HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Assignment Sheet Nr. 5

Paul Monderkamp, Matr.Nr. 2321677

monderkamp@thphy.uni-duesseldorf.de

Contents

1 Exercise 1

1.1 Code

```
#include <iostream>

#include <vector>

#include <cmath>
#include <fstream>
#include <string>
#include <cstdio>
#include <cstdlib>

using namespace std;
const double cutoff = pow(2.0,1.0/6.0);
const double V = pow(2.0,14.0);
#include "readin.cpp"
typedef vector<double> vouble;

class particle
  {
    public:
    particle()
      {
        x = 0.0;
        y = 0.0;
        vx = 0.0;
        vy = 0.0;
      }
    ~particle(){}

    void set_x(const double a){x = a;}
    void set_y(const double a){y = a;}
    void set_vx(const double a){vx = a;}
    void set_vy(const double a){vy = a;}

    double get_x(){return x;}
    double get_y(){return y;}
    double get_vx(){return vx;}
    double get_vy(){return vy;}

    private:
    double x,y,vx,vy;
  };
```

```
vouble f_pq(particle p, particle q) // force on particle q
                                        from particle p
  {
    vouble force(2);
    force[0] = 0.0;
    force[1] = 0.0;

    double dx = q.get_x() - p.get_x();
    double dy = q.get_y() - p.get_y();
    if (dx > 7.0) {dx -= 14.0;}
    if (dx < -7.0) {dx += 14.0;}
    if (dy > 7.0) {dy -= 14.0;}
    if (dy < -7.0) {dy += 14.0;}

    double dr = sqrt(dx*dx+dy*dy);
    if (dr <= cutoff)
      {
        force[0] += (dx/dr)*(48.0*pow(dr,-13.0)-24.0*pow(dr,-7.0));
        force[1] += (dy/dr)*(48.0*pow(dr,-13.0)-24.0*pow(dr,-7.0));
      }

    return force;
  }

vouble f_i(particle *p,int i, int N) // force on particle
                                        with index i
  {
    vouble force(2);
    force[0] = 0.0;
    force[1] = 0.0;

    for (int j=0; j<N;j++)
      {
        if (j != i)
          {
            double dx = p[i].get_x() - p[j].get_x();
            double dy = p[i].get_y() - p[j].get_y();
            if (dx > 7.0) {dx -= 14.0;}
            if (dx < -7.0) {dx += 14.0;}
            if (dy > 7.0) {dy -= 14.0;}
            if (dy < -7.0) {dy += 14.0;}

            double dr = sqrt(dx*dx+dy*dy);
            if (dr <= cutoff)
              {
                force[0] += (dx/dr)*(48.0*pow(dr,-13.0)
                                                        -24.0*pow(dr, -7.0));
                    force[1] += (dy/dr)*(48.0*pow(dr,-13.0)
                                                        -24.0*pow(dr, -7.0));
                  }
              }
          }
      return force;
    }
  double V_pot(particle *p, int N)
    {
      double U = 0.0;
          for (int j=0; j<N;j++)
            {
              for (int i=j+1;i<N;i++)
                {
                  double dx = p[i].get_x() - p[j].get_x();
                  double dy = p[i].get_y() - p[j].get_y();
                  if (dx > 7.0) {dx -= 14.0;}
                  if (dx < -7.0) {dx += 14.0;}
                  if (dy > 7.0) {dy -= 14.0;}
                  if (dy < -7.0) {dy += 14.0;}

                  double dr = sqrt(dx*dx+dy*dy);
                  //cout << dr << endl;

                  if (dr <= cutoff)
                    {
                      U += 4.0*(pow(dr,-12.0)-pow(dr,-6.0));
                    }
                }
            }
      return U;
    }
  double T_kin(particle *p, int N)
    {
      double T = 0.0;
      for (int i =0;i<N;i++)
        {
          T += 0.5*(p[i].get_vx()*p[i].get_vx()+
                    p[i].get_vy()*p[i].get_vy());
        }
      return T;
    }

  double P(particle *p, int N)
    {
```

```
    double P = 0.0;

    for (int i =0;i<N; i++)
      {
        for (int j=0;j<N; j++)
          {
            if (i != j)
              {
                vouble force_ij = f_pq(p[i],p[j]);
                P += (p[i].get_x()-p[j].get_x())*force_ij[0]
                     + (p[i].get_y()-p[j].get_y())*force_ij[1];
              }
          }
      }

    P += (2.0/2.0*V)*T_kin(p,N);
    return P;
  }


int main()
  {
    vouble pos_x = get_column("posdat2.txt",1,3);
    vouble pos_y = get_column("posdat2.txt",2,3);
    vouble vel_x = get_column("veldat2.txt",1,3);
    vouble vel_y = get_column("veldat2.txt",2,3);

    const int N = pos_x.size();
    const double tmax = 10.0;
    double dt = 0.0005;
    const int Nsteps = (int)tmax/dt + 1;
    cout << Nsteps << endl;
    particle *p = new particle[N];
    for (int i=0;i<N; i++)
      {
        p[i].set_x(pos_x[i]);
        p[i].set_y(pos_y[i]);
        p[i].set_vx(vel_x[i]);
        p[i].set_vy(vel_y[i]);
      }
    //ofstream out("U_dt=" + to_string((int)1e4*dt) + "e-4.txt");
    ofstream out("termodyn_dt=" + to_string(dt) + ".txt");
    for (int k=0;k<Nsteps;k++)
      {
        for (int i=0;i<N; i++)
          {
            vouble f = f_i(p,i,N);
            p[i].set_x(p[i].get_x() + dt*p[i].get_vx()
                + 0.5 * f[0] * dt*dt);
            p[i].set_y(p[i].get_y() + dt*p[i].get_vy()
                + 0.5 * f[1] * dt*dt);

            if (p[i].get_x() > 14.0) {p[i].set_x(p[i].get_x()-14.0);}
            if (p[i].get_x() < 0.0) {p[i].set_x(p[i].get_x()+14.0);}
            if (p[i].get_y() > 14.0) {p[i].set_y(p[i].get_y()-14.0);}
            if (p[i].get_y() < 0.0) {p[i].set_y(p[i].get_y()+14.0);}

            //cout << p[i].get_x() << " " << p[i].get_y() << endl;


            p[i].set_vx(p[i].get_vx()+0.5*dt*f[0]);
            p[i].set_vy(p[i].get_vy()+0.5*dt*f[1]);

            f = f_i(p,i,N);

            p[i].set_vx(p[i].get_vx()+0.5*dt*f[0]);
            p[i].set_vy(p[i].get_vy()+0.5*dt*f[1]);
          }
        out << k*dt << "␣␣" << 2.0*T_kin(p,N)/(3.0*N)
        << "␣␣" << P(p,N) << "␣␣"<< V_pot(p,N) << "␣␣"
        << T_kin(p,N) << "␣␣" << T_kin(p,N)+V_pot(p,N) << "␣␣" << endl;
        cout << k*dt << "␣␣" << 2.0*T_kin(p,N)/(3.0*N)
        << "␣␣" << P(p,N) << "␣␣"<< V_pot(p,N) << "␣␣"
        << T_kin(p,N) << "␣␣" << T_kin(p,N)+V_pot(p,N) << "␣␣" << endl;

      }

   out.close();
   delete[] p;
   //getchar();

   return 0;
}
```

1.2 results

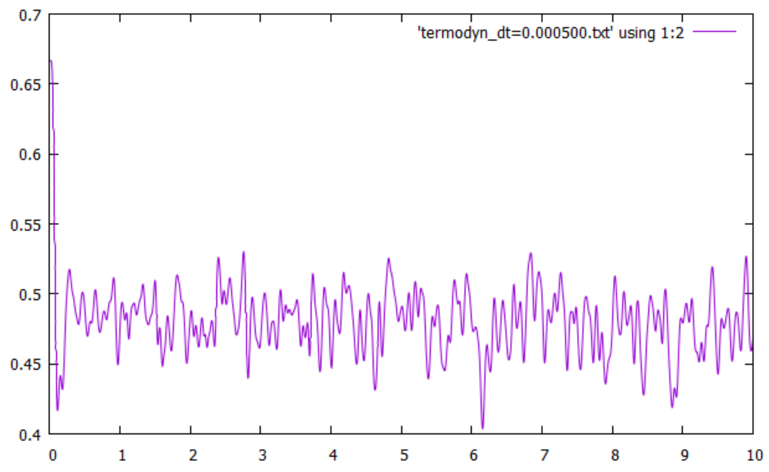

Figure 1.1: $T(t)$

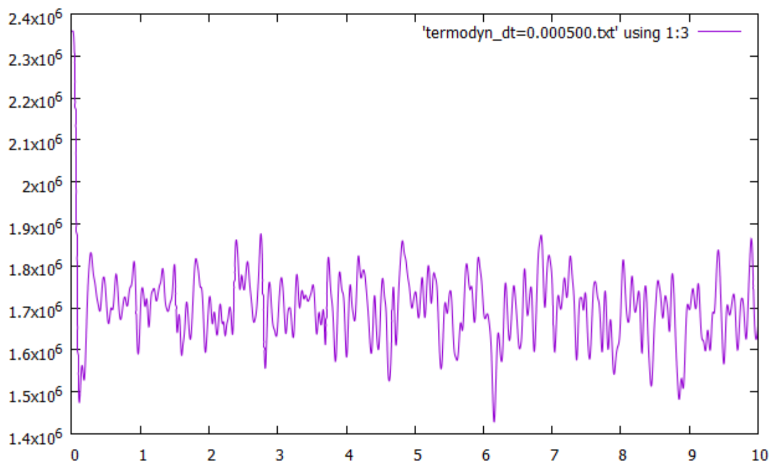

Figure 1.2: $P(t)$

Figure 1.1 and 1.2 show the instantaneous temparature and the pressure in the system as a function of time. The temparature is proportional to the kinetic energy with a factor of $\frac{1}{Nk_B}$. Where k_B is set to one for this plot.

The pressure is calculated over the virial route with the formula

$$P = \frac{Nk_BT}{V} + \frac{1}{6V}\left[\sum_{i=1}^{N}\sum_{j\neq i}^{N} r_{ij}\cdot f_{ij}\right] \tag{1.1}$$

and thus is highly dependent on the temparature.

Figure 1.3 shows the potential energy, the kinetic energy and the total energy in the system as a function of time.

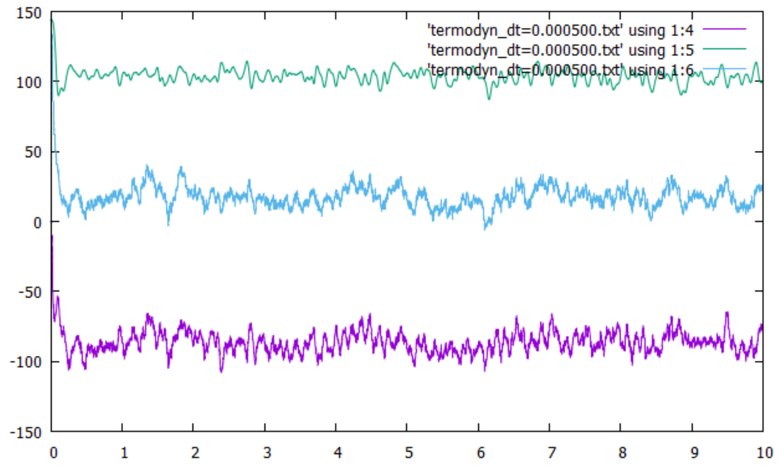

Figure 1.3: $V(t)$ (purple), T_{kin}(t) (turquoise), and $E(t)$ (light blue)