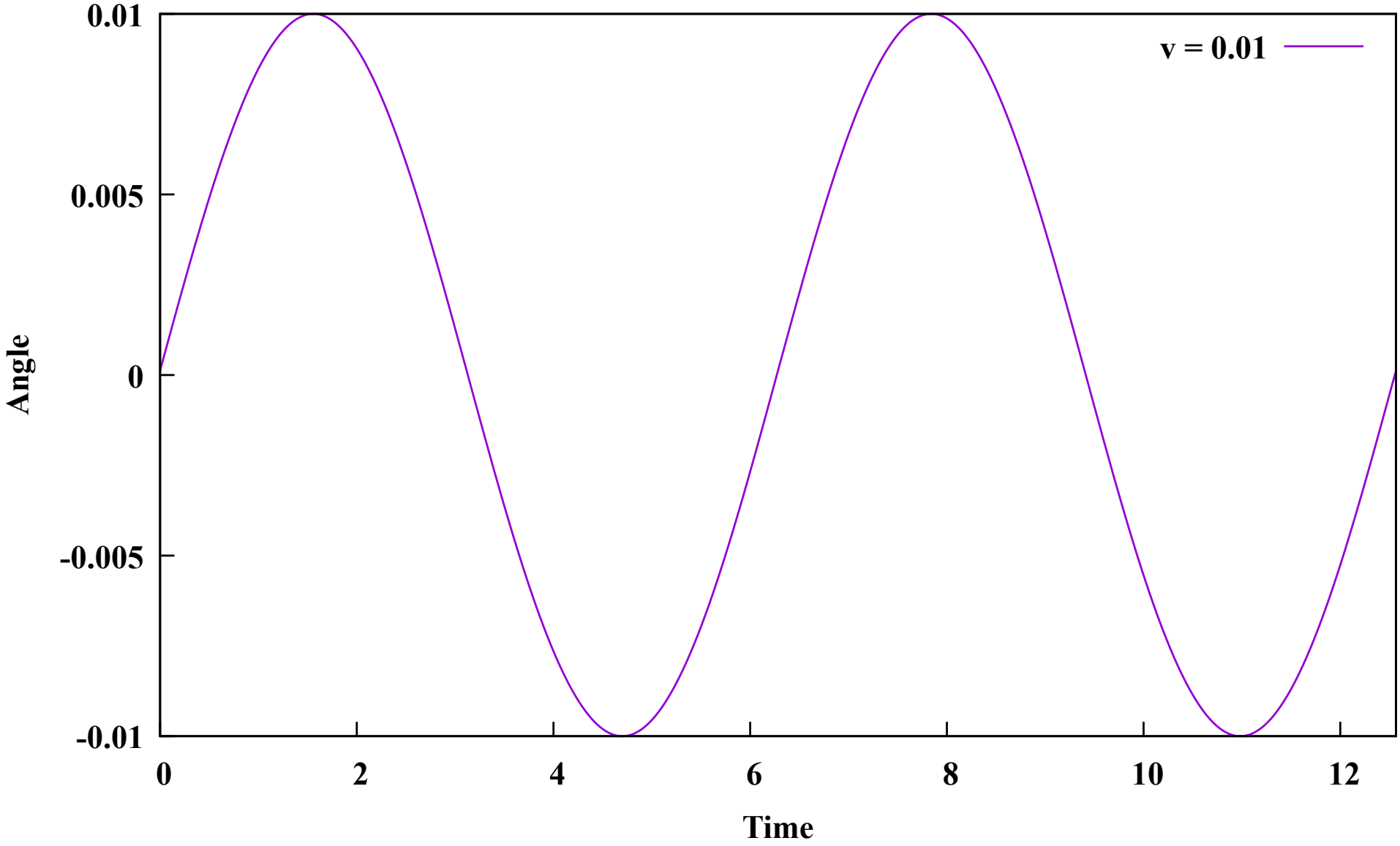
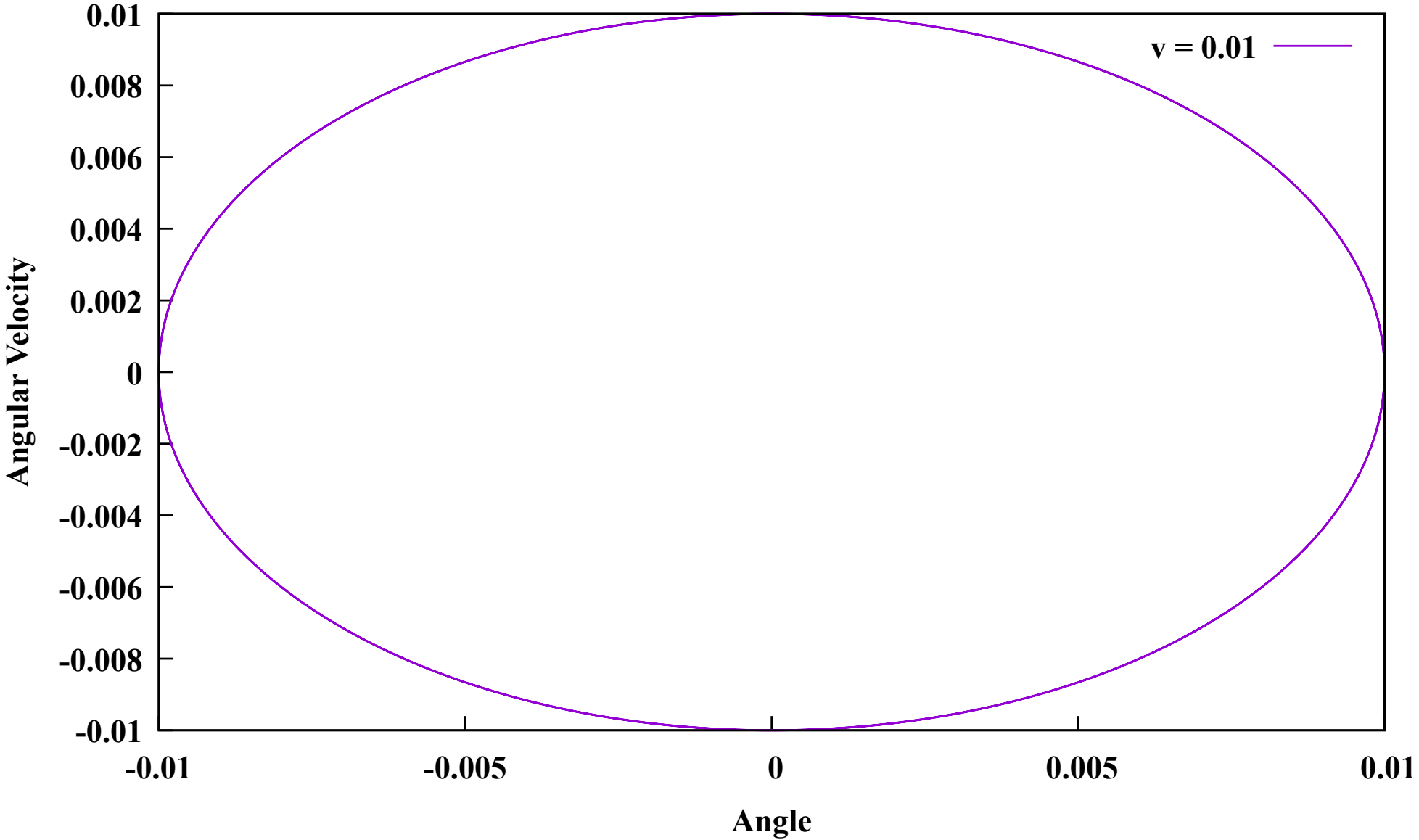
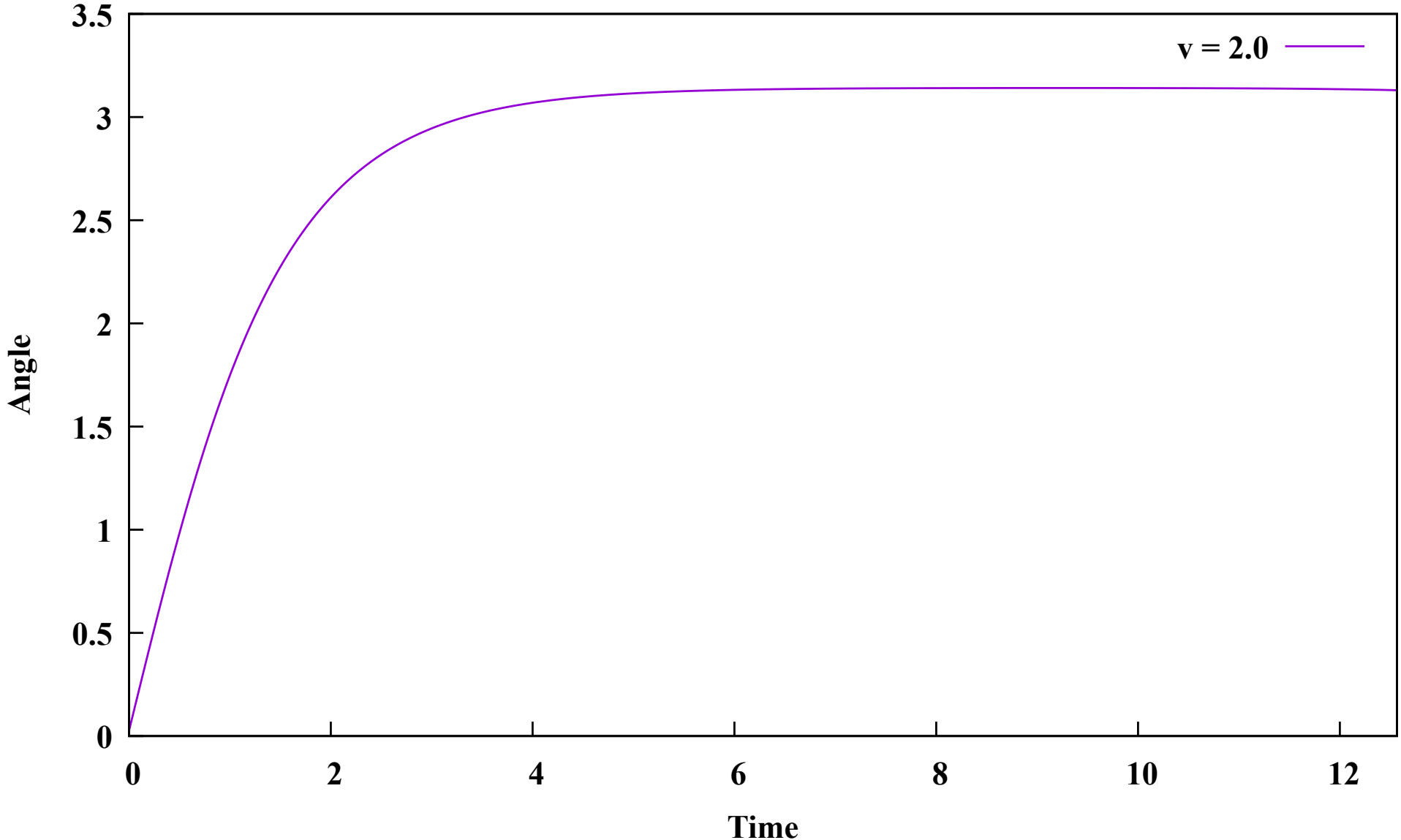
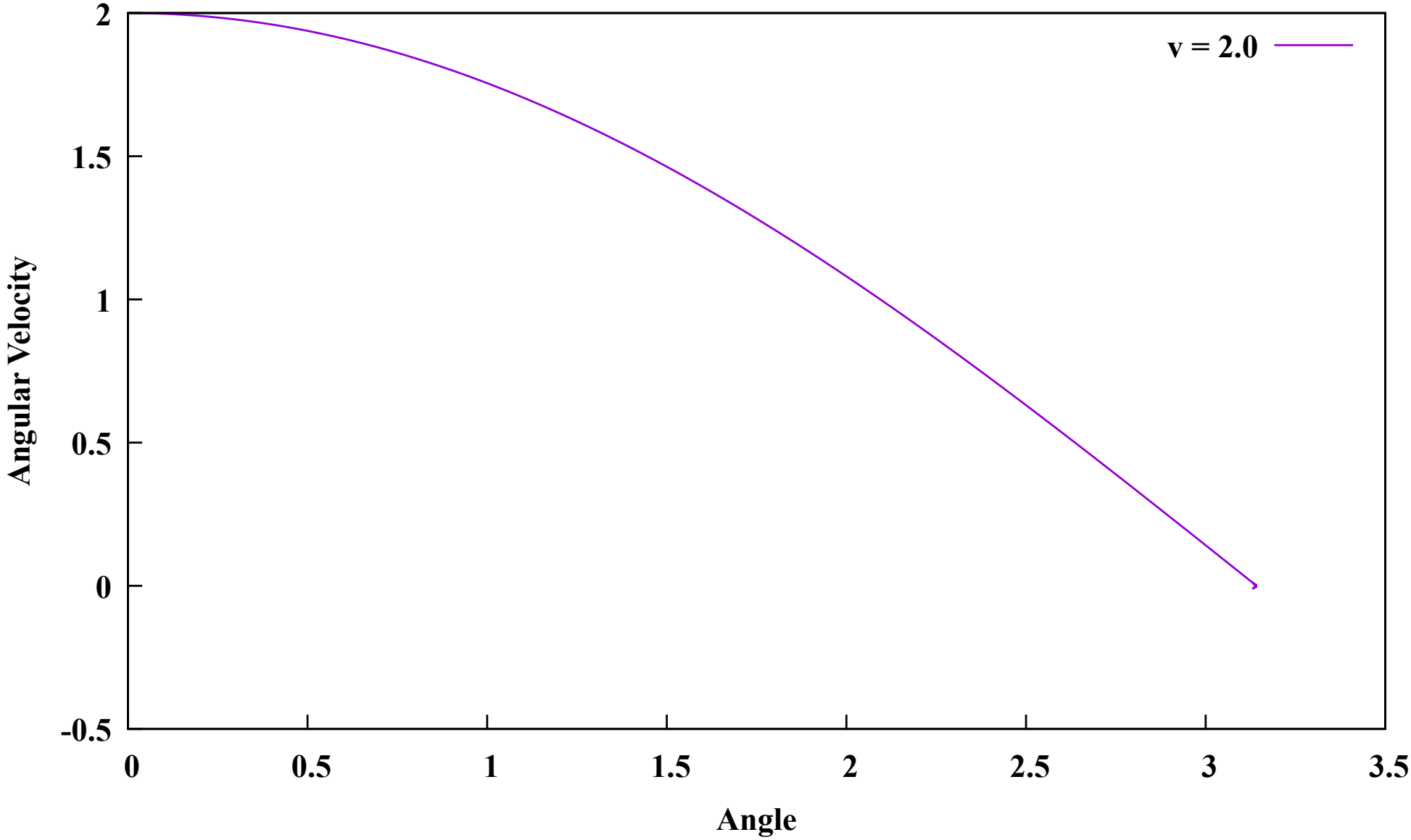
```cpp
#include <iostream>
#include <fstream>
#include <array>
#include <map>

//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)

//This is the differential Equation, with the higher order derivatives
void Pendulum(const state_type& x, state_type& dxdt, state_type& d2xdt2, state_type& d3xdt3, state_type& d4xdt4){
    //This is the differential Equation, reduced to first-order
    dxdt[0] = x[1];
    dxdt[1] = - sin(x[0]);

    //Second derivatives
    d2xdt2[0] = dxdt[1];
    d2xdt2[1] = - cos(x[0]) * x[1];

    //Third derivatives
    d3xdt3[0] = d2xdt2[1];
    d3xdt3[1] = sin(x[0]) * x[1] * x[1] + 0.5 * sin(2.0 * x[0]);

    //Fourth derivatives
    d4xdt4[0] = d3xdt3[1];
    d4xdt4[1] = cos(x[0]) * x[1] * x[1] * x[1] - 2.0 * sin(x[0]) * sin(x[0]) * x[1] + cos(2.0 * x[0]) * x[1];
}

//The stepper function, calculates x_{n+1} given the differential equation, x_{n} and step size
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){
    state_type dxdt, d2xdt2, d3xdt3, d4xdt4;    //temporary variable for storing dx/dt, d2x/dt2, d3x/dt3 etc.
    Diff_Equation(x, dxdt, d2xdt2, d3xdt3, d4xdt4); //calculate dx/dt, d2x/dt2, d3x/dt3, etc. from the differential equation
    for (size_t i = 0; i < dimension; i++) {
        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 *
```

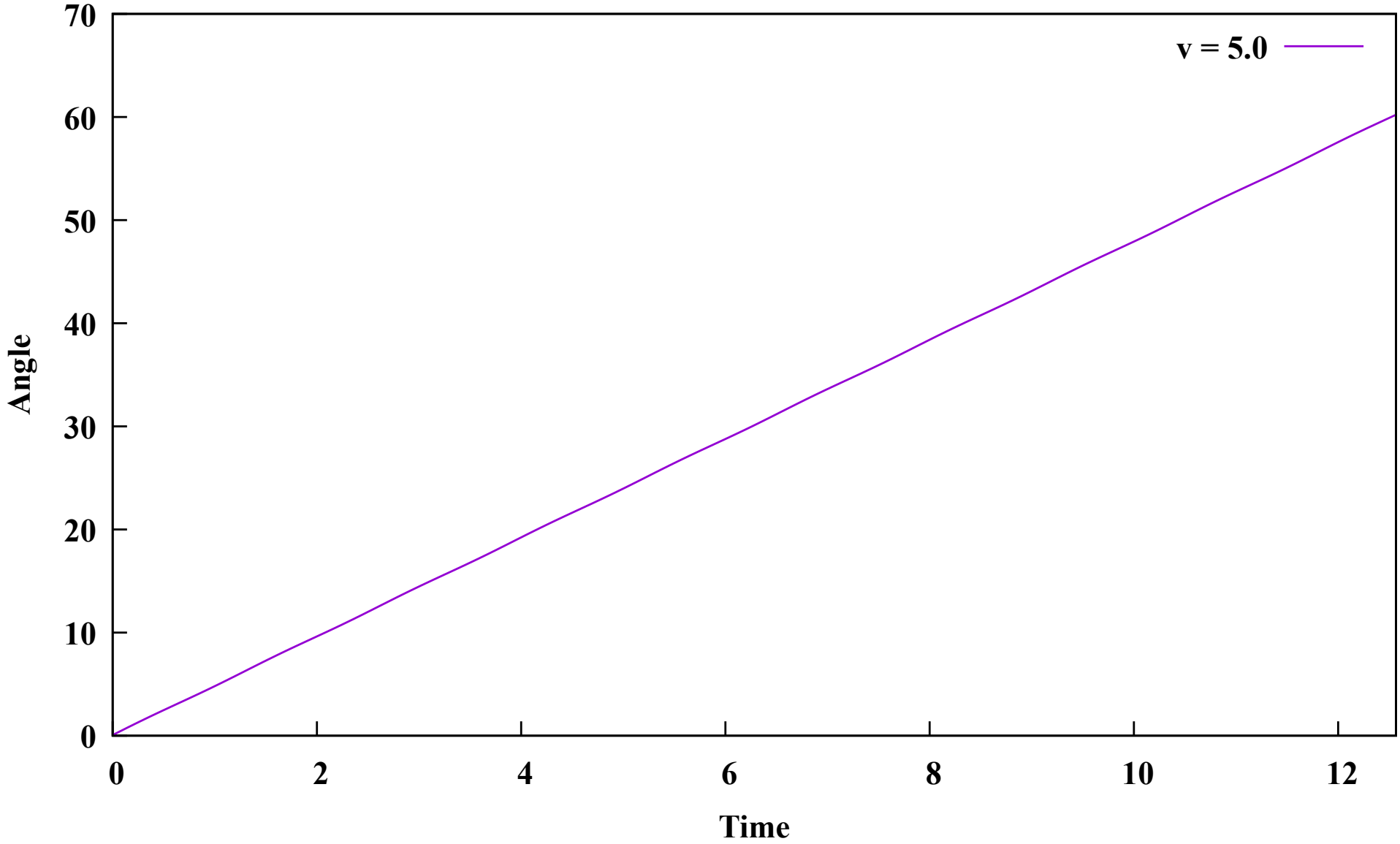
```
            dt; //Euler forward difference formula
    }
}

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 = 4.0 * PI;   //final time
    double dt = (t_1 - t_0) / (STEPS - 1); //step size
    state_type x = {0.0, 5.0};   //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++) {
        euler4_step(Pendulum, x, 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("5.0.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();
}
```

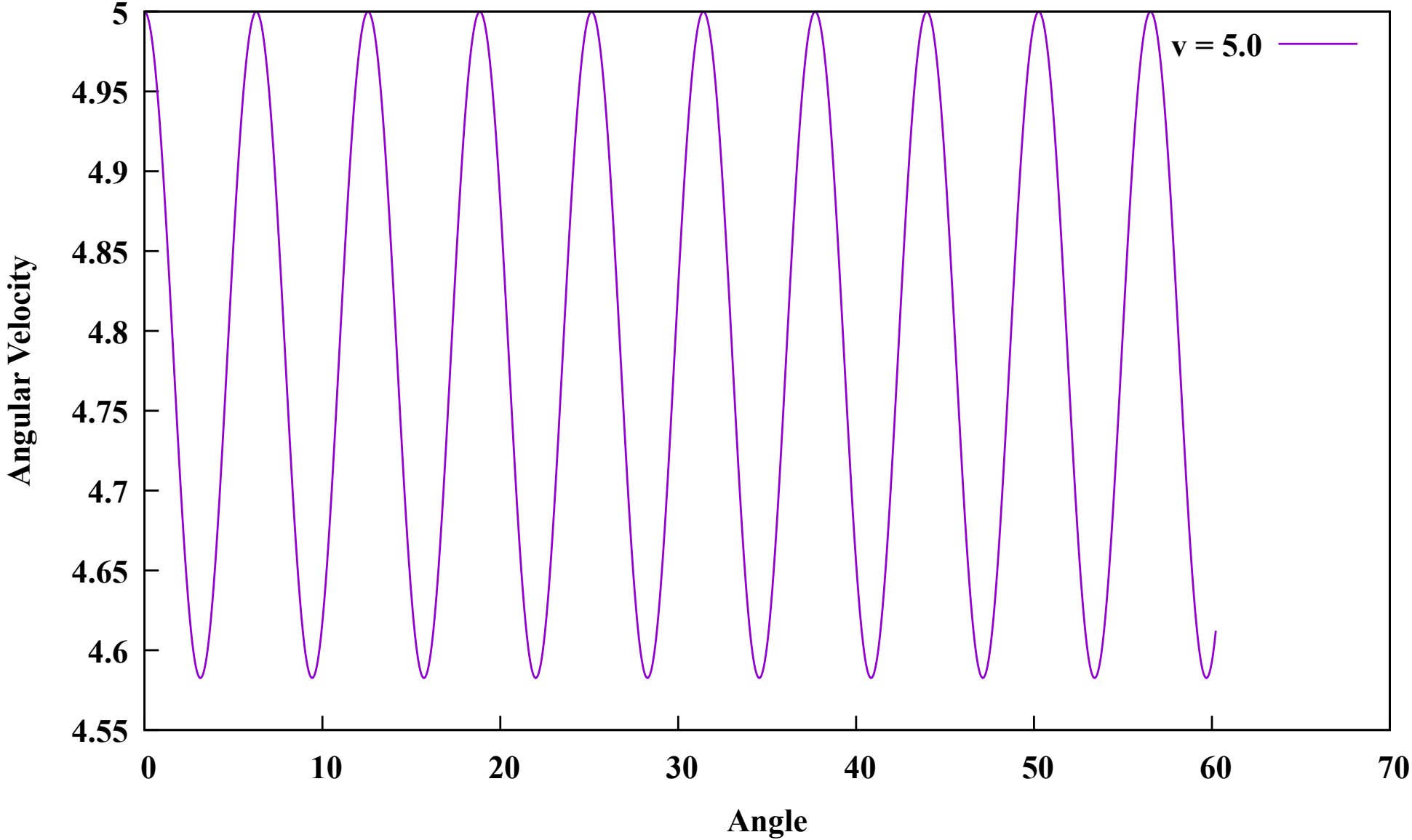

v = 0.01
0.01
0.005
0
-0.005
-0.01
Angle
0
2
4
6
8
10
12
Time

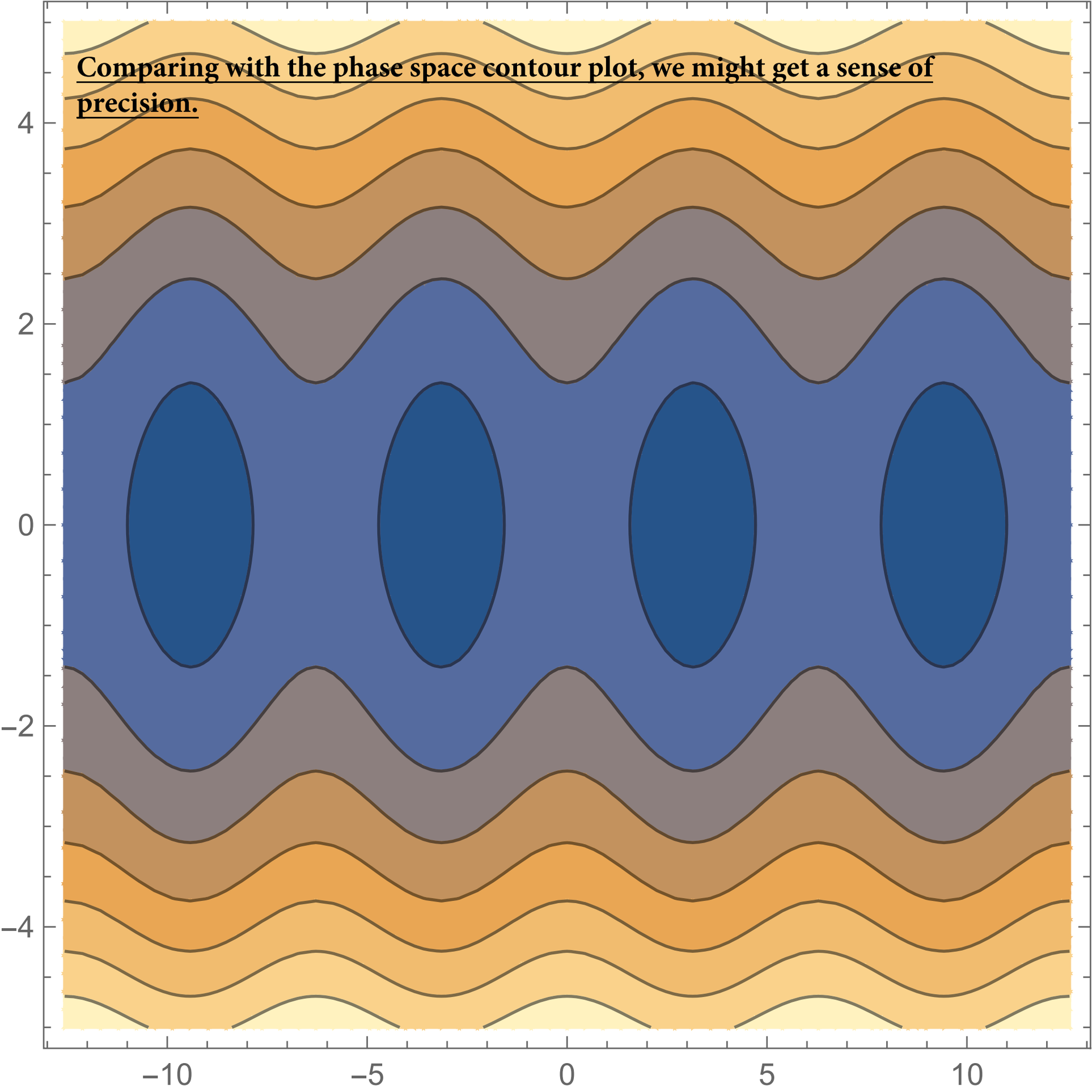

v = 0.01
0.01
0.008
0.006
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
-0.01
Angular Velocity
-0.01
-0.005
0
0.005
0.01
Angle


v = 2.0
Angle
Time
3.5
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12


v = 2.0
Angular Velocity
Angle
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5


v = 5.0
Angle
Time
0
10
20
30
40
50
60
70
0
2
4
6
8
10
12


v = 5.0
Angular Velocity
Angle
5
4.95
4.9
4.85
4.8
4.75
4.7
4.65
4.6
4.55
0
10
20
30
40
50
60
70


Comparing with the phase space contour plot, we might get a sense of precision.
4
2
0
−2
−4
−10
−5
0
5
10