Lab-1

Program Development of Bisection & Newton Raphson's Method

Theory

Bisection Method:

→ If $f(x)$ is continuous in the interval (x_1, x_2) and $f(x_1)$ & $f(x_2)$ have different sign i.e. $f(x_1) * f(x_2) < 0$, then the equation $f(x) = 0$ has at least one root in the interval (x_1, x_2).

Now, the next point for next iteration,

$$x_3 = \frac{x_1 + x_2}{2}.$$

Newton Raphson's Method:

→ Let $f(x)$ be a real and continuous function

- x_0 be the initial approximation.
- Let's draw a tangent to the curve at point x_0.
- The point at which the tangent crosses the x-axis will be the improved estimated of the root.

∴ slope of the tangent,

$$\tan \alpha = \frac{f(x_0)}{x_0 - x_1}$$

Now,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Similarly,

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

So in general,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Algorithm

1- For Bisection Method:

1- Define a function $f(x)$.
2. Input the tolerable error E.
3. Input initial guesses x_1 & x_2.
4. Find $f(x_1)$ & $f(x_2)$.
5. Check if $f(x_1) * f(x_2) < 0$, if so continue otherwise go to step 3.
6. Compute $x_3 = (x_1 + x_2)/2$ & find $f(x_3)$.
7. If $f(x_1) * f(x_3) < 0$, set $x_2 = x_3$ & $f(x_2) = f(x_3)$
   else set $x_1 = x_3$ & $f(x_1) = f(x_3)$.
8. Check $|f(x_3)| <= E$, if so root is x_3 & continue
   else go to step 5.
9. stop.

→ For Newton Raphson's Method

1. ~~x~~ Define a function $f(x)$ & $f'(x)$
2. Input the tolerable error E.
3. Input initial guesses x_1.
4. Find $f(x_1)$ & $f'(x_1)$.
5. Chek if $f(x_1) = 0$, go to step 2 else continue.
6. Compute Next approximation
   $x_2 = x_1 - f(x_1)/f'(x_1)$ & find $f(x_2)$
7. Chek if $|f(x_2)| < E$, if so root is x_2 & stop.
   else $x_1 = x_2$, $f(x_1) = f(x_2)$ & $f'(x_1) = f'(x_2)$
   go to step 6.
8. stop.

Program

Bisection Method

Program

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float calculate(float x){
        return x*x*x + x*x - 3*x - 3;
}
int main( )
{
    float x1, x2, x3, f1, f2, f3, error = 0.001, root;
    up: printf ("Enter x1: ");
        scanf ("%f", &x1);
        printf ("Enter x2: ");
        scanf ("%f", &x2);
        f1 = calulate(x1);
        f2 = calculate (x2);
    next: if ((f1*f2) < 0){
            x3 = (x1+x2)/2;
            f3 = calculate (x3);
            if ( f1* f3 <0){
                x2 = x3
                f2 = f3;
            }
            else {
                x1 = x3
                f1 = f3;
            }
            if (fabs (f3)<=error){
                root = x3;
                printf ("Root is %f", root);
                return 0;
            }
            else {
                goto next;
            }
    }
    else{
        goto up;
    }
}
```

Newton Raphson's Method

Program

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float calculate (float x){
        return x*x+4*x-9;
}
float derivative(float x){
        return 2*x+4;
}
int main()
{
    int iteration = 1;
    float x1, x2, x3, f1, fd, error = 0.001;
    up: printf ("Enter x1: ");
        scanf ("%f", &x1);
        f1 = calculate (x1).
        fd = derivative (x1);
        if (fd == 0){
            goto up;
        }
    next: x2 = x1 - (f1/fd);
        x3 = calculate (x2);
        printf ("%d\t %f\t %f\t %f \t %f \t %f\n", iteration,
          x1, x2, f1, fd, x3);
        iteration++;
        if (fabs(x3) < error){
            printf("Root is %f", x2);
            return 0;
        }
        else{
            x1 = x2;
            f1 = x3;
            fd = derivative (x1);
            goto next;
        }
}
```

Lab - 2 →

Program Development of Secant & Fixed Point Iteration Method.

Theory

Secant Method:

→ It is a root-finding procedure in numerical analysis that uses a series of roots of secant lines to better approximate a root of a function f.

Equation of Secant Method:

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

Fixed Point Method:

→ It is an iterative method to find the roots of algebric & transcendental equations by converting them to a fixed point function.

Algorithm

For Secant Method:

1. Define a function f(x).
2. Input the tolerable error E.
3. Input initial guesses x1 & x2.
4. Calculate f1 = f(x1) & f2 = f(x2).
5. Compute x3 = (x1 * f2 - x2 * f1)/(f2 - f1)
6. Calculate f3 = f(x3)
7. Check |(x3 - x2)/x3| <= E, if so root is x3 & stop.
8. Else set x1 = x2, x2 = x3 & f1 = f2, f2 = f3
9. Go to step 5.
10. End.

For Fixed Point Iteration Method

1. Define function f(x)
2. Define function g(x) from f(x)=0 such that $x = g(x)$ & $|g'(x)| < 1$
3. Input tolerable Error E.
4. Input initial guess x1.
5. Calculate $x2 = g(x1)$
6. If $|f(x2)| > E$ then

   x1 = x2

   goto step 5
7. Else root is x2.
8. End.

Program

For Secant Method

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

float function (float x){
        return 2*x*x+4*x-10;
}
int main()
{
    float x1,x2,x3,f1,f2,f3,error=0.05,root;
    printf("Enter two initial guesses: ");
    scanf("%f %f", &x1, &x2);
    f1= function(x1);
    f2 = function(x2);
    up: x3 = (x1*f2-x2*f1)/(f2-f1);
        f3 = function(x3);
        if( fabs(x3-x2)/x3 <= error){
            root = x3;
            printf("Root is %f", root);
            return 0;
        }
        else{
            x1=x2;
            x2=x3;
            f1=f2;
            f2=f3;
        } goto up;
}
```

For Fixed Point Iteration Method

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

float fx(float x){
    return x*x+x-2;
}
float gx(float x){
    return 2/(x+1);
}
int main()
{
    float x1, x2, root, e=0.0001;
    printf("Enter initial guess:");
    scanf("%f", &x1);
    if (fabs(-2/((x1+1)*(x1+1)))>1){
        printf("Root can't be converged.");
        return 0;
    }
    up: x2 = gx(x1);
        if(fabs(fx(x2))>e){
            x1=x2;
            goto up;
        }
        else{
            root=x2;
            printf("The root is %f", root);
            return 0;
        }
}
```