Rajat Kumar Mandal - 226121014


```cpp
#include <string>
#include <iostream>
#include <fstream>
#include <cmath>
#include <array>
#include <map>

//Constant expressions appearing in the problem
constexpr size_t dimension = 2;   //dimension of the reduced 1st-order problem
constexpr double PI = 3.14159265359;    //value of PI

//Definition of data types in the problem
typedef std::array<double, dimension> state_type;  //data type definition for dependant variables - array of x_0, x_1, ... x_n
typedef std::map<double, state_type> solution;   //data type definition for storing the list of calculated values ((hash)map of time -> state)

//Overload the + operator to be able to add two vectors
state_type operator + (state_type const& x, state_type const& y) {
    state_type z;
    for (size_t i = 0; i < dimension; i++) {
        z[i] = x[i] + y[i]; //add the individual components and store in z
    }
    return z;   //return the resulting vector z
}

//Overload the * operator to be able to multiply numbers and vectors
state_type operator * (double const& a, state_type const& x) {
    state_type z;
    for (size_t i = 0; i < dimension; i++) {
        z[i] = a * x[i];    //multiply the individual components and store in z
    }
    return z;   //return the resulting vector z
}

//This is the differential Equation, reduced to first-order
void Pendulum(const state_type& x, const double& t, state_type& dxdt) {
    dxdt[0] = x[1];
    dxdt[1] = -PI * PI * sin(x[0]);
}

//The stepper function, iteratively calculates x_{n+1} given the differential equation, x_{n} and step size
void rk4_step(void (*Diff_Equation)(const state_type& x, const double& t, state_type& dxdt), state_type& x, const double& t, const double& dt) {
    //temporary variables for intermediate steps
    state_type k1, k2, k3, k4;
```

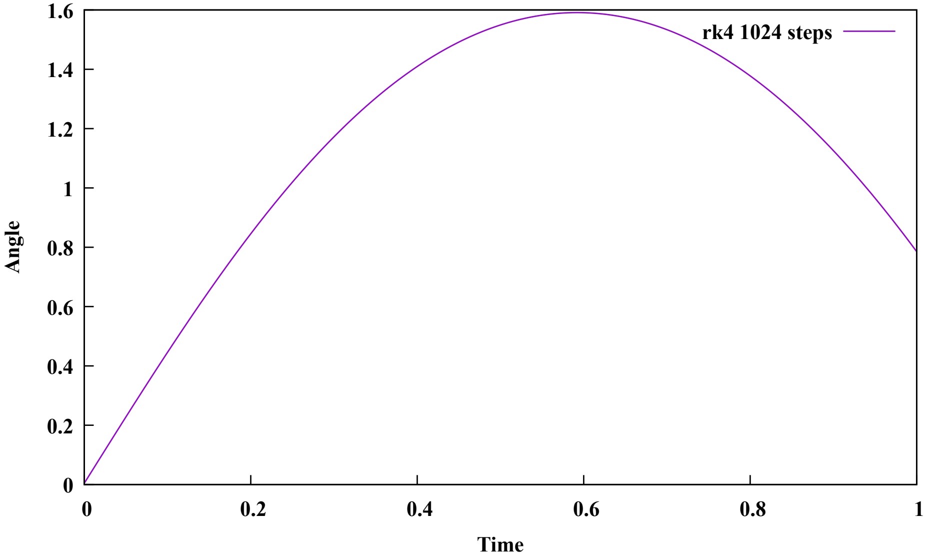
```cpp

    //calculate the intermediate values
    Diff_Equation(x, t, k1);    //calculate k1
    Diff_Equation(x + (dt / 2.0) * k1, t + dt / 2.0, k2);   //calculate k2
    Diff_Equation(x + (dt / 2.0) * k2, t + dt / 2.0, k3);   //calculate k3
    Diff_Equation(x + dt * k3, t + dt, k4); //calculate k4

    //calculate x_{n+1} using the RK4 formula and return the results
    x = x + (dt / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4);
}

int main() {
    solution x_t_0, x_t_1, x_t_12;   //variable to store the calculations

    size_t STEPS = 1024;  //number of steps
    double t_0 = 0.0;   //initial time
    double t_1 = 1.0;   //final time
    double dt = (t_1 - t_0) / (STEPS - 1); //step size

    size_t iteration = 0;
    double left = 0, right = 10, middle = 5;

    while (iteration < 100) {
        state_type x = { 0.0, left };   //initial values for dependant variables
        //Step through the domain of the problem and store the solutions
        x_t_0[t_0] = x;   //store initial values
        for (size_t i = 0; i < STEPS; i++) {
            rk4_step(Pendulum, x, NULL, dt);    //step forward
            x_t_0[t_0 + i * dt] = x;  //store the calculation
        }
        x = { 0.0, right };   //initial values for dependant variables
        x_t_1[t_0] = x;   //store initial values
        for (size_t i = 0; i < STEPS; i++) {
            rk4_step(Pendulum, x, NULL, dt);    //step forward
            x_t_1[t_0 + i * dt] = x;  //store the calculation
        }
        middle = (left + right) / 2.0;
        x = { 0.0, middle };   //initial values for dependant variables
        x_t_12[t_0] = x;   //store initial values
        for (size_t i = 0; i < STEPS; i++) {
            rk4_step(Pendulum, x, NULL, dt);    //step forward
            x_t_12[t_0 + i * dt] = x;  //store the calculation
        }
        double l = x_t_0[1][0] - PI / 4, r = x_t_1[1][0] - PI / 4, m = x_t_12[1][0] - PI / 4;
        if ((l > 0 && r > 0) || (l < 0 && r < 0)) {
            return -1;
        }
```

```cpp

        if ((l < 0 && m < 0) || (l > 0 && m > 0)) {
            left = middle;
        }
        else if ((r < 0 && m < 0) || (r > 0 && m > 0)) {
            right = middle;
        }
        else {
            return -1;
        }

        std::cout << iteration << "\t" << left << "\t" << l << "\t" << right << "\t" << r << std::endl;
        iteration++;
    }



    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("./output/BVP.txt", std::ios::out | std::ios::trunc);
    for (auto const& temp : x_t_12) {
        outfile << temp.first << "\t" << temp.second[0] << "\t" << temp.second[1] << std::endl;
    }
    outfile.close();
}
```

Output for bisection iterations:

x'(0) converges to 0.00438692 for x(1)=pi/4

0	0	-0.785398	5	8.14313
1	2.5	-0.785398	5	0.39954
2	3.75	-0.679297	5	0.39954
3	4.375	-0.373822	5	0.39954
4	4.375	-0.0706149	4.6875	0.39954
5	4.375	-0.0706149	4.53125	0.139387
6	4.45312	-0.0706149	4.53125	0.0286944
7	4.45312	-0.0223153	4.49219	0.0286944
8	4.47266	-0.0223153	4.49219	0.00284247
9	4.48242	-0.00982215	4.49219	0.00284247
10	4.4873	-0.00351139	4.49219	0.00284247
11	4.4873	-0.000339866	4.48975	0.00284247
12	4.4873	-0.000339866	4.48853	0.00124995
13	4.4873	-0.000339866	4.48792	0.000454704
14	4.48761	-0.000339866	4.48792	5.73348e-05
15	4.48776	-0.000141287	4.48792	5.73348e-05
16	4.48776	-4.19812e-05	4.48784	5.73348e-05
17	4.4878	-4.19812e-05	4.48784	7.67545e-06
18	4.48782	-1.71532e-05	4.48784	7.67545e-06
19	4.48782	-4.73897e-06	4.48783	7.67545e-06
20	4.48782	-4.73897e-06	4.48783	1.46822e-06
21	4.48783	-1.63538e-06	4.48783	1.46822e-06
22	4.48783	-8.35825e-08	4.48783	1.46822e-06
23	4.48783	-8.35825e-08	4.48783	6.92318e-07
24	4.48783	-8.35825e-08	4.48783	3.04367e-07
25	4.48783	-8.35825e-08	4.48783	1.10392e-07
26	4.48783	-8.35825e-08	4.48783	1.3405e-08
27	4.48783	-3.50887e-08	4.48783	1.3405e-08
28	4.48783	-1.08419e-08	4.48783	1.3405e-08
29	4.48783	-1.08419e-08	4.48783	1.28156e-09
30	4.48783	-4.78015e-09	4.48783	1.28156e-09
31	4.48783	-1.7493e-09	4.48783	1.28156e-09
32	4.48783	-2.33864e-10	4.48783	1.28156e-09
33	4.48783	-2.33864e-10	4.48783	5.23851e-10
34	4.48783	-2.33864e-10	4.48783	1.44991e-10
35	4.48783	-4.44352e-11	4.48783	1.44991e-10
36	4.48783	-4.44352e-11	4.48783	5.02776e-11
37	4.48783	-4.44352e-11	4.48783	2.92344e-12
38	4.48783	-2.07542e-11	4.48783	2.92344e-12
39	4.48783	-8.91676e-12	4.48783	2.92344e-12
40	4.48783	-3.00016e-12	4.48783	2.92344e-12
41	4.48783	-3.21965e-14	4.48783	2.92344e-12
42	4.48783	-3.21965e-14	4.48783	1.44218e-12
43	4.48783	-3.21965e-14	4.48783	7.00329e-13
44	4.48783	-3.21965e-14	4.48783	3.25073e-13
45	4.48783	-3.21965e-14	4.48783	1.48215e-13
46	4.48783	-3.21965e-14	4.48783	5.54001e-14
47	4.48783	-3.21965e-14	4.48783	1.04361e-14
48	4.48783	-8.88178e-15	4.48783	1.04361e-14
49	4.48783	-8.88178e-15	4.48783	2.77556e-15
50	4.48783	-7.77156e-16	4.48783	2.77556e-15
51	4.48783	-7.77156e-16	4.48783	1.22125e-15
52	4.48783	-1.11022e-16	4.48783	1.22125e-15
53	4.48783	-1.11022e-16	4.48783	5.55112e-16
54	4.48783	-1.11022e-16	4.48783	3.33067e-16



rk4 1024 steps
Angle
Time
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6