```cpp
#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
constexpr double rollnum = 0.226121014;  //my roll number

//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] = -4.0 * 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;   //variable to store the calculations

    size_t STEPS = 1000;  //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
    state_type x = {0.0, rollnum};   //initial values for dependant variables

    //Step through the domain of the problem and store the solutions
    x_t[t_0] = x;   //store initial values
    for (size_t i = 0; i < STEPS; i++) {
        rk4_step(Pendulum, x, NULL, dt);    //step forward
        x_t[t_0 + i * dt] = x;  //store the calculation
    }

    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("rk4.txt", std::ios::out | std::ios::trunc );
    for (auto const& temp : x_t){
        outfile << temp.first << "\t" << temp.second[0] << "\t" << temp.second[1] << std::endl;
    }
    outfile.close();
}
```