TUTORIAL 4

**Code:**

#include"bits/stdc++.h"

using namespace std;

double f(double x)

{

    return x - (exp(x) / 3);

}

double df(double x)

{

    return 1 - (exp(x) / 3);

}

double g(double x)

{

    return (exp(x) / 3);

}

pair<double, int> bisection(double a, double b, double tole = 1e-7)

{

    double c;

    int iter = 0;

    while ((b - a) / 2.0 > tole)

    {

        c = (a + b) / 2.0;

        double fa = f(a);

        double fc = f(c);

        if (fc == 0)

        {

            return make\_pair(c, iter);

        }

        if (fa \* fc < 0)

        {

            b = c;

        }

        else

        {

            a = c;

        }

        iter++;

    }

    c = (a + b) / 2.0;

    return make\_pair(c, iter);

}

pair<double, int> raphson(double x0, double tole = 1e-7)

{

    double x = x0;

    int iter = 0;

    while (true)

    {

        double fx = f(x);

        if (fabs(fx) < tole)

        {

            return make\_pair(x, iter);

        }

        double dfx = df(x);

        if (dfx == 0)

        {

            cerr << "Derivative is zero, can't converge by this method." << endl;

            exit(1);

        }

        x = x - fx / dfx;

        iter++;

    }

}

pair<double, int> fp(double x0, double tole = 1e-7)

{

    double x = x0;

    int iter = 0;

    while (true)

    {

        double x\_next = g(x);

        double error = abs(x\_next - x);

        if (error < tole)

        {

            return make\_pair(x\_next, iter);

        }

        x = x\_next;

        iter++;

    }

}

pair<double, int> secant(double x0, double x1, double tole = 1e-7)

{

    double x2;

    int iter = 0;

    while (true)

    {

        double fx0 = f(x0);

        double fx1 = f(x1);

        x2 = x1 - (fx1 \* ((x1 - x0) / (fx1 - fx0)));

        double error = abs(x2 - x1);

        if (error < tole)

        {

            return make\_pair(x2, iter);

        }

        x0 = x1;

        x1 = x2;

        iter++;

    }

}

pair<double, int> falsi(double l, double r, double tole = 1e-7)

{

    double c;

    int iter = 0;

    while (true)

    {

        double fl = f(l);

        double fr = f(r);

        if (fl \* fr >= 0)

        {

            cerr << "Can't converge because of same sign of both." << endl;

            exit(1);

        }

        c = l - fl \* ((r - l) / (fr - fl));

        if (abs(l - c) < tole)

        {

            return make\_pair(c, iter);

        }

        double fc = f(c);

        if (fc == 0)

        {

            return make\_pair(c, iter);

        }

        if (fl \* fc < 0)

        {

            r = c;

        }

        else

        {

            l = c;

        }

        iter++;

    }

}

int main()

{

    cout << setprecision(8)<<fixed;

    double ig1;

    double ig2;

    double ig3, tole;

    tole = 1e-7;

    fstream tt;

    tt.open("input.txt",ios::in);

    tt>>ig1 ;

    tt>>ig2;

    tt>>ig3;

    tt.close();

    tt.open("output.txt",ios::out);

    tt << "Tolerance "<< tole << endl;

    tt << endl;

    pair<double, int> result0 = bisection(ig1, ig2);

    tt << "BISECTION" << endl;

    tt << "Initial guesses " << ig1 <<" "<< ig2  << endl;

    tt << "Approx root " << result0.first << endl;

    tt << "No. of iterations " << result0.second << endl;

    tt << endl;

    pair<double, int> result = raphson(ig3);

    tt << "NEWTON RAPHSON" << endl;

    tt << "Initial guesses " << ig3<< endl;

    tt << "Approx root " << result.first << endl;

    tt << "No. of iterations " << result.second << endl;

    tt << endl;

    pair<double, int> result1 = fp(ig3);

    tt << "FIXED POINT ITERATION" << endl;

    tt << "Initial guess " << ig3 << endl;

    tt << "Approx root " << result1.first << endl;

    tt << "No. of iterations " << result1.second << endl;

    tt << endl;

    pair<double, int> result2 = secant(ig1, ig2);

    tt << "SECANT" << endl;

    tt << "Initial guesses " << ig1 <<" "<< ig2  << endl;

    tt << "Approx root " << result2.first << endl;

    tt << "No. of iterations " << result2.second << endl;

    tt << endl;

    pair<double, int> result3 = falsi(ig1, ig2);

    tt << "REGULA FALSI" << endl;

    tt << "Initial guesses " << ig1 <<" "<< ig2  << endl;

    tt << "Approx root " << result3.first << endl;

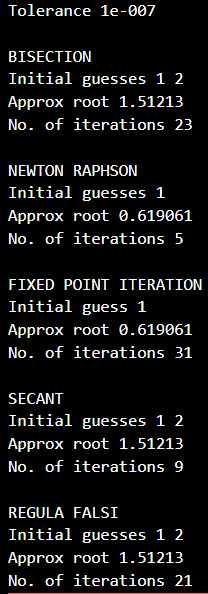
    tt << "No. of iterations " << result3.second << endl;

    tt << endl;

    return 0;

}

**Output:**

****

**Inference:**

Using 5 methods, we found the roots of given equation which came out to be 0.619061 and 1.51213 and found out that Newton Raphson method was fastest among all.