G.E. - Numerical Methods

Name - Arun Kumar

Roll no- 16015

Course - B.Sc.(Hons)Computer Science

Section - A

1.Bisection Method -

```
f[x_] := x^2 - 5;

a = 2;

b = 3;

c = 0;

e = 0.01;

n = 10;


Print["Bisection Method"];

If[f[a] * f[b] > 0,

  Print["No root exists in the given interval. Choose different values for a and b."],

  For[i = 0, i < n, i++,

    c = (b + a)/2;

    Print["The root at iteration ", i, " is approximately: ", N[c]];
```

```
    If[f[c] * f[b] > 0, b = c, a = c];
    If[Abs[b - a] < e,
      Print["Root found with tolerance at iteration ", i, ": ", N[c]];
      Break[];
    ];
  ]
];
```

2.<u>Secant method</u>

```
f[x_] := 3 + 2 x - (2 - 3 x - 1);
a = 1;
b = 2;
e = 0.01;
n = 10;

Print["The Secant Method"];
For[i = 0, i < n, i++,
  p = b - f[b] * (b - a) / (f[b] - f[a]);
  Print["The root at ", i, "th iteration: ", N[p]];
  a = b;
  b = p;
];
Plot[f[x], {x, 1, 2}]
```

```
NSolve[f[x] == 0, x]
```

3.<u>Regula - Falsi method -</u>

```
f[x_] := x^2 - 5;
a = 2;
b = 3;
e = 0.01;
n = 10;

If[f[a] * f[b] > 0,
  Print["Take another value for a and b."],
  For[i = 0, i < n, i++,
     p = b - f[b] * (b - a) / (f[b] - f[a]);
     Print["The root at ", i, "th iteration: ", N[p]];
     If[Abs(f[p]) < e,
        Print["Root found at iteration ", i, ": ", N[p]];
        Break[];
     ];
     If[f[p] * f[b] > 0, b = p, a = p];
  ]
];
Plot[f[x], {x, 2, 3}]
```

4. Newton-Raphson method-

```
f[x_] := x^3 + 2 x^2 - 3 x - 1;

df[x_] := D[f[x], x];

a = 1;

e = 0.0001;

n = 10;


Print["The Newton-Raphson Method"];

For[i = 0, i < n, i++,

   p = a - f[a]/df[a];

   Print["The root at ", i, "th iteration: ", N[p]];

   If[Abs[p - a] <= e,

     Print["Root found with tolerance at iteration ", i, ": ", N[p]];

     Break[];

   ];

   a = p;

];

Plot[f[x], {x, 1, 2}]

NSolve[f[x] == 0, x]
```

5. Gauss-Jacobi method -

```
n = 3;
a = {{5, 2, 1}, {3, 7, 4}, {1, 1, 9}};
MatrixForm[a]
x = {0, 0, 0};
y = {0, 0, 0};
b = {10, 21, 12};

For[k = 1, k <= 25, k++,
   For[i = 1, i <= n, i++,
      y[[i]] = (b[[i]] - Sum[a[[i, j]] * x[[j]], {j, 1, i - 1}] -
              Sum[a[[i, j]] * x[[j]], {j, i + 1, n}]) / a[[i, i]];
   ];
   For[m = 1, m <= n, m++, x[[m]] = N[y[[m]]]];
   For[p = 1, p <= n, p++, Print["x[", p, "] = ", x[[p]]]];
]
```

6.**Gauss-Seidel method -**

```
n = 3;
a = {{5, 2, 1}, {3, 7, 4}, {1, 1, 9}};
MatrixForm[a]
x = {0, 0, 0};
```

```
y = {0, 0, 0};
b = {10, 21, 12};


For[k = 1, k <= 25, k++,
   For[i = 1, i <= n, i++,
      y[[i]] = (b[[i]] - Sum[a[[i, j]] * y[[j]], {j, 1, i - 1}] -
              Sum[a[[i, j]] * y[[j]], {j, i + 1, n}]) / a[[i, i]];
   ];
   For[m = 1, m <= n, m++, x[[m]] = N[y[[m]]]];
   For[p = 1, p <= n, p++, Print["x[", p, "] = ", x[[p]]]];
]
```

7. Langrange Interpolation -

```
Clear[x];
sum = 0;


points = {{1, 2}, {2, 5}, {3, 10}};
No = Length[points];
Print["Given values of x[i] are as follows: ", y = points[[All, 1]]];
Print["Given values of f[x[i]] are as follows: ", f = points[[All, 2]]];


lagrange[No_, n_] := Product[
```

```
    If[Equal[k, n], 1, (x - y[[k]]) / (y[[n]] - y[[k]])],
    {k, 1, No}
];

For[i = 1, i <= No, i++,
    sum += f[[i]] * lagrange[No, i];
];
Print[sum];
Print["The polynomial function will be: ", Expand[sum]];
Print["Polynomial at x=2.5 is: ", sum /. x -> 2.5];
```

8.**Newton Interpolation -**

```
points = {{1, 2}, {2, 5}, {3, 10}, {4, 17}};
n = Length[points];
xVals = points[[All, 1]];
yVals = points[[All, 2]];
ddTable = Table[0, {i, n}, {j, n}];
For[i = 1, i <= n, i++, ddTable[[i, 1]] = yVals[[i]]];

For[j = 2, j <= n, j++,
    For[i = 1, i <= n - j + 1, i++,
        ddTable[[i, j]] = (ddTable[[i + 1, j - 1]] - ddTable[[i, j - 1]]) / (xVals[[i + j - 1]] - xVals[[i]]);
```

```
    ];
];

poly = ddTable[[1, 1]];
term = 1;
For[k = 2, k <= n, k++,
    term *= (x - xVals[[k - 1]]);
    poly += ddTable[[1, k]] * term;
];
Print["The Newton Interpolation Polynomial is: ", Expand[poly]];
Print["Value of the polynomial at x = 2.5: ", N[poly /. x -> 2.5]];
```

9.<u>Trapezoidal rule -</u>

```
a = Input["Enter the left-hand point of the interval"];
b = Input["Enter the right-hand point of the interval"];
h = b - a;
f[x_] := 1/x;
tz = (h/2) * (f[a] + f[b]);
Print["Trapezoidal estimate is: ", tz];
```

10.<u>Simpson rule -</u>

```
f[x_] := x^2 - 5;
```

```
a = 2;

b = 3;

n = 6;

h = (b - a)/n;


sum = f[a] + f[b];

For[i = 1, i <= n - 1, i++,

   If[Mod[i, 2] == 0, sum += 2 * f[a + i * h], sum += 4 * f[a + i * h]];

];

integral = (h/3) * sum;  (* Final result using Simpson's Rule *)

Print["The approximate integral is: ", integral];
```

11.<u>Euler's method -</u>

```
f[x_, y_] := x + y;

x0 = 0;

y0 = 1;

h = 0.1;

xEnd = 1;

x = x0;

y = y0;

Print["x = ", x, ", y = ", y];
```

```
While[x < xEnd,
    y = y + h * f[x, y];
    x = x + h;
    Print["x = ", x, ", y = ", y];
]
Print["Final value at x = ", xEnd, " is: ", y];
```

12.<u>Rungekutta method -</u>

```
f[x_, y_] := x + y;
x = 0;
y = 1;
h = 0.1;
xTarget = 1.7;

While[x < xTarget,
    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];
    y = y + (k1 + 2*k2 + 2*k3 + k4)/6;
    x = x + h;
    Print["x=", x, ", y=", y];
```

```
]

Print["Final value of y at x=", xTarget, " is: ", y];
```