Lab 1: Bisection Method

#Objective:

To calculate the real root of $x^3+x^2-3x-3=0$ correct upto 3 decimal points using bisection method.

#Tools:

DevC++

#Algorithm:

1. Start
2. Define f(x)
3. Read a, b, E
4. If f(a) * f(b) > 0
   Print: Error, IVP is not satisfied, stop the program.
5. Else:
   do $c \leftarrow \frac{a+b}{2}$
   if f(c) . f(a) < 0 then
   $b \leftarrow c$
   else
   $a \leftarrow c$
   end if
   while $|f(c)| \geq E$
6. Print : c
7. Stop

Code:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) ((x)*(x)*(x) + (x)*(x) - 3(x) - 3)

    int main() {
        float x1, x2, xm = 0, acc = 0.001, temp;
        do{
          Printf ("Enter the first number = ");
          scanf ("%f", &x1);
          Printf ("Enter the second number = ");
          scanf ("%f", &x2);
        }
        while (f(x1) * f(x2) >= 0);
        do{
          temp = xm;
          xm = (x1 + x2) / 2;
          if (f(x1) * f (xm) < 0)
            {
              x2 = xm;
            }
            else
              x1 = xm;
        }
        while (fabs(temp - xm) >= acc);
        Printf("The real roots is : %.4f", xm);
        return 0;
    }
```

#. Observation and Result:

Enter the first number = 1

Enter the second number = 2

The real root is: 1.7314

Conclusion:

The real root is 1.7314 of Bisection Method.

Lab 2: Regula Falsi Method:

#Objectives:

To calculate the real root of $x^3 - 2x - 5 = 0$ correct upto 3 decimal points using Regula Falsi Method.

Tools:

Dev C++

Algorithm:

1. Start
2. Define function $f(x)$ and error E.
3. Input two initial guess 'a' and 'b'
4. Compute $f(a)$ and $f(b)$.
5. If $f(a) * f(b) > 0$
   Print: Error, root doesn't exit, stop the program
6. Else:
   calculate root : $xn = \frac{af(b) - bf(a)}{f(b) - f(a)}$
   calculate $f(xn)$
7. If $f(xn) > 0$
   set $b = xn$
   $f(b) = f(xn)$

   Else:
   $a = xn$
   $f(a) = f(xn)$

8. Repeat till step '7' until absolute value of $\frac{b-a}{b}$ is less than E then print root.

$$root = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

9. Stop

Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)((x)*(x)*(x)-2(x)-5)
    int main(){
        float x1,x2,x0=0, acc=0.001, temp;
        do{
         Printf("Enter the first number=");
         scanf("%f", &x1);
         Printf("Enter the second number=");
         scanf("%f", &x2);
        }
        while(f(x1)*f(x2)>=0);
         do{
           temp=x0;
           x0=x1-(f(x1)* (x2-x1)/f(x2)-
                f(x1));
           x1=x0;
         }
```

```
    while(fabs(temp-x0)>=acc);
    Printf("The real root is : %.0.4f", x0);
    return 0;
}
```

#. Observation And Result:

Enter the first number: 2
Enter the second number: 3
The real root is: 2.0943

#. Conclusion:

The real root is 2.0943 of Regula Falsi Method.

Lab 3: Secant Method

#Objective:

To calculate the real root of given equation $xe^x - \cos x$ correct upto 3 decimal points using secant method.

#Tools:

Dev C++

#Algorithm:

1. Start
2. Define function f(x) and Error E.
3. Input initial guess 'a' and 'b'.
4. Compute f(a) and f(b).
5. If (f(a)*f(b)>0) then root doesn't lies between 'a' and 'b', stop the program.
6. Else:
   calculate root: $xn = \frac{af(b) - bf(a)}{f(b) - f(a)}$ and f(xn).
7. If absolute value of error $\frac{b-a}{b}$ is less than error 'E' then print:

   $$root = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

   Else:
   set a = b
   b = xn
   f(a) = f(b)
   f(b) = f(xn)
8. Stop.

#Code:

```
#include<stdio.h>
#include<math.h>
#define f(x)((x)* exp(x)-cos(x));
    int main(){
        float x1,x2,x3=0, acc = 0.0001, temp;
        Printf ("Enter the first number =");
         scanf ("%.f" ,&x1);
         Printf ("Enter the second number =");
         scanf ("%.f",&x2);
         do{
            temp = x3;
             x3 = (x1 * f(x2) - x2 * f(x1) / f(x2)-f(x1))
              x1 = x2;
               x2 = x3;
            }
            while(fabs (temp-x3) >= acc);
             Printf ("The real root is: %.0.5f", x3);
              return 0;

        }
```

#. Observation & Result:

Enter the first number = 5
Enter the second number = 6
The real root is : 0.51776

Lab 5: Fixed Point Method

Lab 4: Newton Raphson Method

#. Objective:

To calculate the real root of $3x - \cos x - 1$ correct upto 3 decimal places using N-R method.

#. Tools:

Dev C++

#. Algorithm:

1. Start
2. Define function $f(x)$, Derivative of function $f'(x)$ and Error 'E'.
3. Input initial guess x_n.
4. Compute $f(x_n)$ and $f'(x_n)$.
5. Calculate $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
6. If absolute value of $\left|\frac{x_{n+1} - x_n}{x_{n+1}}\right| < F$ then
   print root $= x_n - \frac{f(x_n)}{f'(x_n)}$
7. Stop

#. Code:

```
#include<stdio.h>
#include<math.h>
#define f(x) (3*(x)-cos(x)-1)
#define g(x) (3+sin(x))
int main(){
    float x1,x2=0, temp, acc=0.001;
    Printf("Enter the initial guess");
    scanf("%f", &x1);
    do{
      temp=x2;
      x2=x1-f(x1)/g(x1);
      Printf("\n %.0.4f", x1);
      Printf("\n%.0.4f", x2);
        x1=x2;
    }while(fabs(temp-x2)>=acc);
      Printf("The real root is %.0.4f", x2);
      return 0;
}
```

#. Observation & Result:

```
Enter the initial guess 1
    1.0000
    0.6200
    0.6200
    0.6071
    0.6071
```

The real root is 0.6071.

Lab 5: Fixed Point Method

#. Objective:

To calculate the real root of $x^2 - 3x + 2 = 0$ correct upto 3 decimal points.

#. Tools:

Dev C++

#. Algorithm:

1. Start
2. Define function $f(x)$.
3. Define function $g(x)$ which is obtained from $f(x)=0$ such that $x = g(x)$ and $|g'(x) < 1|$.
4. Choose initial guess $x0$, Error 'E' and xn.
5. Initial iteration counter : step = 1
6. Calculate $x1 = g(x0)$
7. Increment iteration counter : step = step + 1
8. If step > N
   Print: Not convergent, stop the program.
9. set $x0 = x1$
10. If $|f(x1)| > E$, then go to step 6
11. Display $x1$ as root.
12. Stop.

#. Code:

```
#include<stdio.h>
#include<math.h>
#define g(x)((x*x+2)/3)
    int main() {
        float x, y=0, temp, acc=0.001;
        Printf("Enter the number = ");
        scanf("%f",&x);
        do{
          temp=y;
           y=g(x);
           x=y;
         }while(fabs(temp-y)>=acc);
         Printf("The real root is %.0.4f", y);
         return 0;
    }
```

#. Observation & Result:

Enter the number = 2

The real root is 2.0000

Lab 6: Gauss Jacobi Method:

#. Objective:

To solve the equation $20x + y - 2z = 17$, $3x + 20y - z = -18$, $2x - 3y + 20z = 25$ using Gauss Jacobi Method correct upto 0.01.

#. Tools:

Dev C++

#. Algorithm:

1. Start
2. Arrange given system of linear equation in diagonally dominant in
3. Read tolerable error (ϵ)
4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.
5. Set initial guess for $x0, y0, z0$ and so on.
6. Substitute value of $x0, y0, z0$.... from step 5 in equation obtained in step 4 to calculate new values $x1, y1, z1$ and so on.
7. If $|x0 - x1| > e$ and $|y0 - y1| > e$ and $|z0 - z1| > e$ and so on then stop the program.
8. Set $x0 = x1$, $y0 = y1$, $z0 = z1$ and so on.
9. Print value of $x1, y1, z1$ and so on.
10. Stop.

code:

```
#include<stdio.h>
#include<math.h>
int main(){
    float x0,y0,z0,x1=0,y1=0,z1=0,acc=0.01
    tempx,tempy,tempz;
    Printf("Enter the initial guess");
    scanf("%f %f %f",&x0,&y0,&z0);
    do{
        tempx=x0;
        tempy=y0;
        tempz=z0;
        x1=(17-y0+2*z0)/20;
        y1=(-18+z0-3*x0)/20;
        z1=(25-2*z0+3*y0)/20;
        x0=x1;
        y0=y1;
        z0=z1;
    }
    while(fabs(tempx-x1)>acc &&
    fabs(tempy-y1)>=acc&&
    fabs(tempz-z1)>=acc);
    Printf("Solutions are x=%.0.3f, y=
    %.0.3f, z=%.0.3f,"x1,y1,z1);
    return 0;
}
```

#. Observation & Result

Enter the initial guess 0 0 0

Solutions are $x = 1.000$, $y = -1.000$, $z = 1.000$