HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Assignment Sheet Nr. 4

Paul Monderkamp, Matr.Nr. 2321677

monderkamp@thphy.uni-duesseldorf.de

Contents

1 Exercise 1

1.1 Code

```
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;
struct particle
        {
                double x,v,F;
        };

int main()
{
double maxtime = 50.0;

const int N = 100;
const int Nsteps = 1000;

double dt = maxtime/(Nsteps-1);
particle part[N];

for (int i=0;i < N;i++)
    {
        part[i].x = i;
        part[i].v = 0.5*(2.0*(double)rand()/RAND_MAX-1.0);
    }
double T[Nsteps];
double V[Nsteps];
double P[Nsteps];
double E[Nsteps];

for (int i=0;i < N-1;i++)
    {
        T[0]+= 0.5*part[i].v*part[i].v;
        V[0]+= (part[i].x-part[i+1].x)*(part[i].x-part[i+1].x);
        P[0]+=part[i].v;
    }
T[0]+= 0.5*part[N-1].v*part[N-1].v;
V[0]+= (part[N-1].x-N-part[0].x)*(part[N-1].x-N-part[0].x);
P[0]+=part[N-1].v;
E[0]=T[0]+V[0];

ofstream out("output2.txt");
```

```
out << 0 << "   " << T[0] << "  " << V[0]
    << "         " << P[0] << "  " << E[0] << endl;

for (int i=1;i < Nsteps;i++)
    {
        for (int j=1;j < N-1;j++)
            {
                part[j].F = part[j+1].x-2.0*part[j].x+part[j-1].x;
            }

        part[0].F = part[1].x-2.0*part[0].x + part[N-1].x-N;
        part[N-1].F = part[0].x*N+ part[N-2].x-2*part[N-1].x;

        for (int j=1;j < N-1;j++)
            {
                part[j].x = part[j].x + part[j].v *dt
                                      + 0.5*part[j].F*dt*dt;
                part[j].v = part[j].v + 0.5*dt*(part[j].F+part[j+1].x
                                      +part[j-1].x-2.0*part[j].x);
            }

        for (int k=0;k < N-1;k++)
            {
                T[i]+= 0.5*part[k].v*part[k].v;
                V[i]+= (part[k].x-part[k+1].x)*(part[k].x-part[k+1].x);
                P[i]+=part[k].v;
            }

        T[i]+= 0.5*part[N-1].v*part[N-1].v;
        V[i]+= (part[N-1].x-N-part[0].x)*(part[N-1].x-N-part[0].x);
        P[i]+=part[N-1].v;
        E[i]=T[i]+V[i];

        out << i*dt << "         " << T[i] << "  " << V[i]
            << "         " << P[i] << "  " << T[i]+V[i] << endl;
        }
out.close();
return 0;
}
```

1.2 results

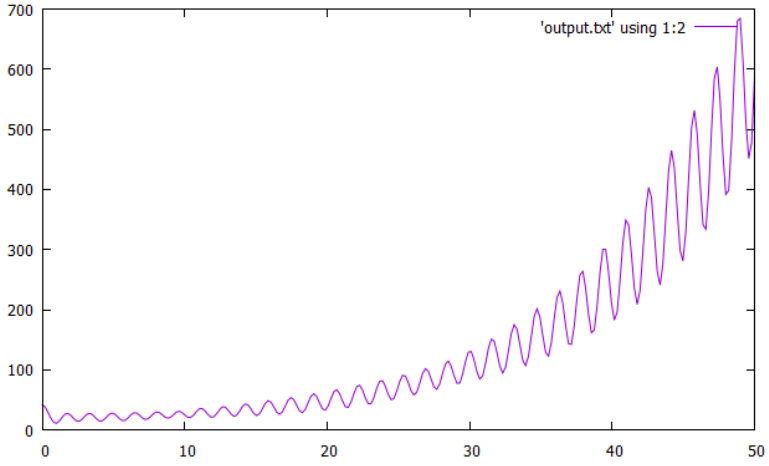

Figure 1.1: T(t)

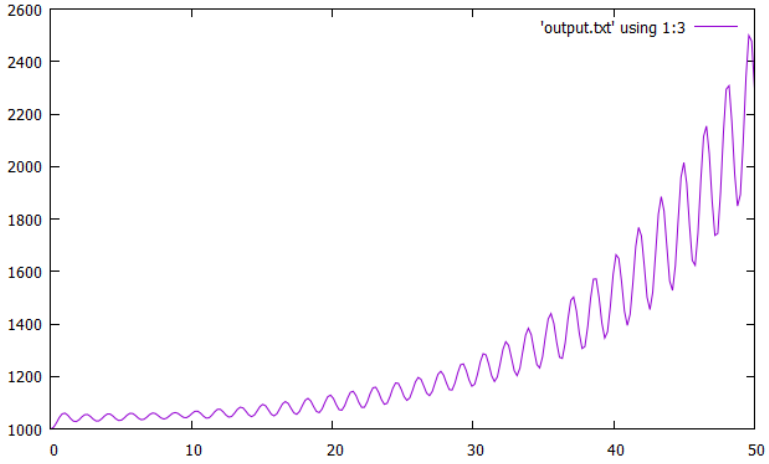

Figure 1.2: V(t)

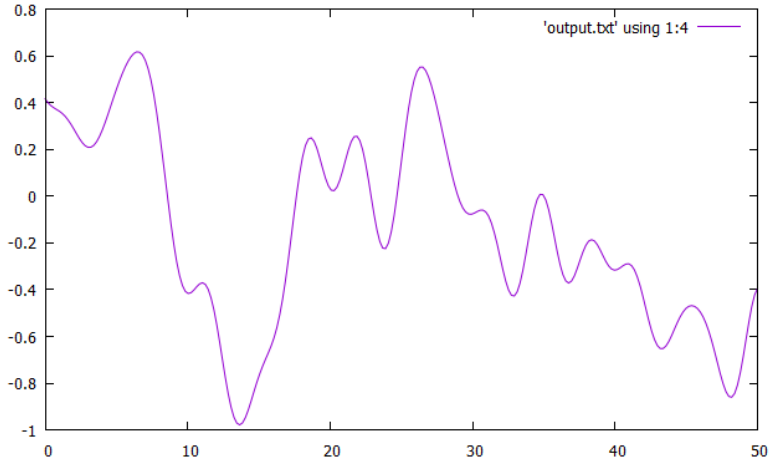

Figure 1.3: P(t)

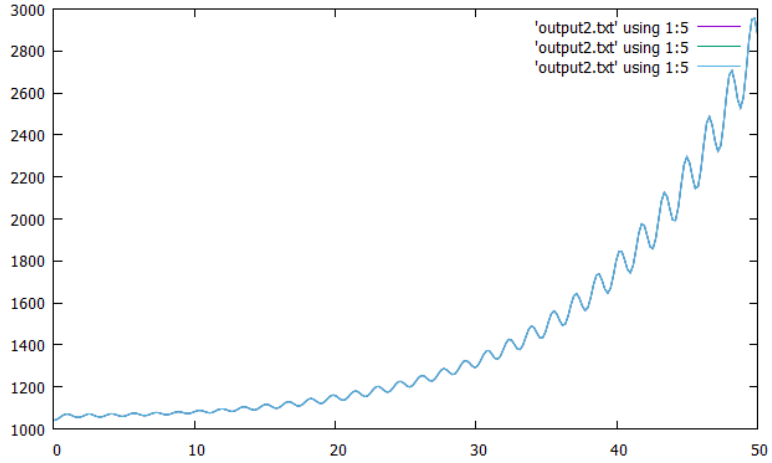

Figure 1.4: E(t)

Figure 1.1 shows the total kinetic energy as function of time,
Figure 1.2 shows the total potential energy as function of time and
Figure 1.3 shows the total momentum of the system.
It can be observed that while T and V oscillate, the momentum does not show oscillatory behaviour. It just fluctuates.

Figure 1.4 shows the total energy of the system that is computed as sum as of the kinetic and the potential energy. The three plots with the different time steps 0.2, 0.1, 0.05 lie accurately on top of each other due to the numerical precision. The global error for the energy is proportional to dt^2 and thus the energy diverges from what should be a constant.