LAB MANUAL

APPLIED NUMERICAL TECHNIQUES AND COMP.

ME-319-E

LIST OF EXPERIMENTS

APPLIED NUMERICAL TECHNIQUES AND COMP. M E-319-E

S. No.	NAME OF EXPERIMENTS
1.	Solution of Non-linear equation in single variable using the method of successive bisection.
2.	Solution of Non-linear equation in single variable using the Regula-Falsi & Newton Raphson method.
3.	Solution of a system of simultaneous algebraic equations using the Gaussian elimination procedure.
4.	Solution of a system of simultaneous algebraic equations using the Gauss-Seidel iterative method.
5.	Numerical solution of an ordinary differential equation using the Euler's method.
6.	Numerical solution of an ordinary differential equation using the Runge-Kutta 4^{th} order method.
7.	Numerical solution of an ordinary differential equation using the predictor –corrector method.
8.	Numerical solution of a system of two ordinary differential equation using numerical integration.
9.	Numerical solution of an elliptic boundary value problem using the method of finite differences.

EXPERIMENT No. -1

AIM:-
Solution of Non- Linear Equation in single variable using the method of successive bisection.

ALGORITHM:-
Bisection method.

1. Decide initial values of a & b and stopping criterion, E.
2. Compute f_1 =f(a) & f_2 =f(b)
3. if $f_1 * f_2 > 0$, a & b do not have any root and go to step 7; otherwise continue.
4. Compute *x=(a+b) /2 and compute f_0 =f(*x)
5. If $f_1 * f_0 < 0$ then
   Set b=*x
   Else
   Set a= *x
   Set $f_1 = f_0$
6. If absolute value of (b-a)/ b is less than error E, then
   root=(a+b)/2
   write the value of root
   go to step 7
   else
   go to step 4
7. stop

PROGRAM
/* Bisection method. */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x)
{
return(x*x*x-4*x-9);
}
void bisect(float *x,float a,float b ,int *itr)
{
*x=(a+b)/2;
++(*itr);
printf("iteration no. %3d x = %7.5f\n",*itr,*x);
}
main()
{
int itr=0,maxitr;
float x,a,b,aerr,x1;
clrscr();
printf("enter the value of a,b,aloowed error,maximum iteration \n");
```

```
scanf("%f%f%f%d",&a,&b,&aerr,&maxitr);
bisect(&x,a,b,&itr);
do
{
if(f(a)*f(x)<0)
b=x;
else a=x;
bisect(&x1,a,b,&itr);
if (fabs(x1-x)<aerr)
{
printf("after %d iteration ,root =%6.4f\n",itr,x1);
getch();
return 0;
}
x=x1;
}
While (itr<maxtir);
Printf("solution does not coverage," "iteration not sufficient");
return1;
}
```

Result:-

```
Enter the value of a, b, allowed error, maximum iterations
.........................................
Iteration No.  1 x=
Iteration No.  2 x=
-----------------------
-----------------------
After iteration, root=
```

EXPERIMENT NO.– 2

Aim: - Solution of Non – linear equation in single variable using the Regula – Falsi, Newton – Raphson method.

ALOGRITHM:: -

Newton – Raphson Method: -

1. Assign an initial value to x, say x_0
2. Evaluate $f(x_0)$ & (x_0)
3. Find the improved estimation of x_0

   $X_1 = x_0 – f(x_1)/f\ (x_0)$
4. Check the accuracy of the latest estimate.
   Compare relative error to a predefined value E. if $[(x_1 – x_0)/x_1]<=E$
   Stop; otherwise continue.
5. Replace x_0 by x_1 and repeat step 3 and 4.

PROGRAM

```
/* Regula falsi method*/

#include< stdio.h >
#include< conio.h >
#include<math.h>
float f(float x)
{
return cos (x) – x*exp(x);
}
void regula (float *x, float x0, float x1, float fx0, float fx1, int*itr)
{
*x= x0-( (x1-x0)/(fx1-fx0) ) *fx0;
++(*itr);
printf("iteration no. %3d x=%7.5f\n",*itr,*x);
}

main()
{
int itr=0,maxitr;
float x0,x1,x2, aerr, x3;
printf("enter the values for x0,x1,allowed error ,maxinum iterations\n");
scanf("%f %f %f %d ",&x0,&x1,&aerr,&maxitr);
regula(&x2,x0,x1,f(x0),f(x1),&itr);
do
{
if (f(x0)*f(x2)< 0)
```

```
x1=x2;
else
x0=x2;
regula(&x3,x0,x1,f(x0),f(x1),&itr);
if(fabs(x3-x2) < aerr)
{
printf("after %d iterations,root = %6.4f\n",itr,x3);
getch();
return 0;
}
x2=x3;

}
while(itr<maxitr);
printf("solution doesnt converge,iterations not sufficient");
return 1;
}
```

Result: -

Enter the value for x0, x1, allowed error, maximum iterations.

....

Iteration No.1 x=

Iteration No.2 x=

....

....

After iterations, root=

PROGRAM

```
/* Newton Raphson Method*/

#include< stdio.h >
#include< conio.h >
#include<math.h>
float f(float x)
{
return x*log10(x)-1.2;
}
float df(float x)
{
return log10(x)+0.43429;
}
main()
{
int itr,maxitr;
float h,x0,x1,aerr;
clrscr();
```

```
printf("enter the value of  x0", "allowed errorm maximum iteration \n");
scanf("%f%f%d",&x0,&aerr,&maxitr);
for(itr=1;itr<=maxitr;itr++)
{
h=f(x0)/df(x0);
x1=x0-h;
printf("iteration %3d,x=%9.6f\n",itr,x1);
if(fabs(h)<aerr)
{
printf("after %3d iteration,root=%8.6f\n",itr,x1);
return 0;
}
x0=x1;
}
printf("solution does not converge," "iteration not sufficient ");
return 1;
}
```

Result: -

Enter the x0, allowed error, maximum iterations.

....

Iteration No.1 x=

Iteration No.2 x=

....

....

After iterations, root=

EXPERIMENT NO.-3

Aim: - Solution of a system of simultaneous algebraic equation using Gaussian elimination procedure.

ALOGRITHM: -

1. Arrange equation such that a11 !=0
2. Eliminate x1 from all but the first equation. This is done as follows:
   Normalilse the first equation by dividing it by a11.
   (ii) Subtract from the second equation a21 times the nomalised first equation.
   The result is
   [a21-a21*(a11/a11]x1+[a22-a21*(a12/a11]x2+…….=b2-a21*(b11/a11)
   a21-a21*(a11/a11)=0
   Thus, the resultant equation does not contain x1. the new second equation is
   0+a’22x2+….+a’2nxn=b’2
   (iii) Similarly, subtract from the third equation. a31 times the normalized first equation.
   The result would be
   0+a’32x2+…….+a’3nXn=b’3
   if we repeat this procedure till the n^{th} equation is operated on, we will get the following new system of equation:
   a11x1+a12x2+…..+a1xn=b1
   a’22x2+…..+a’2nxn=b’2
   . . .
   . . .
   . . .
   a’n2x2+….+a’nnxnn+b’n

The solution of these equation is same as that of the original equation

3. Eliminate x2 from the third to last equation in the new set.
   Again, we assume that a’22!=0
   (i) Subtract from the third equation a’32 times the normalized second equation.
   (ii) Subtract from the fourth equation, a’42 times the normalized second equation and so on.

This process will continue till the last equation contains only one unknown, namely, xn. The final form of the equations will look like this:

A11 x1+a12 x2+…+a1n xn=b1
a'22x2+….+a’2n xn+b’2
………
………

This process is called triangulation. The number of primes indicate the number of times the coefficient has been modified.

4. Obtain solution by back substitution.
   The solution is as follows:
   Xn=bn(n-1)/ann(n-1)
   This can be subsitituted back in the (n-1) th equation to obtain the solution for xn-1.
   This back substitution can be continued till we get the solution for x1.

PROGRAM: -

```
/*gauss elimination method*/
#include<stdio.h>
#define N 4
main()
{
float a[N][N+1],x[N],t,s;
int i,j,k;
clrscr();
printf("enter the elements of the augmented matrix rowwise\n");
for(i=0;i<N;i++)
for(j=0;j<N+1;j++)
scanf("%f",&a[i][j]);
for(j=0;j<N-1;j++)
for(i=j+1;i<N;i++)
{
t=a[i][j]/a[j][j];
for(k=0;k<N+1;k++)
a[i][k]-=a[j][k]*t;
}
/*now print the upper triangular matrix*/
printf("the upper triangular matrix is \n");
for(i=0;i<N;i++)
{
for(j=0;j<N+1;j++)
printf("%8.4f",a[i][j]);
printf("\n");
}
/*now performing back substitution*/
for(i=N-1;i>=0;i--)
{
s=0;
for(j=i+1;j<N;j++)
s+=a[i][j]*x[j];
x[i]=(a[i][j]-s)/a[i][i];
}
/*now printing the result*/

printf("sol is \n");
for(i=0;i<N;i++)
printf("x[%3d]=%8.4f\n",i+1,x[i]);
getch();
}
```

Result: -

Enter the elements of augmented matrix row wise.
…………………….
………………….
………………….
………………….
The upper triangular matrix is: -
…………………..
…………………..
…………………..
The solution is: -
X[1]=
X[2]=
….
….
X[n]=

EXPERIMENT NO.- 4

Aim: - Solution of a system of simultaneous algebraic equation using Gauss – siedel iterative method.

ALOGRITHM: -

1. Obtain n, aij and bi values
2. Set xi=bi/aii   fir i=1 to n
3. Set key =0
4. for i=1 to n
    (i) Set sum = bi
    (ii) For j=1 to n (j#i)

        Set sum = sum – aij xj
        Repeat j
    (iii) Set dummy= sum/aij
    (iv) If key =0 then
        [(dummy – si)/dummy]>error then
        Set key = 1
    (v) Set xi=dummy
        Repeat i
5. If key = 1 then go to step 3
6. Write results.

PROGRAM:-

```
/* gauss sedial method */
#include<stdio.h>
#include<math.h>
#define N 4
 main ()
{

float a[N][N+1],x[N],aerr,maxerr,t,s,err;
int i,j,itr,maxitr;
clrscr();
for(i=0;i<N;i++)
x[i]=0;
printf("Enter the element of argument matrix row wise \n");
for(i=0;i<N;i++)
for(j=0;j<N+1;j++)
scanf("%f",&a[i][j]);
printf("Enter the allowed error,maximum iterations\n");
scanf("%f %d",&aerr,&maxitr);

printf("Iterations  x[1]      x[2]    x[3]\n");
for(itr =1;itr<=maxitr;itr++)
{
```

```
maxerr=0;
for(i=0;i<N;i++)
{
s=0;
for(j=0;j<N;i++)
if(j!=i)
s+=a[i][j]*x[j];
t=(a[i][N]-s)/a[i][i];
err=fabs(x[i]-t);
if(err!=maxerr)
maxerr=err;
x[i]=t;
}}
printf("%5d",itr);
for(i=0;i<N;i++)
printf("%9.4f",x[i]);
printf("\n");
if(maxerr<aerr)
{
printf("Coverage in %3d iterations \n",itr);
for(i=0;i<N;i++)
printf("x[%3d] = %7.4f\n", i+1,x[i]);
return 0;
}
}
printf (“Solution does not coverage,” “iteration not sufficient \n”);
return 1;
}
}
```

Result: -

Enter the elements of augmented matrix rowwise

……….

……….

……….

Enter the allowed irror, maximum iterations

……….

Iteration	x(1)	x(2)	x(3)
……….	….	….	….
……….	….	….	….

Converges in iterations

X[1]=

X[2]=

X[3]=

EXPERIMENT NO.– 5

Aim: - Numerical solution of an ordinary differential equation using the Euler's method.

PROGRAM: -

```
/* EULERS' METHOD */

#include<stdio.h>
#include<conio.h>
#include<math.h>

float df(float x,float y)
{
return x+y;
}

main()
{
float x0,y0,x,x1,y1,h;
clrscr();
printf("enter the values of x0,y0,h,x");
scanf("%f %f %f %f",&x0,&y0,&h,&x);
x1=x0;
y1=y0;
while(x1<1)
{
if(x1>x)
return;
y1+=h*df(x1,y1);
x1=x1+h;
printf("when x=%3.1f  ; y=%4.2f\n",x1,y1);
}
}
```

Result: -

Enter the values of x0, y0, h, x.

……………………

When x=….. y =………

………………………..

EXPERIMENT NO.- 6

AIM: - Numerical solution of an ordinary differential equation using the Runga – Kutta 4th order method.

PROGRAM: -

```
/* Runga- kutta method */


#include<stdio.h>
#include<conio.h>
#include<math.h>

float f(float x,float y)
{
return x+y*y;
}

main()
{
float x0,y0,h,xn,x,y,k1,k2,k3,k4,k;
clrscr();
printf("enter the values of x0,y0,h,xn \n");
scanf("%f %f %f %f",&x0,&y0,&h,&xn);
x=x0;
y=y0;
while(1)
{
if(x==xn) break;
k1=h*f(x,y);
k2=h*f(x+h/2,y+k1/2);
k3=h*f(x+h/2,y+k2/2);
k4=h*f(x+h,y+k3);
k=(k1+(k2+k3)*2+k4)/6;
x=x+h;
y+=k;
printf("when x=%8.4f" " y=%8.4f \n",x,y);
}
}
```

Result: -

Enter the values of x0, y0, h, xn.

....

When x=..... y =.........

....

EXPERIMENT NO.– 7

Aim: - Numerical solution of an ordinary equation using the Predictor – Corrector method.
PROGRAM: -

PROGRAM: -

```
/* Runga- kutta method */


#include<stdio.h>
#include<conio.h>
#include<math.h>

float f(float x,float y)
{
return x+y*y;
}

main()
{
float x0,y0,h,xn,x,y,k1,k2,k3,k4,k;
clrscr();
printf("enter the values of x0,y0,h,xn \n");
scanf("%f %f %f %f",&x0,&y0,&h,&xn);
x=x0;
y=y0;
while(1)
{
if(x==xn) break;
k1=h*f(x,y);
k2=h*f(x+h/2,y+k1/2);
k3=h*f(x+h/2,y+k2/2);
k4=h*f(x+h,y+k3);
k=(k1+(k2+k3)*2+k4)/6;
x=x+h;
y+=k;
printf("when x=%8.4f" " y=%8.4f \n",x,y);
}
}
```

Result: -

Enter the values of x0, y0, h, xn.

……………………

When x=….. y =………

………………………