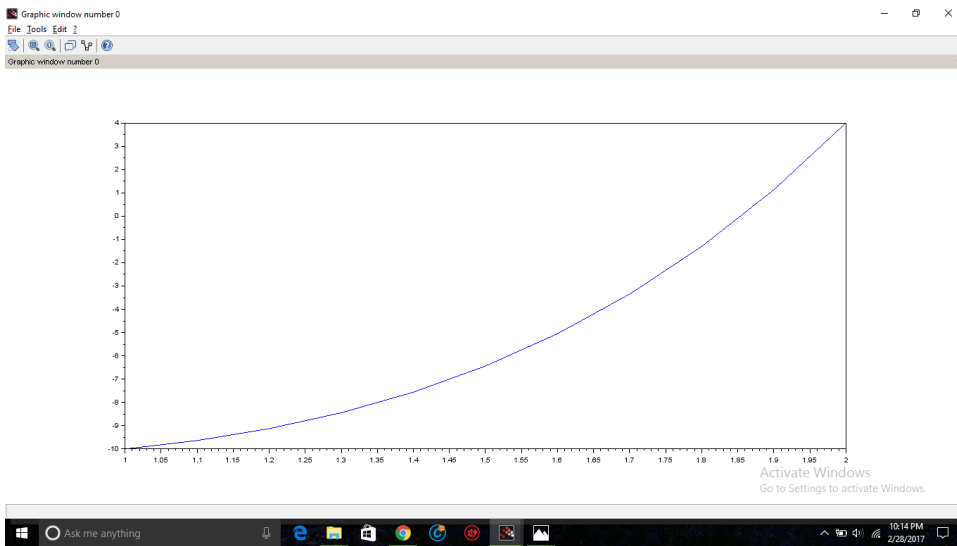


Figure 2.2: REGULA FALSI

```

10 //c=0
11 //d=-1
12 //e=-10
13 //Enter the range p and q for checking the function
14 //p=1
15 //q=2
16 //such that f(p) is positive and f(q) is negative
17 //then x=(a+b)/2
18 //the graph will be plotted
19 clc
20 clear
21 disp("Enter value of a,b,c,d,e ax^4+bx^3+cx^2+dx+e=0
    ")
22 a=input("Enter a: ");
23 b=input("Enter b: ");
24 c=input("Enter c: ");
25 d=input("Enter d: ");
26 e=input("Enter e: ");
27 function y=fun(x)
28     y=a*x.^4+b*x.^3+c*x.^2+d*x+e;
29 endfunction
30 //deff[" fun(int x)=(a*x^4)+(b*x^3)+(c*x^2)+(d*x)+e",
    x];
31     p=input("Enter 1st initial approximation: ");
32     q=input("Enter 2nd initial approximation: ");
33 while(fun(p)*fun(q)>0)
34     disp("Enter correct range of approximations");
35     p=input("Enter 1st initial approximation: ");
36     q=input("Enter 2nd initial approximation: ");
37 end
38 p1=[p:.1:q]
39 x=(p*fun(q)-q*fun(p))/(fun(q)-fun(p));
40 i=0;
41 while(abs(fun(x))>.0000001)
42 x=(p*fun(q)-q*fun(p))/(fun(q)-fun(p));
43 if(fun(x)*fun(q)<0) then
44     p=x;
45 else

```

```
46     q=x;
47 end
48 i=i+1;
49 mprintf( '%d:-\ tf (%f)=%f\n ', i, x, fun(x))
50 end
51 mprintf('Solution of equation is %f',x)
52 plot(p1,fun(p1))
```

Experiment: 3

To find out the root of the Algebraic and Transcendental equations using Newton Raphson

Scilab code Solution 3.3 NewtonRaphson

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.3
4 //Objective–Root of algebraic and transcendental
   equations using Newton Raphson
5 //for example for solution of given equation
6 //ax4+bx3+cx2+dx+e=0
7 //Enter the value of a,b,c,d and e for the
   polynomial as
8 //a= 1
9 //b= 1
```

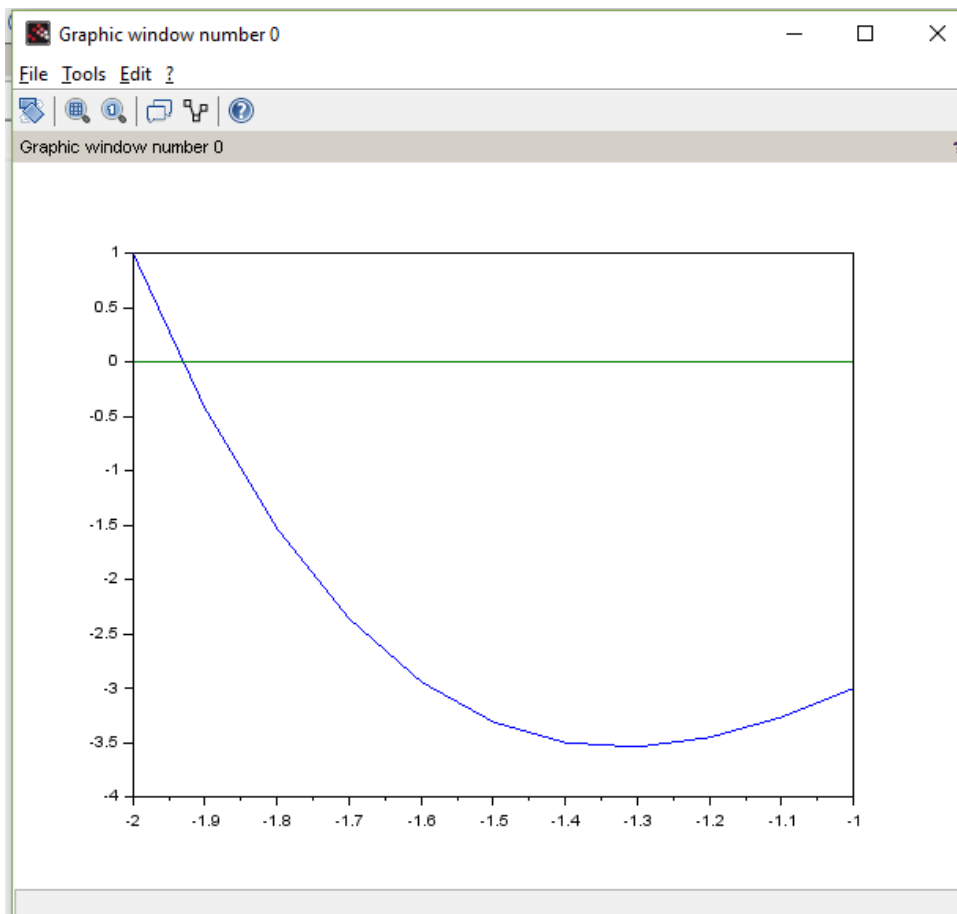



Figure 3.2: NewtonRaphson

```
    x];
29 x=input("Enter 1st initial approximation: ");
30 y=x;
31 i=0;
32 while(abs(fun(x))>.000001)
33     i=1+i;
34     mprintf(' %d:-\ tf(%f)=%f\n',i,x,fun(x))
35     x=x-(fun(x)/numderivative(fun,x));
36 end
37 if(y<x) then
38     p=[y:.1:x+1]
39 else
40     p=[x:.1:y+1]
41 end
42 mprintf("Solution of equation is %f",x)
43 plot(p,fun(p),p,p*0)
```

Experiment: 4

To find out the root of the Algebraic and Transcendental equations using Method of Iteration.

Scilab code Solution 4.4 ITERATION

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.4
4 //Objective: Root of algebraic and transcendental
   equations using iteration
5 //for example for solution of given equation
6 //x3-5x+1=0
7 //Enter the initial approximation x=1
8 //then x=g(x)
9 clc
10 clear
11 function y=f(x)
12     y=x3-5*x+1
```

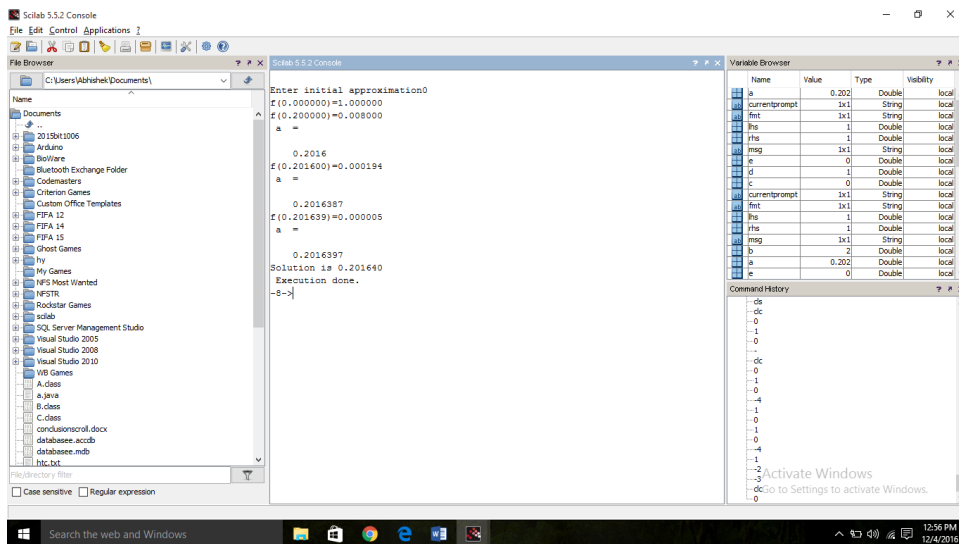


Figure 4.1: ITERATION

```

13 endfunction
14 function y=fi(x)
15     y=(x^3+1)/5
16 endfunction
17
18 a=input("Enter initial approximation ");
19 mprintf(" f (%f)=%f\n",a,f(a))
20 a=fi(a);
21 while(abs(f(a))>.000001)
22     mprintf(" f (%f)=%f\n",a,f(a))
23     a=fi(a)
24 end
25 mprintf(" Solution is %f",a)

```

Experiment: 5

To implement Newtons
Forward and Backward
difference table.

Scilab code Solution 5.5 NewtonFwdAndBckwd

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.5
4 //objective: newton's forward difference
   interpolation table and newton's backward
   interpolation table
5 //for example for forward difference select 1
6 //enter number of values 5
7 //matrix is formed of 5*5
8 //enter the value of x and y simultaneously
9 //Enter the element x: 1
10 //Enter the element y: 6
11 //Enter the element x: 2
```

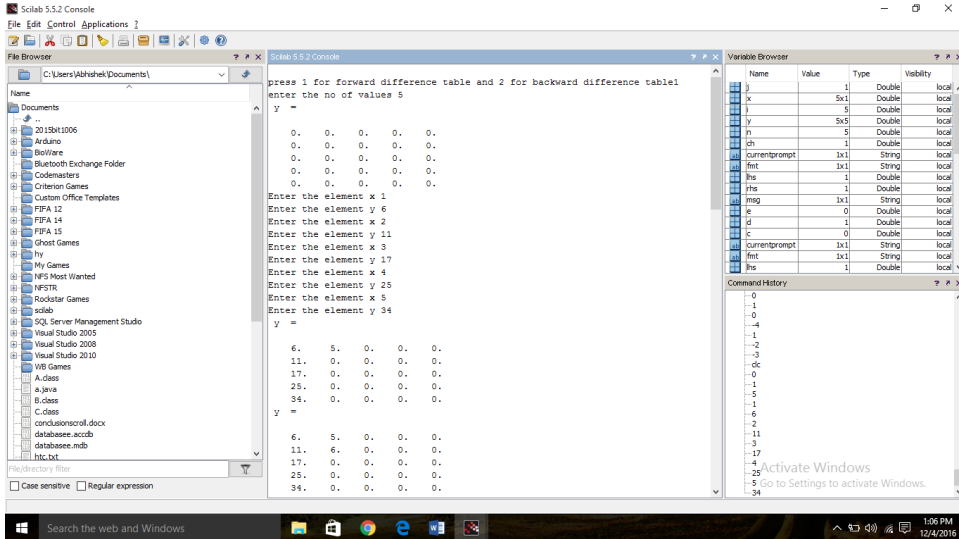


Figure 5.1: NewtonFwdAndBckwd

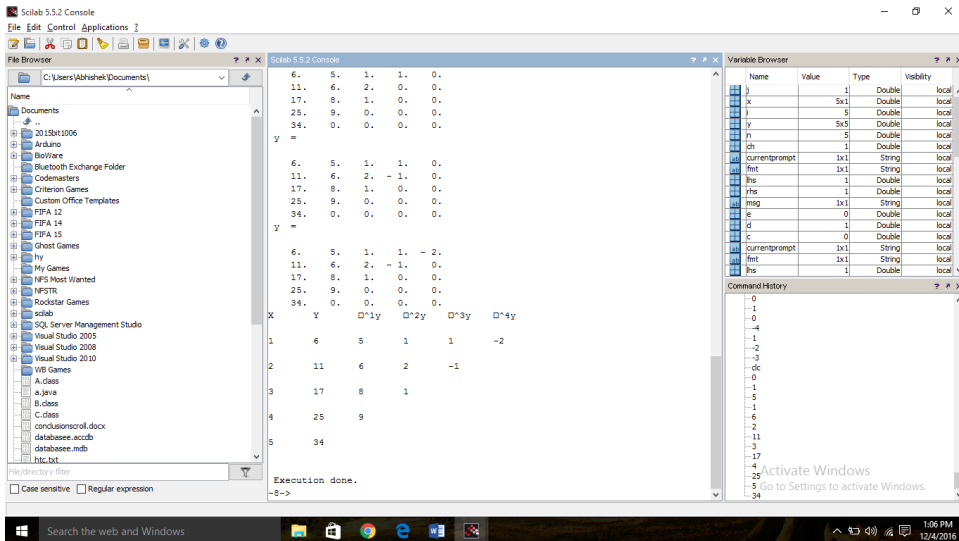


Figure 5.2: NewtonFwdAndBckwd

```

12 //Enter the element y: 11
13 //Enter the element x: 3
14 //Enter the element y: 17
15 //Enter the element x: 4
16 //Enter the element y: 25
17 //Enter the element x: 5
18 //Enter the element y: 34
19 clc
20 clear
21 ch=input("press 1 for forward interpolation table
           and 2 for backward interpolation table");
22 if(ch==1) then
23 n=input("enter the no of values ");
24 y=zeros(n,n)
25 for i=1:n
26     x(i)=input("Enter the element x ");
27     y(i,1)=input("Enter the element y ");
28 end
29 for i=2:n
30     for j=1:n-i+1
31         y(j,i)=y(j+1,i-1)-y(j,i-1)
32     end
33 end
34 mprintf("X\tY\t")
35 for i=1:n-1
36     mprintf("%c^%dy\t",char(30),i)
37 end
38 disp("")
39 for i=1:n
40     mprintf("%d\t",x(i))
41     for j=1:n-i+1
42         mprintf("%d\t",y(i,j))
43     end
44     disp("")
45 end
46 else
47     n=input("enter the no of values ");
48 y=zeros(n,n)

```

```
49 for i=1:n
50     x(i)=input("Enter the element x ");
51     y(i,1)=input("Enter the element y ");
52 end
53 for i=2:n
54     for j=1:n-i+1
55         y(j,i)=y(j+1,i-1)-y(j,i-1)
56     end
57 end
58 mprintf("X\tY\t")
59 for i=1:n-1
60     mprintf("%c^%dy\t",char(31),i)
61 end
62 disp("")
63 for i=1:n
64     mprintf("%d\t",x(i))
65     for j=1:n-i+1
66         mprintf("%d\t",y(i,j))
67     end
68     disp("")
69 end
70 end
```

Experiment: 6

To implement Lagrange's Interpolation formula

Scilab code Solution 6.6 Lagrange Interpolation

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.6
4 //Objective:Lagrange's Interpolation formula
5 //for example Enter the number of terms in table as
6 //n=2
7 //enter the values of x and y as
8 //x= 1
9 //y= 1
10 //x= 6
11 //y= 36
12 //Enter the value of x to calculate y as
13 //a=5
14 clc
15 clear
16 n=input("Enter the no of terms in the table :-")
17 for i=1:n
```

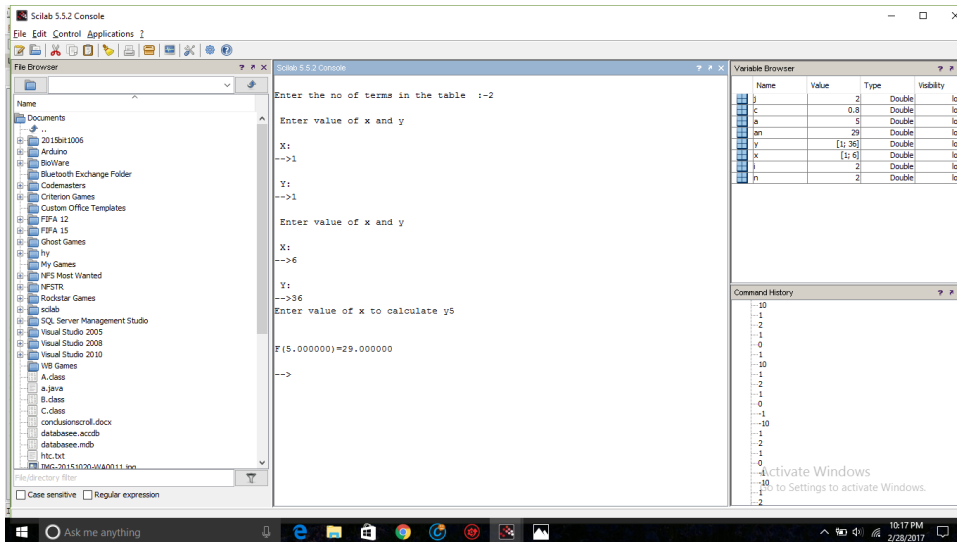


Figure 6.1: Lagrange Interpolation

```

18     disp("Enter value of x and y")
19     disp("X: ")
20     x(i)=mscanf("%f")
21     disp("Y: ")
22     y(i)=mscanf("%f")
23 end
24 an=0;
25 a=input("Enter value of x to calculate y")
26 for i=1:n
27     c=1;
28     for j=1:n
29         if(i~=j)
30             c=c*(a-x(j))/(x(i)-x(j))
31         end
32     end
33     an=an+c*y(i)
34 end
35 mprintf("\n\nF(%f)=%f\n",a,an)

```

Experiment: 7

To implement Numerical
Integration using Trapezoidal,
Simpson 1/3 and Simpson 3/8
rule.

Scilab code Solution 7.7 Trapezoidal Simpsons

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.7
4 //Objective:to impliment numerical integration using
   trapezoidal ,simpson 1/3 and simpson 3/8 rule
5 //For Example for trapezoidal press 1
6 //Enter the Start limit as
7 //a= -1
8 //Enter the End limit as
9 //b=2
10 //Enter the number of Intervals as
```

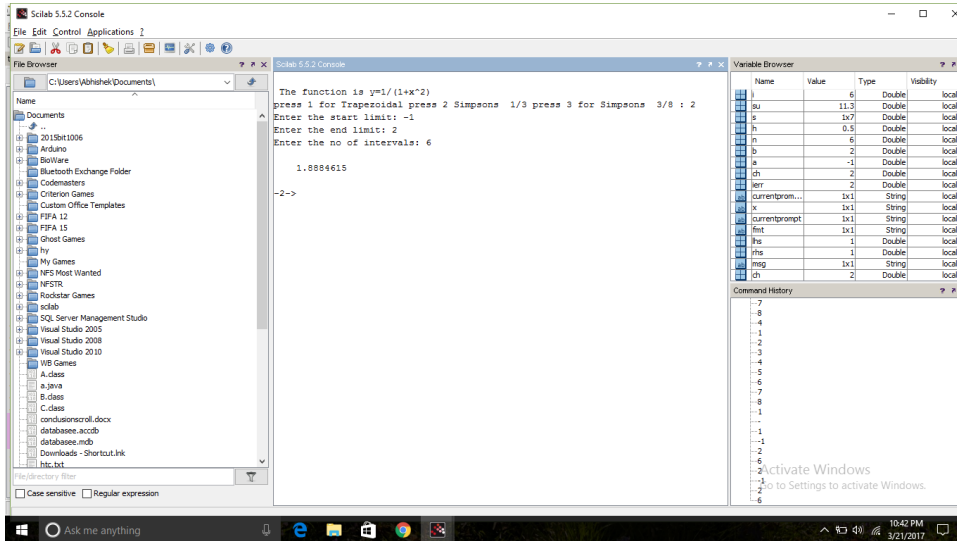


Figure 7.1: Trapezoidal Simpsons

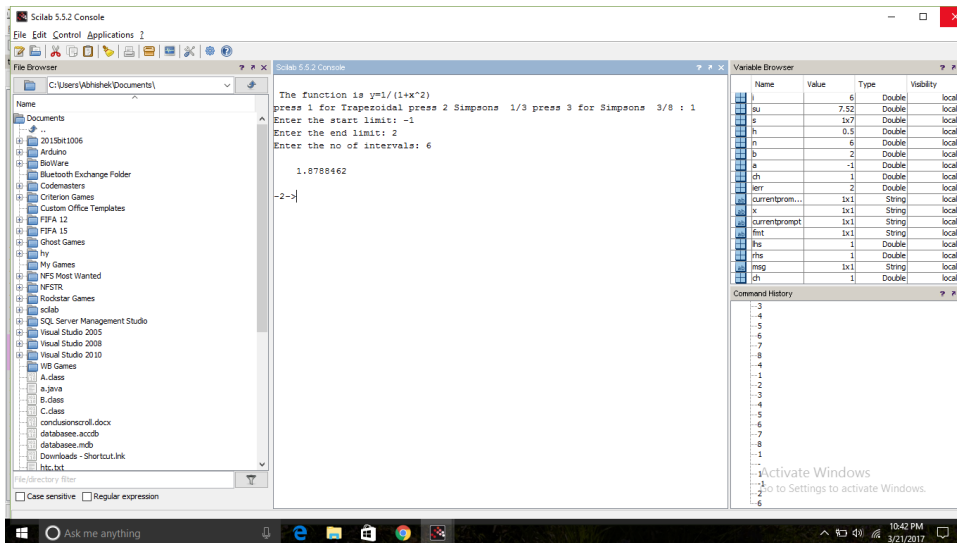


Figure 7.2: Trapezoidal Simpsons

```

11 //n=6
12 //For Example for simpsons 1/3 rule press 2
13 ///Enter the Start limit as
14 //a= -1
15 //Enter the End limit as
16 //b=2
17 //Enter the number of Intervals as
18 //n=6
19 //For Example for simpsons 3/8 rule press 3
20 ///Enter the Start limit as
21 //a= -1
22 //Enter the End limit as
23 //b=2
24 //Enter the number of Intervals as
25 //n=6
26 clc
27 clear
28 function y=f(x)
29     y=1/(1+x^2)
30 endfunction
31 disp("The function is y=1/(1+x^2)")
32 ch=input("press 1 for Trapezoidal press 2 Simpsons
33     1/3 press 3 for Simpsons 3/8 : ")
34 if(ch==1) then
35     a=input("Enter the start limit: ")
36     b=input("Enter the end limit: ")
37     n=input("Enter the no of intervals: ")
38     h=(b-a)/n
39     s=[a:h:b]
40     su=0
41     for i=2:length(s)-1
42         su=su+2*f(s(i))
43     end
44     su=su+f(s(1))+f(s(length(s)))
45     ans=h*su/2
46     disp(ans)
47 //Simpson's 1/3
48 elseif(ch==2) then

```

```

48 function y=f(x)
49     y=1/(1+x^2)
50 endfunction
51 a=input("Enter the start limit: ")
52 b=input("Enter the end limit: ")
53 n=input("Enter the no of intervals: ")
54 h=(b-a)/n
55 s=[a:h:b]
56 su=0
57 for i=2:length(s)-1
58     if(modulo(i-1,2)==0)
59         su=su+2*f(s(i))
60     else
61         su=su+4*f(s(i))
62     end
63 end
64 su=su+f(s(1))+f(s(length(s)))
65 ans=h*su/3
66 disp(ans)
67 //Simpson's 3/8
68 elseif(ch==3) then
69 function y=f(x)
70     y=1/(1+x^2)
71 endfunction
72 a=input("Enter the start limit: ")
73 b=input("Enter the end limit: ")
74 n=input("Enter the no of intervals: ")
75 h=(b-a)/n
76 s=[a:h:b]
77 su=0
78 for i=2:length(s)-1
79     if(modulo(i-1,3)==0)
80         su=su+2*f(s(i))
81     else
82         su=su+3*f(s(i))
83     end
84 end
85 su=su+f(s(1))+f(s(length(s)))

```

```
86 ans=3*h*su/8
87 disp(ans)
88 end
```

Experiment: 8

To implement Least Square Method for curve fitting

Scilab code Solution 8.8 Curve Fitting

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.8
4 //fitting a curve by least square method
5 //For example for fitting a curve enter the number
  of terms as
6 //n=4
7 //Enter x : 1
8 //Enter y : 2
9 //Enter x : 3
10 //Enter y : 4
11 //Enter x : 5
12 //Enter y : 5
13 //Enter x : 9
14 //Enter y : 10
15
16 clc
17 clear
18 n=input("Enter no of terms : ")
```

```

19 for i=1:n
20     x(i)=input("Enter x : ")
21     y(i)=input("Enter y : ")
22 end
23 x0=0
24 y0=0
25 x2=0
26 xy=0
27 for i=1:n
28     x0=x0+x(i)
29     y0=y0+log10(y(i))
30     x2=x2+x(i)^2
31     xy=xy+x(i)*log10(y(i))
32 end
33 p=[x0:.1:x2]
34 q=[y0:.1:xy]
35 b=((xy/x0)-(y0/n))/((x2/x0)-(x0/n))
36 a=(xy-b*x2)/x0
37 A=10^a
38 B=b/log10(exp(1))
39 p=[x(1):.1:x(n)]
40 y=A*exp(B*p)
41 mprintf("A=%f\nB=%f\nIs the solution of the equation
         y=A*e^Bx",A,B)
42 plot(p,y)

```

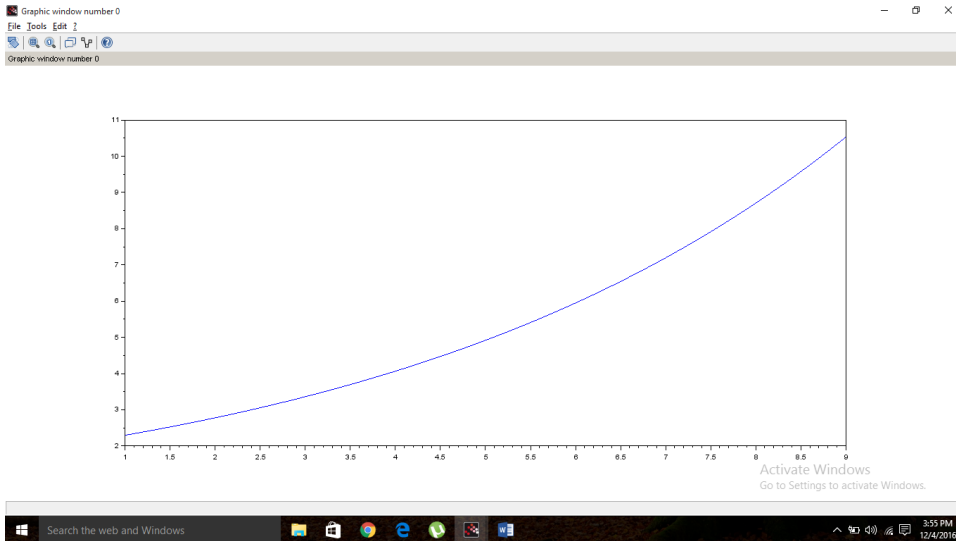


Figure 8.1: Curve Fitting

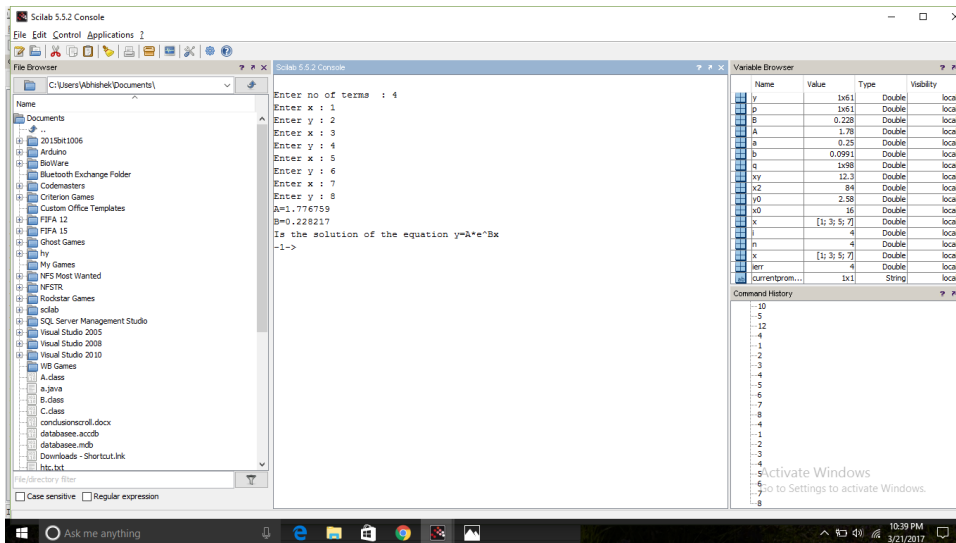


Figure 8.2: Curve Fitting

Experiment: 9

To estimate regression equation
from sampled data and
regression coefficient

Scilab code Solution 9.9 REGRESSION

```
1 //Operating System – Windows 10
2 //SCILAB version 5.5.2
3 //Experiment No.9
4 //to estimate regression equation from sampled data
   and regression coefficient
5 //Enter number of terms as
6 //n=4
7 //Enter x : 2
8 //Enter y : 3
9 //Enter x : 3
10 //Enter y : 7
11 //Enter x : 4
12 //Enter y : 10
13 //Enter x : 5
14 //Enter y : 12
```

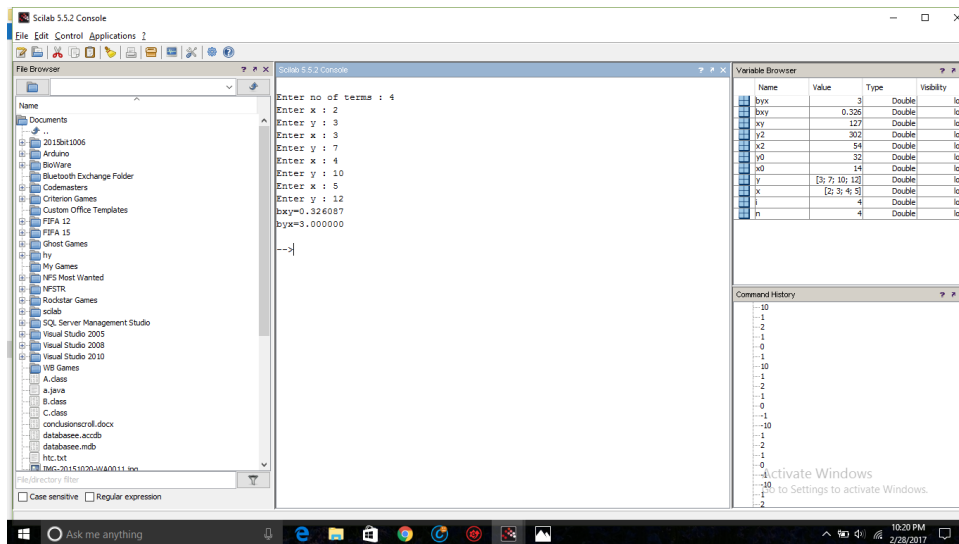


Figure 9.1: REGRESSION

```

15
16 clc
17 clear
18 n=input("Enter no of terms : ")
19 for i=1:n
20     x(i)=input("Enter x : ")
21     y(i)=input("Enter y : ")
22 end
23 x0=0
24 y0=0
25 x2=0
26 y2=0
27 xy=0
28 for i=1:n
29     x0=x0+x(i)
30     y0=y0+y(i)
31     x2=x2+x(i)^2
32     y2=y2+y(i)^2
33     xy=xy+x(i)*y(i)
34 end

```

```
35 bxy=(n*xy-x0*y0)/(n*y2-y0^2)
36 byx=(n*xy-x0*y0)/(n*x2-x0^2)
37 mprintf(" bxy=%f\nbyx=%f\n", bxy, byx)
```
