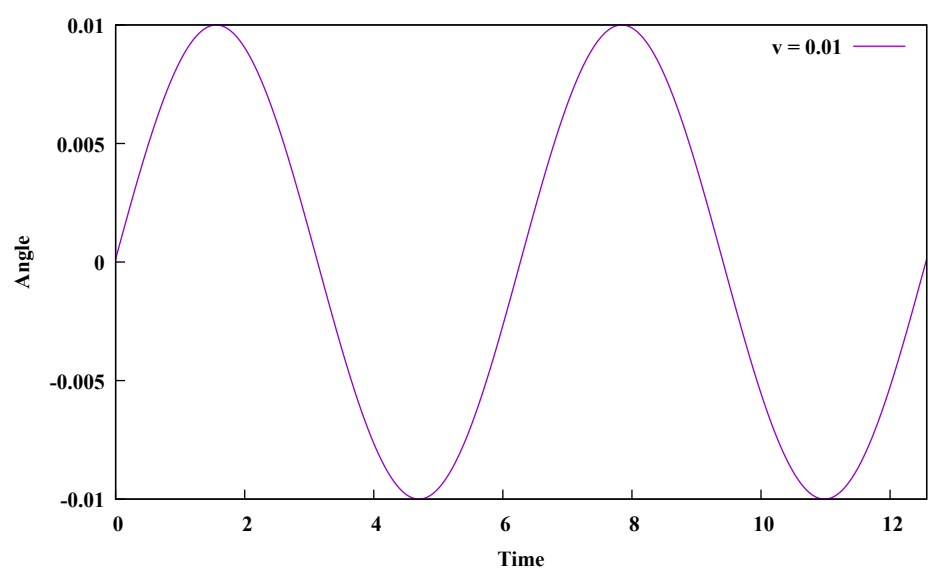


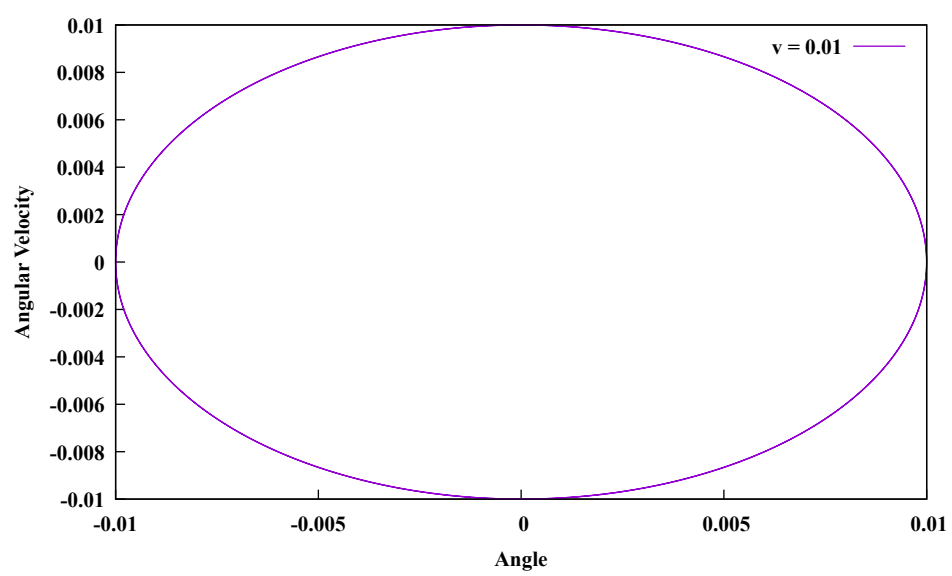
```

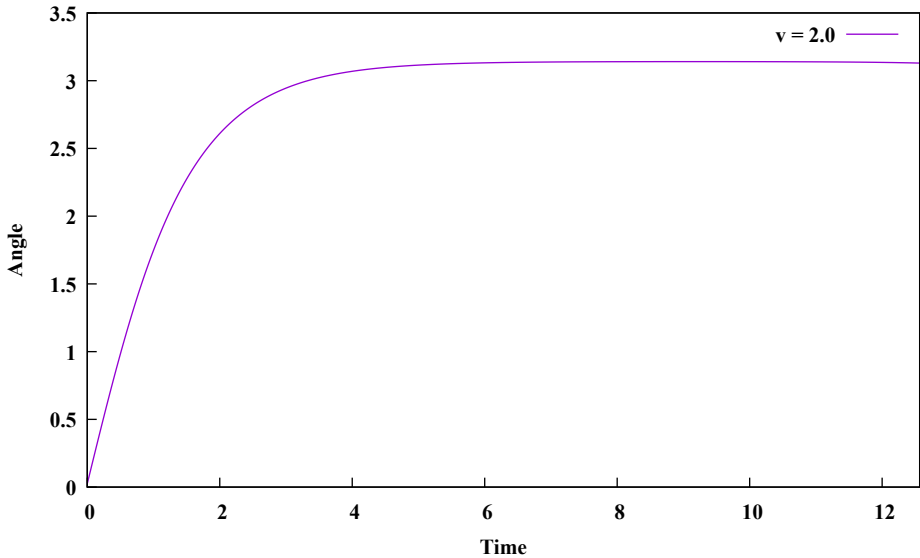
1  #include <iostream>
2  #include <fstream>
3  #include <array>
4  #include <vector>
5  #include <map>
6
7  //Constant expressions appearing in the problem
8  constexpr size_t dimension = 2;    //dimension of the reduced 1st-order    ↗
   problem
9  constexpr double PI = 3.14159265359;    //value of PI
10
11 //Definition of data types in the problem
12 typedef std::array<double, dimension> state_type; //data type definition    ↗
   for dependant variables - array of x_0, x_1, ... x_n
13 typedef std::map<double, state_type> solution;    //data type definition for ↗
   storing the list of calculated values ((hash)map of time -> state)
14
15 //This is the differential Equation, with the higher order derivatives
16 void Pendulum(const state_type& x, state_type& dxdt, state_type& d2xdt2,    ↗
   state_type& d3xdt3, state_type& d4xdt4){
17     //This is the differential Equation, reduced to first-order
18     dxdt[0] = x[1];
19     dxdt[1] = - sin(x[0]);
20
21     //Second derivatives
22     d2xdt2[0] = dxdt[1];
23     d2xdt2[1] = - cos(x[0]) * x[1];
24
25     //Third derivatives
26     d3xdt3[0] = d2xdt2[1];
27     d3xdt3[1] = sin(x[0]) * x[1] * x[1] + 0.5 * sin(2.0 * x[0]);
28
29     //Fourth derivatives
30     d4xdt4[0] = d3xdt3[1];
31     d4xdt4[1] = cos(x[0]) * x[1] * x[1] * x[1] - 2.0 * sin(x[0]) * sin(x    ↗
   [0]) * x[1] + cos(x[0]) * x[1];
32 }
33
34 //The stepper function, calculates x_{n+1} given the differential equation, ↗
   x_{n} and step size
35 void euler4_step(void (*Diff_Equation)(const state_type& x, state_type&    ↗
   dxdt, state_type& d2xdt2, state_type& d3xdt3, state_type& d4xdt4),    ↗
   state_type& x, const double dt){
36     state_type dxdt, d2xdt2, d3xdt3, d4xdt4;    //temporary variable for    ↗
   storing dx/dt, d2x/dt2, d3x/dt3 etc.
37     Diff_Equation(x, dxdt, d2xdt2, d3xdt3, d4xdt4); //calculate dx/dt, d2x/ ↗
   dt2, d3x/dt3, etc. from the differential equation
38     for (size_t i = 0; i < dimension; i++) {
39         x[i] = x[i] + dxdt[i] * dt + 0.5 * d2xdt2[i] * dt * dt + 1.0/6.0 *    ↗

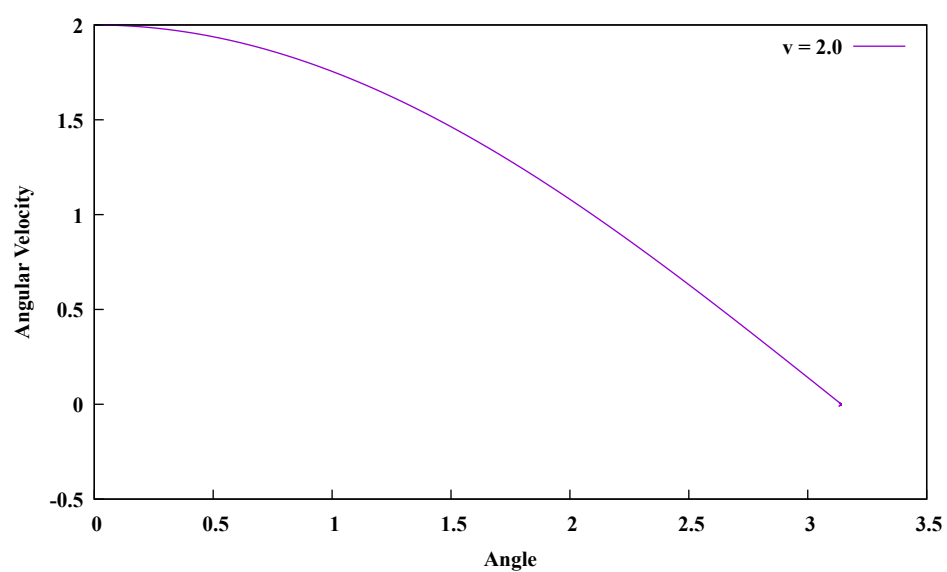
```

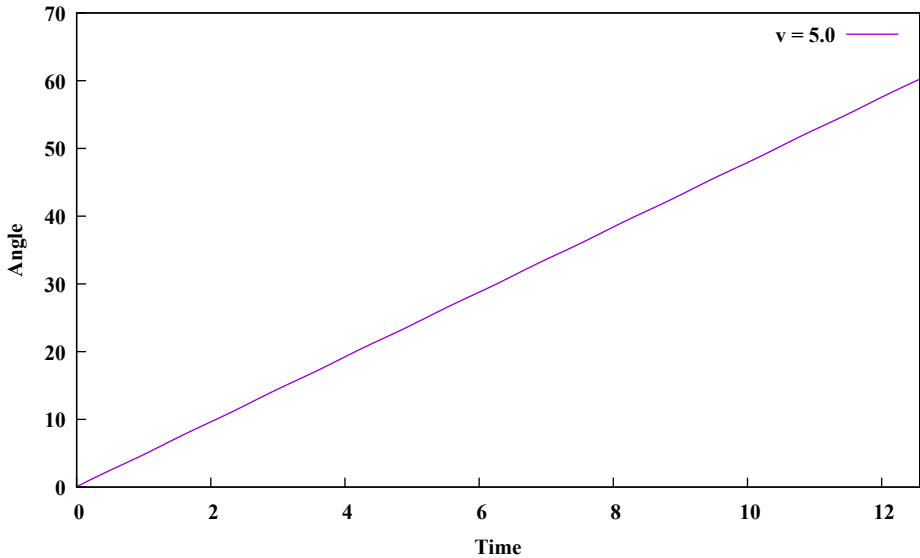
```
        d3xdt3[i] * dt * dt * dt + 1.0/24.0 * d4xdt4[i] * dt * dt * dt * dt; //Euler forward difference formula
40    }
41 }
42
43 int main(){
44     solution x_t;    //variable to store the calculations
45
46     size_t STEPS = 1000; //number of steps
47     double t_0 = 0.0;    //initial time
48     double t_1 = 4.0 * PI; //final time
49     double dt = (t_1 - t_0) / (STEPS - 1); //step size
50     state_type x = {0.0, 5.0}; //initial values for dependant variables
51
52     //Step through the domain of the problem and store the solutions
53     x_t[t_0] = x; //store initial values
54     for (size_t i = 0; i < STEPS; i++) {
55         euler4_step(Pendulum, x, dt); //step forward
56         x_t[t_0 + i * dt] = x; //store the calculation
57     }
58
59     std::ofstream outfile; //file handle to save the results in a file
60     outfile.open("5.0.txt", std::ios::out | std::ios::trunc );
61     for (auto const& temp : x_t){
62         outfile << temp.first << "\t" << temp.second[0] << "\t" <<
            temp.second[1] << std::endl;
63     }
64     outfile.close();
65 }
```

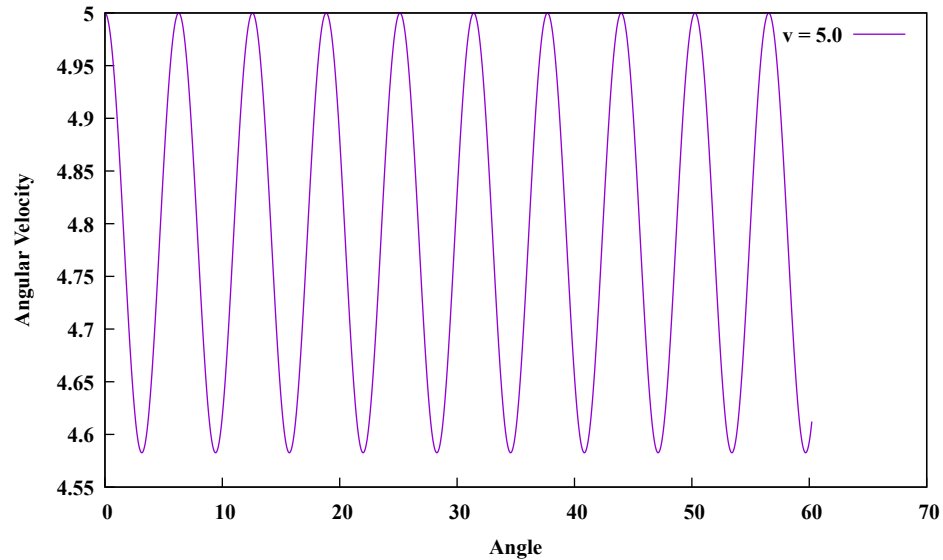












Comparing with the phase space contour plot, we might get a sense of precision.

