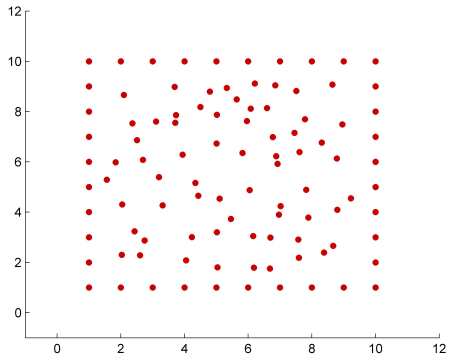

Simulations of the motional and thermodynamic behavior of a simple solid state model

Computational physics project WS12/13

Carsten Bauer

January 31, 2013

Contents

1 Introduction

This report is about the project part of the computational physics lecture by Daniel Cocks at the Goethe-University, Frankfurt. The task was to model a solid state in MATLAB and to investigate its motional and thermodynamic behavior depending on specific properties (e.g. particle count, dimension, interaction type) and constraints (e.g fixed or none-fixed boundaries). Different time-integration methods were used to achieve the solutions to the equations of motion whereby the limits of the different solving approaches became visible.

The following chapters will provide an overview of the MATLAB implementation and some selected simulation results. As a matter of course the presented is not even close to what the implemented model can accomplish and what has been noted during the last month.

1.1 Model

In our model particles, e.g. atoms, are point-like structures interacting with each other (mostly next-neighbor interaction). The type of interaction is being modeled through different two-particle-potentials, e.g a classical spring potential. Simulating the time-evolution of this very basic system leads to macroscopical thermodynamic quantities, as for example temperature, which has no microscopical definition. After starting with a one-dimesional chain, we extended the model to 2D. To get a first impression, the following illustration shows our 20x20 particle model in some random state.

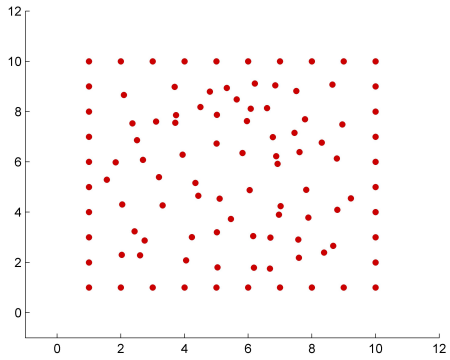

Figure 1: 2D model of 20x20 particles in some random state

2 MATLAB Implementation

2.1 Overview

In our implementation there are MATLAB files of two kinds: functions and scripts.
Functions have some specific purpose as e.g. the calculation of the energy of the system, whereas scripts are simply the sequence of some commands and function callings. Only the latter can be executed directly and will produce some real result (e.g. plots). There are three scripts, more or less based on the predefined extensions in the project description, which are

1. *base.m*: The basic model with focus on motion and energy conservation
2. *boundaries.m*: Similar to the above but with attention on boundaries like e.g. a moving piston or an oscillating wall
3. *thermalization.m*: Focus on thermalization and thermodynamic equilibrium

All scripts are following the same process design, which is illustrated in figure 2. Moreover 2D model files are indicated by a '2D' in the naming and possess basicly the same format.

In every case the parameters of the simulation can be adjusted at the very top. The following solvers, potentials and boundaries are available

- Solvers: Euler-Richardson, Euler-Richardson-Adaptive, Ode45
- Potentials: Spring (1D,2D), Coulomb (1D,2D), Combined (2D), Lennard-Jones (1D)
- Boundaries: Fixed (1D,2D), Moving Piston (1D), Oscillating Wall (1D)

Besides the solvers and potentials, which are of course functions, there are

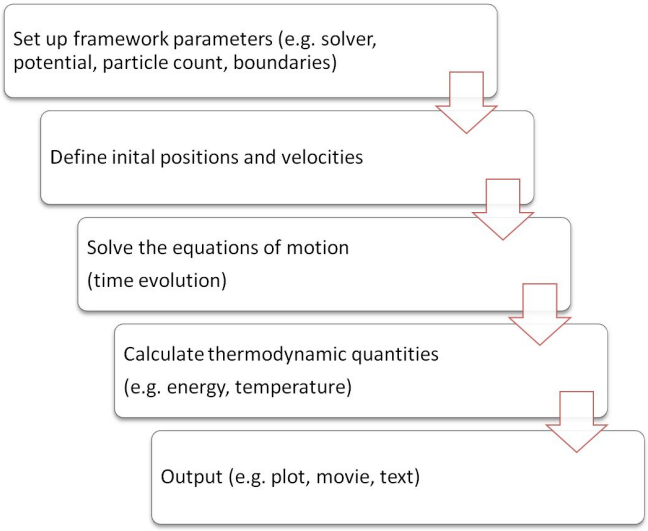

Figure 2: Illustration of the simulation process

1. *thermos.m*: Calculates the kinetic, potential and total energy as well as pressure based on a system timestamp and the systems parameters.

2. *temperature.m*: Measures the temperature of the system by assuming thermodynamic equilibrium (Boltzmann-Distribution) and fits the kinetic energy per particle distribution. To avoid unwanted dependancy on the discretization (number of 'bins') we are really fitting the cumulative distribution function. Hint: this function returns a temperature always, even if its not physical as in a none-equilibrium situation. For further information see section Thermalization below. Furthermore this function handles also the 2D case.

3. *thermalizedVelDist.m*: This file contains an approach of a function which consumes a temperature value as a parameter and should create a thermalized velocity distribution for a 1D system based on it. Oddly the temperature of such a prepared system varies about around approximately 5% at short timescales (100s) whereas the long time behavior is completely wrong (temperature increases a lot). Unfortunately there was neither time to advance this function nor to analyse the strange behavior it generates.

Last but not least there are *analysis*.m* files. These contain (often botchy) scripts based on the ones listed above to perform some analysis on the system. Some of the plots in this report were generated by those files.

2.2 Comments

- The **design of the code** was chosen in such a way, that it is comfortable to use for analysis and especially easy to extend. Its few effort to change parameters and to integrate new solvers and new interaction types. At least for some not to fancy configurations some nice looking plots and movies are being generated also (see Simulation Results).

- Considering simulation speed the **implementation of the derivative functions** is noteworthy. The solving time dramatically increases when for loops are used to set the changes in velocity (dv) of the particles. Instead most of the here presented derivatives are taking use of the *diff* function, which, after some long lasting contemplation (especially in 2D), delivers an vectorized version. Figure 3 shows a comparison in one dimension (2D is even more important, since the particle number scales as N_{1D}^2).

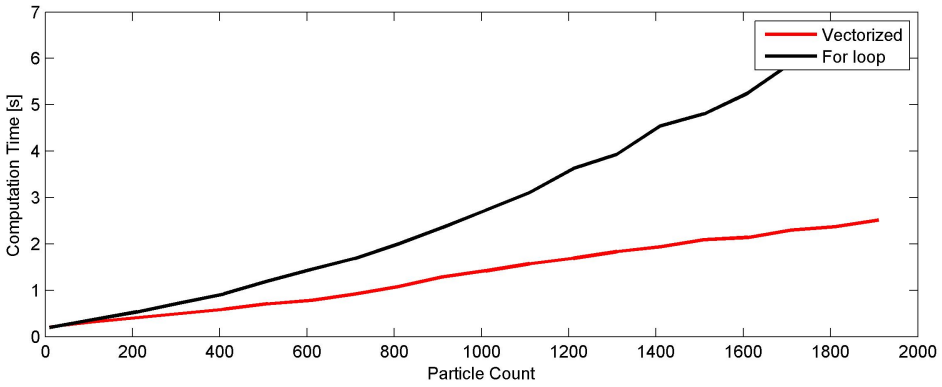

Figure 3: Comparison of vertorized and for-loop based implementation of the derivative according to the harmonic spring potential in 1D. The calculation has been done by the ode45 solver.

- Another mentionable detail is the **storage of the system information in the 2D** case. Usually one has to handle four matrices $(x_{pos}, y_{pos}, x_{vel}, y_{vel})$ with a total cellcount of N^4. To improve speed and particularly manageability we use complex numbers to hold the information. This leaves us with two matrices, one for positions, one for velocites, whereby the complex number in each cell holds the x component in the real and the y component in the imaginary part.

- In case of using the 2D harmonic spring potential one has the option to enable **second-neighbor-interaction**, which means, that all nearest and second-nearest neighbors (total eight) are taken into account for interaction.

- **Lennard-Jones-Potential**: This is an realistic potential which models the Van-der-Waal dipolar attraction at long distance and a hard repulsion due to the Pauli principle at short distance in the form

$$V_{LJ} = \frac{B}{r^{12}} - \frac{A}{r^6}$$

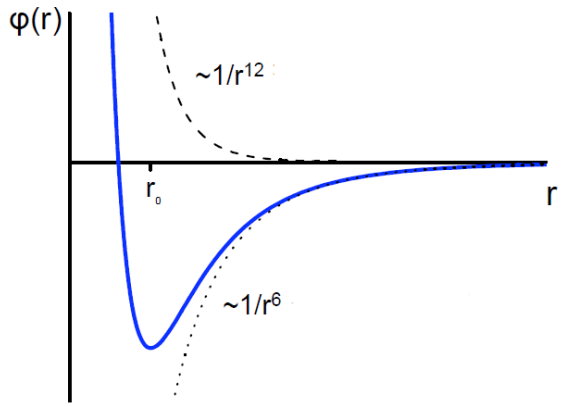

Figure 4: Lennard-Jones-Potential [2]

3 Simulation Results

3.1 Basic

3.1.1 Potentials and Solvers

First of all you should get an impression of the motion of the 1D system. Therefore watch the movie *motion1D.mpg*. We want to make some basic analysis of the model, the solvers and the potentials in one dimension. We will start with the harmonic spring potential. The motivation for using such a potential is given by the first-order taylor expansion of the real interaction potential in solids, which is quadratic. Being realistic to some degree the spring potential still has some weaknesses. As it allows particles to be at the same place, they can fly over each other and especially out of the borders of the model. As we are considering only next neighbor interactions this behavior is strongly problematic. Therefore one should adjust the system settings to keep the simulation reasonable. In contrast the coulomb and Lennard-Jones potential prevent this kind of motion (in 1D), since these potentials diverge for vanishing particle spacing (see figure 5).

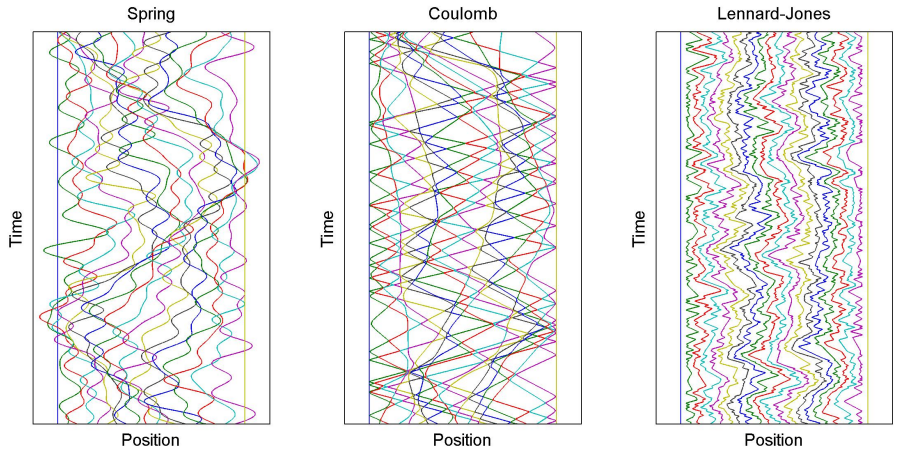

Figure 5: Motion of one system with different interaction potentials

Only due to numerical inaccuracies one can get particles with coulomb interaction to escape over the system borders. This can be achieved by increasing the initial velocities. The reason is that a particle can then be set over another particle by the time integrator during a too large timestep (relativ to the velocity).
We want to quantify the velocity dependent breaking of the simulation (ins sense of particles leaving the model) for the spring and coulomb potential and for each solver. For that we simulate both potentials several times, increasing the maximal initial velocity amplitude in each run by 0.1, watching for runaways in each solvers solution. This will provide upper bounds to the initial velocities. For better reliability we repeat this measurement five times and take averages. Table 6 shows the results.

	Spring potential	Coulomb potential
ODE45	1.3	higher than 5
Euler Richardson	1.3	3.5
Euler Richardson Adaptive	1.26	higher than 5

Figure 6: Critical initial velocity amplitudes (mean values) for which particles slip out of the system boundaries. ($a = 1, k = 1, dt = 0.01, RelTol = 0.001, t_{final} = 10$)

In two dimensions the behavior is quite different. Since the systems geometrical constraints are not continuous any more - there is free space between the border particles - not even but especially the coulomb potential makes particles slip away, as the force acting upon them is approaching zero for large distances. The movies *motion2D.mpg* and *slipping2D.mpg* show the 'normal' 2D motion and the slipping effect. The harmonic spring potential - like in 1D - allows particles to leave the system, too, but it at least keeps them near since

$$\vec{F}_{ij}^{spr} \to \infty \text{ for } r_{ij}$$

To get a maybe more realistic 2D potential one can introduce a combined potential

$$V_{comb} = V_{spring} + V_{coulomb}$$

on which the 'normal' motion above based on. To be stable the constant k_{spr} should be greater than k_{coul} because of the mentioned reason.

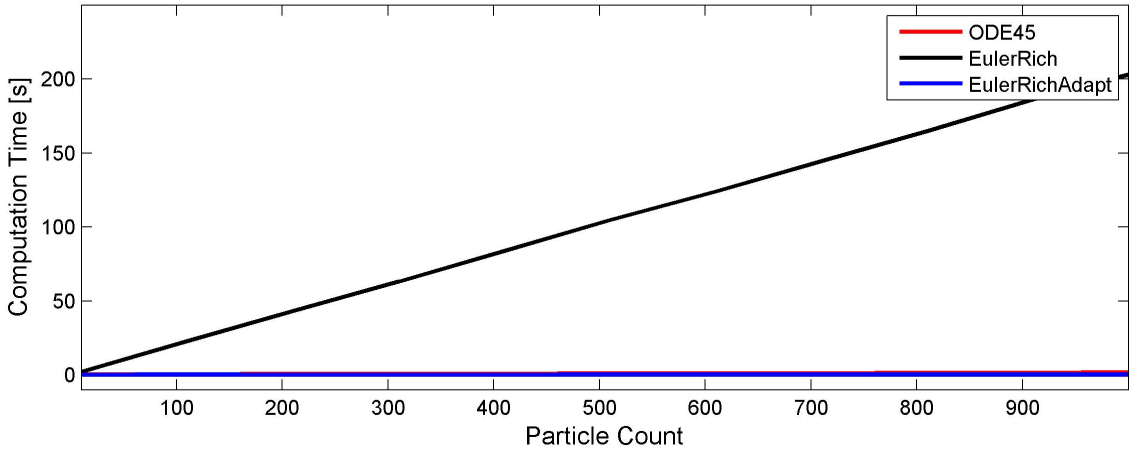

Figure 7: Computation time comparison of different solvers as a function of N (1D, $dt = 0.01$)

One essential part of the simulation is the solver. In this work there are *ode45*, *Euler-Richardson* and *Euler-Richardson-Adaptive*. Especially the simple Euler-Richardson distinguishs from the adaptive solvers in case of computation time. Figure 7 compares the simulation effort of the three solvers as a function of the total particle count N in 1D. As the 2D simulation is in principle similiar to 1D with N^2 particles (neglecting potential influences of the more complex derivatives) the character of this plot still holds for 2D (y-ticks will increase). Therefore particularly for 2D systems and 1D systems considering thermalization (large N) an adaptive solver should be used. But even this choice has problems e.g. regarding energy conservation, which we will analyse in the next section.

3.1.2 Energies and Pressure

From the gathered simulation results one can calculate the thermodynamic properties: Energy E, kinetic Energy T_k, potential Energy V and Pressure P. Depending on the solver choice and the simulation parameters the accuracy of the quantities can be great or vary unphysically. Lets take a look at energy conservation: At reasonable potential and solver settings for the harmonic spring potential one gets nice results in 1D and 2D (figure 8).

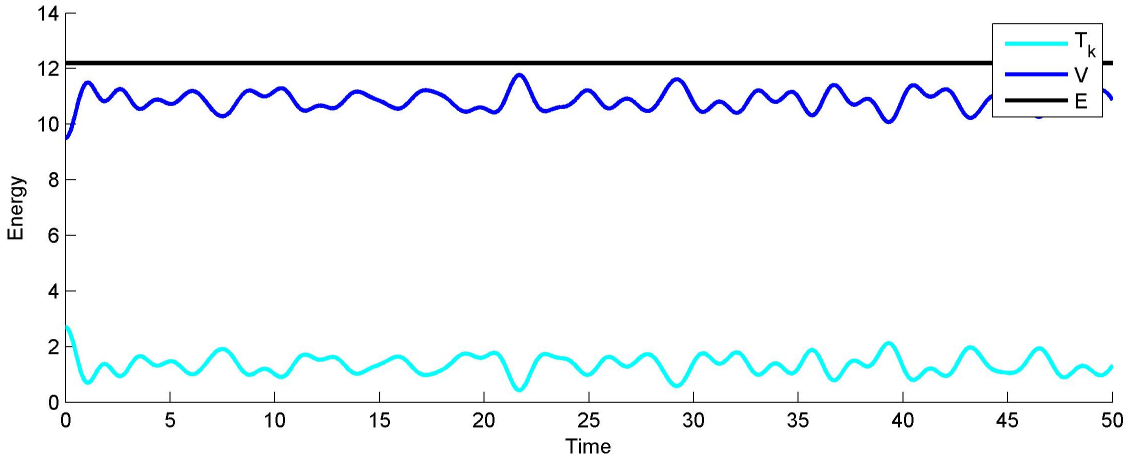

Figure 8: 1D Energy conservation for $N = 20$, $dt = 0.01$ and $k = 1$ (the same for all solvers)

In this case the total energy is conserved to at least $\approx 0.001\%$ accuracy for all solvers. Even for long-time runs ($t_{final} = 1000s$) the energy increase is less than 0.01%, which is perfectly acceptable. What parameter changes do disturb or completely ruin energy conservation? As our numerical solver accumulates little errors in each timestep violations of energy conservation should be visible in long time runs even if the other parameters are moderate. We can show this effect in shorter time by reducing the timestep of the Euler-Richardson to $dt = 0.1$. As displayed in figure 9 one sees a total increasment in energy of $\approx 27.4\%$ (spring potential).
Moreover at this stepsize one can nicely see the growing deviation of the Euler Richardson solution from the ode45 solution (figure 10).

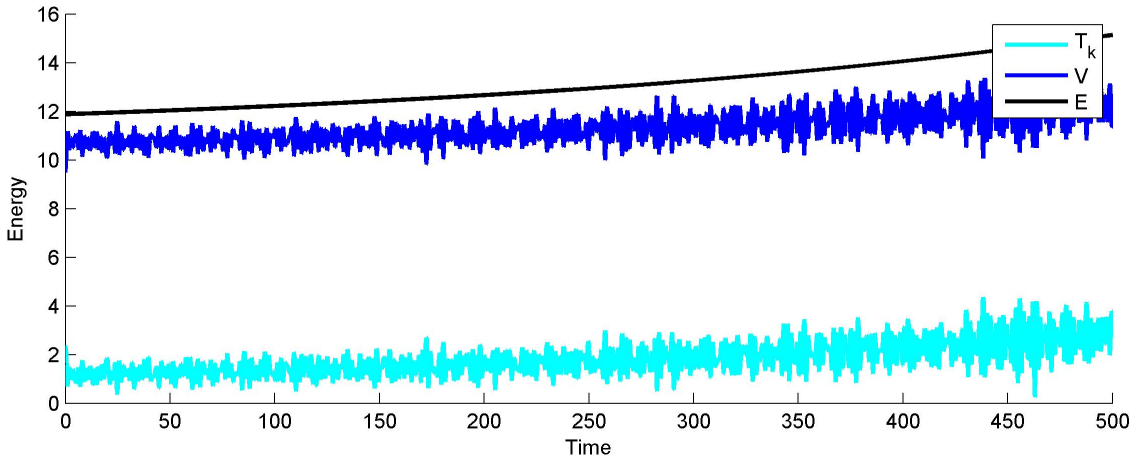

Figure 9: Disturbtion of energy conservation with Euler-Richardson ($N = 20$, $dt = 0.1$ and $k = 1$)

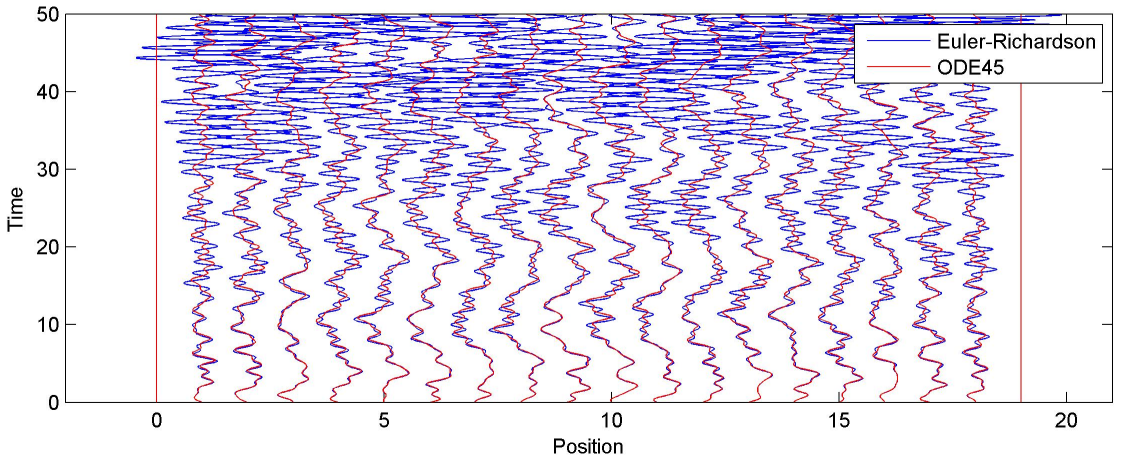

Figure 10: Deviation of the ode45 and Euler Richardson solution ($N = 20$, $dt = 0.1$ and $k = 7$)

The change of timestep doesn't affect the adaptive solvers, since they adjust their stepsize during integration. Anyway by increasing the simulation time and spring constant one will observe a collapse of energy conservation also for the Euler-Richardson Adaptive ($k = 1000$, $t_{final} = 100$, $RelTol = 10^{-2}$). Thereby the simulation duration seems to be more important, as for $k = 10$, $t_{final} = 1000$ one sees the breakdown, too. Figure 11 illustrates this behaviour. In contrast to the instruction sheet [1] (1a) we see no energy explosion for an increased relative tolerance $RelTol = 10^{-4}$. Instead energy is conserved to $\approx 0.03\%$ accuracy. Moreover the ode45 adaptive solver did conserve energy for the just mentioned cases.

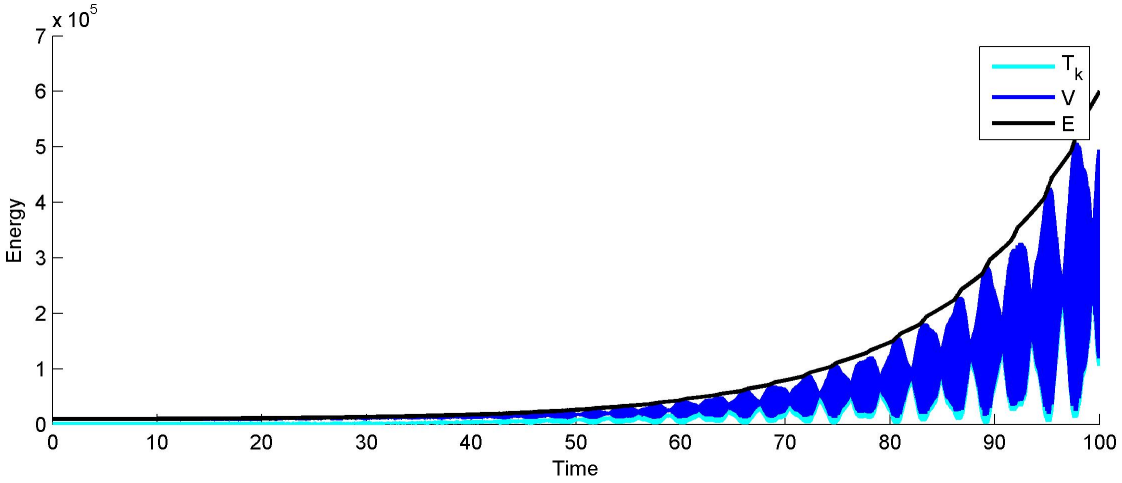

Figure 11: Ruin of energy conservation with Euler-Richardson Adaptive for high spring constant ($N = 20$, $k = 1000$, $t_{final} = 100$, $RelTol = 10^{-2}$)

In case of the inverse coulomb potential the effect is even worse. If particles leave the model due to numerical errors (see above), they gain a lot of kinetic energy which results in a step-function like increasing of the energy to the order of $14 \cdot 10^5$.

Last in this chapter we will take a look at pressure. In this report pressure in 1D for the harmonic spring potential is defined as

$$P_{left} = \frac{d}{dx} V_{21}^{spr} + a = -k\,(x_2 - x_1) + a$$

which corresponds to the magnitude of the force on the left border particle (x_1) due to the one to its right (x_2). The constant a represents the initial spacing of the particles and ensures that pressure is zero in the initial setup (equally spaced particles). The definition implicates that the pressure is negative when the particle x_2 pulls on the border and positive when it presses against it. Figure 12 shows an typical pressure trend for our 1D system at moderate settings (green) compared to a smiliar initialized 2D model (brown).
As we initialized with random velocities, the mean pressure is near to zero. Furthermore the 2D pressure is almost constant, which is also sensible.

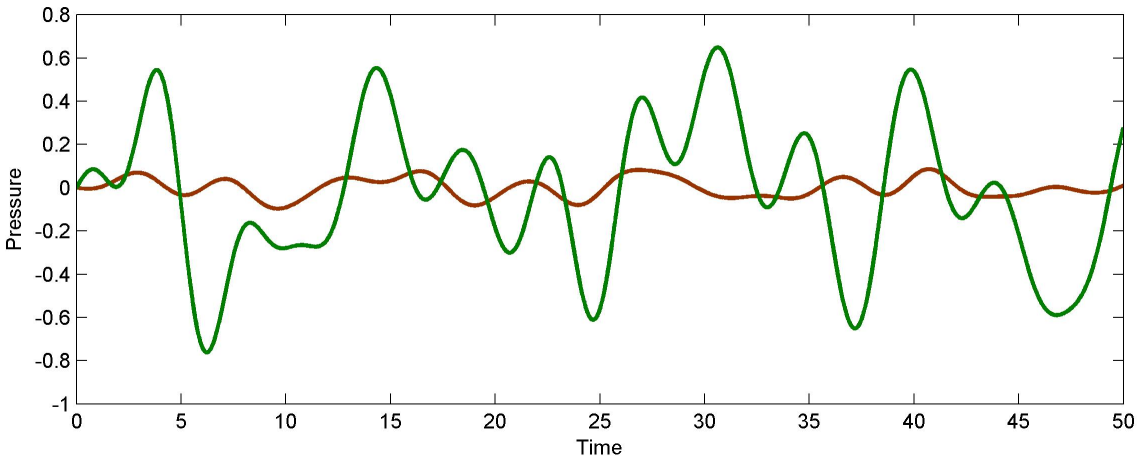

Figure 12: Pressure trends in 1D (green) and 2D (brown)

3.2 Thermalization

In this section we will inspect temperature and thermalization. Initially we want to describe our thermostat algorithm (*temperature.m*) which is used for measuring the systems temperature. Lets assume we know the kinetic energies per particle ϵ_i at a timestamp t. First we discretize the continuous ϵ_i-distribution $n(\epsilon_i)$ by using *hist* for a specific number of bins *nbins*, which leaves us with $n(\epsilon_{bin})$. From that we calculate the cumulative distribution function (CDF)

$$N(\epsilon_{bin}) = N^{-1} \sum_{\{\text{bins with } \epsilon \leq \epsilon_{bin}\}} n(\epsilon) \tag{1}$$

where N is the total particle count. Since we know that

$$n(\epsilon_{bin}) = C \cdot e^{-\lambda \epsilon_{bin}} \tag{2}$$

with $\lambda = T^{-1}$ holds in thermodynamic equilibrium we also have from the integral definition of $N(\epsilon_{bin})$

$$N(\epsilon_{bin}) = 1 - e^{-\lambda \epsilon_{bin}} \tag{3}$$

By fitting (1) against (3) we can determine λ which equals the inverse temperature.
The fitting can be done by using *fminsearch* which minimizes the summed up, squared difference of (1) and (3) by using *boltzmann.m* as criterion. Going this way one does not only gain the temperature $T = 1/\lambda$ but also a measure for its quality (In further plots the quality axes have been manipulated by a constant to have some quality measure in the region of $[0, 10]$). To get better statistics we also always average over a timespan of 2 seconds.
One could ask, why we aren't fitting our experimental $n(\epsilon_{bin})$ directly against (2). The reason is that this fit would strongly depend on the chosen *nbins*; by increasing *nbins* the distribution will

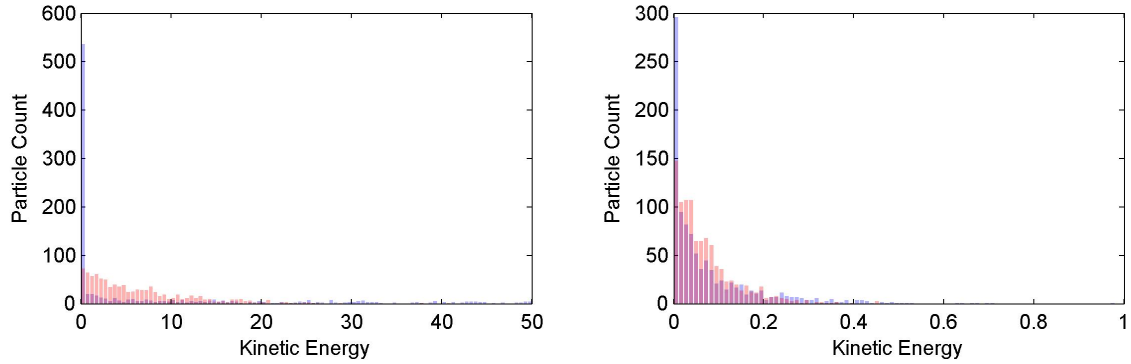

Figure 13: Histogram fits as part of temperature calculation. The blue bars indicate the systems state, whereas the red ones are theoretic based on the measured temperature value. In contrast to the left system, the right is in approximate equilibrium.

sink down, resulting in an different temperature. By considering the CDF we avoid this problem. Figure 13 shows some exemplary histogram fits using the described method.

We now want to 'oak' our quality scale. For this we consider our model initialized with some moderate random velocities. We expect the system to quickly thermalize, which should result in an almost constant temperature and temperature quality plot (after some very short time). Figure 14 shows the simulated results (ode45, spring potential), which greatly accords to our expectations (except the fact that the quality curve comes from low values!).

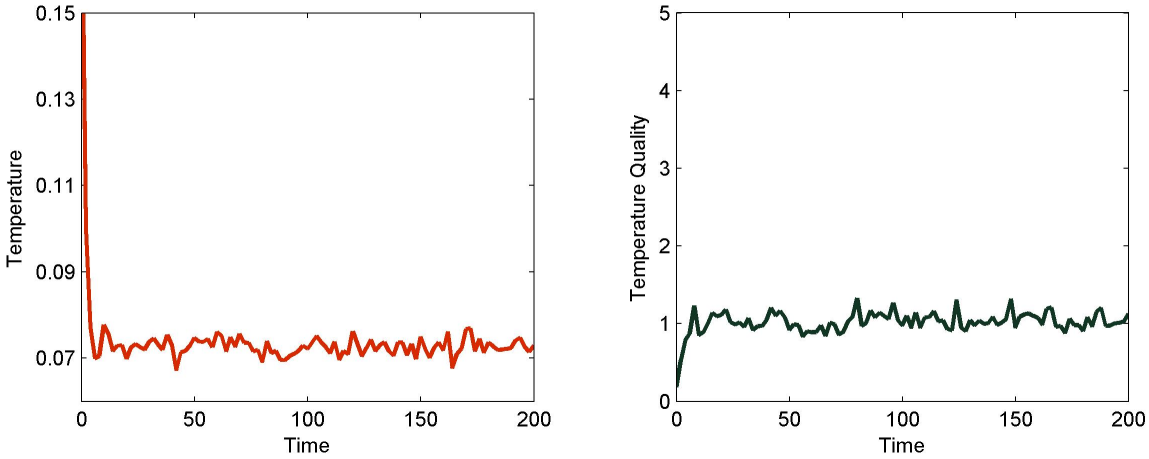

Figure 14: Thermalization of a system with random initial velocities ($N = 1000$, $k = 1$, $dt = 0.1$)

From the right plot (and not mentioned measurements) we see that stable quality values of around one seem to indicate a good fitting and therfore thermodynamic equilibrium.
Lets perform a long time simulation ($t_{final} = 1000$) of a system with random velocities only in the left half and still standing particles in the right (figure 15).

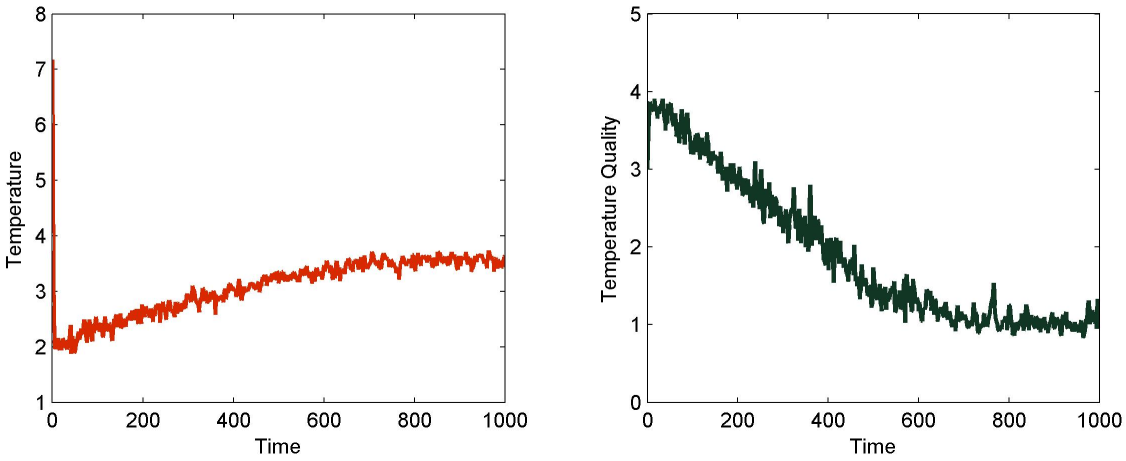

Figure 15: Thermalization of a system with random velocities in the left half and still standing particles in the right ($N = 1000$, $k = 1$, $dt = 0.1$)

Again, up to one's expectations the system starts in a none-equilibrium situation and reaches temperature stability after approximately 700 seconds.

If we (for both simulations) calculate the mean kinetic energy per particle $\overline{\epsilon_i}$ at the systems thermodynamic equilibrium (averaging over equilibrium time and afterwards over all particles) and try to see a relation between $\overline{\epsilon_i}$ and temperature we almost perfectly get

$$\overline{\epsilon_i} \approx T$$

which misses a factor of $1/2$ in comparison with the relation given on the instructions sheet [1].

All of the so far done considerations concerning temperature have been done in 1D, but it works in 2D as well (see *thermalization2D.m*. To keep this report from degenerating we omit to present the qualitatively equivalent results.

(Hint: We should mention, that we do **not** define potential energy to be zero in the beginning state to avoid confusion. Thus in case of the harmonic interaction the potential and kinetic energy do not both contain $\frac{1}{2}k_BT$ in thermal equilibrium (equipartition theorem))

3.3 Boundaries

In the previous sections we initialized the system and analysed its intrinsic motional and thermodynamic behavior. In the following we change the extent of the system (piston) or apply some external stimulation (oscillating left wall) during the time evolution and present a few of the gathered results. All of the simulations in this section have been done by using ode45.

3.3.1 Piston

We consider our 1D model consisting of $N = 1000$ particles equally distributed over a total length $L = (N-1)a = 999$, but this time we set some piston to reduce the size from L to $L' = L - \Delta L$ constantly over a time τ. The movie *1DpistonINV.mpg* shows the motional behaviour of this setup for the inverse potential with less particles ($N = 20$, $\Delta L = 30$,$L = 60$,$\tau = 25$). We now want to analyse the energy and temperature trend for this process in two regions: a slowly and fast moving piston. To investigate temperature changes we shoud assure that our model is in equilibrium initially. Therfore we always quickly simulate a random velocity system for 200 seconds for which we now from the analysis in the previous section, that it is nicely thermalized after that timespan. To stay comparable to the above results we choose the harmonic spring potential for our simulation.

First the slow piston. Let's take a look at the results (figure 16)

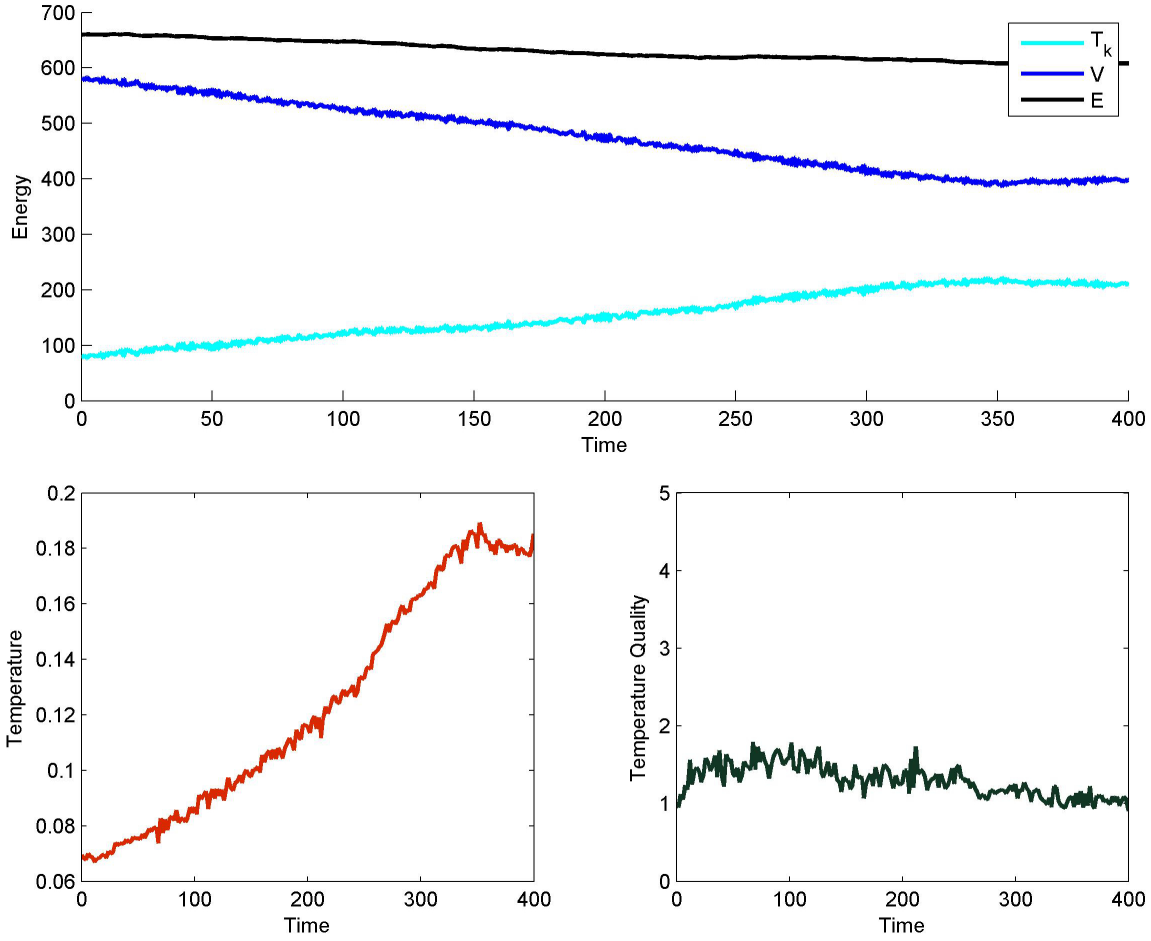

Figure 16: Energy and Temperature trend with a slowly moving piston from $t = 0$ till $\tau = 350$ ($N = 1000$, $k = 1$, $\Delta L = 300$)

Due to the piston the mean distance between the particles shrinks. Thus potential energy V decreases as contrary, the kinetic energy increases until the piston stops at $t = 350$. Up to one's expectations (probably) the temperature rises, while the system stays in approximate thermal equilibrium over the whole process (as indicated by the quality plot).
Now the case of a fast piston (figure 17). As nicely shown by the temperature quality, the system completley leaves thermal equilibrium (and doesn't reach it again within 400 seconds). Since we are pumping kinetic energy into the system T_k climbs up very high. Oddly (compared to the slow piston) the potential energy increases, too. This could be because of the harmonic spring potential. As the particles are not bound to the systems initial extent they probably use their gained kinetic energy to leave the system rigorous. The mean distance between two particles could then raise

effectively in contrast to the shrinking effect. A simulation for $\tau = 150$ shows the transition between these effects as V shows a nearly constant trend. Moreover the critical piston velocity conserving thermalization seems to be around $\tau = 280$.

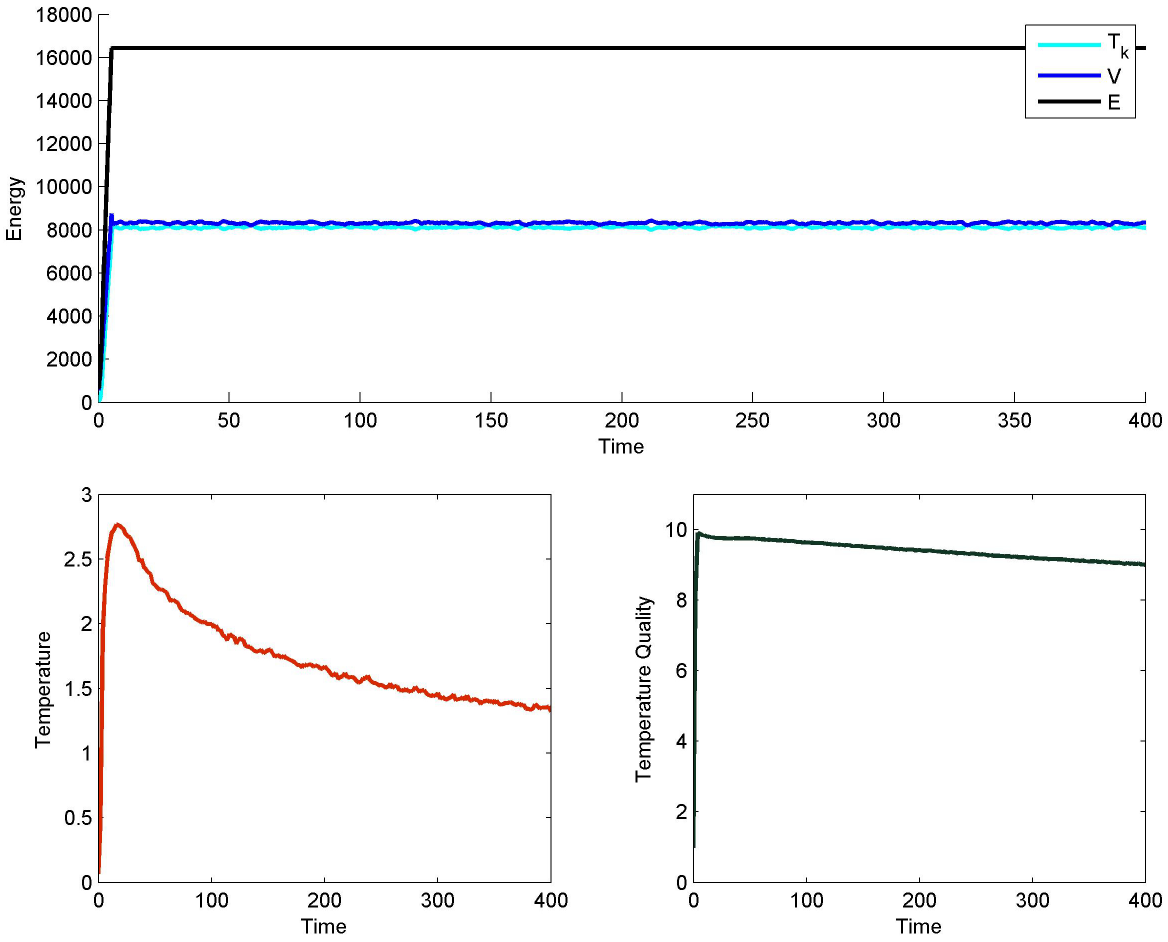

Figure 17: Energy and Temperature trend for a fast piston reducing the systems size within the timespan $[0, 5]$ ($N = 1000$, $k = 1$, $\Delta L = 300$)

3.3.2 Oscillating Walls

After some quick discussion of an piston disturbing a thermalized 1D system we are going to shortly investigate an oscillating wall next. This is a quite realistic case as this corresponds to external sound wave excitation. In an experimental setup this could for instance be realised through a piezo-electric crystall placed on the sample. One purpose could be the determination of the speed of sound in the material (as dependent e.g. on the particles mass). The attributive movie *1DoscwallINV.mpg* shows some typical case in which one can exactly see the wave travelling through the crystall ($v_{wave} \approx 1.5$). Thereby the stimulation is sinusoidal as for all further investigations.

Since our system has a fixed right border the incoming wave will be reflected with a phase shift of π. This reflected wave superposes with the other and in theory forms a standing wave. When excited with the resonance frequency ω_0 the amplitude and thus the energy should explode. We want to analyse our model on this view. For that we firstly measure the systems ω_0 by determining the maximal energy for several excitation frequency. The resonance frequency (and multiples of it) is then clearly visible (figure 18).

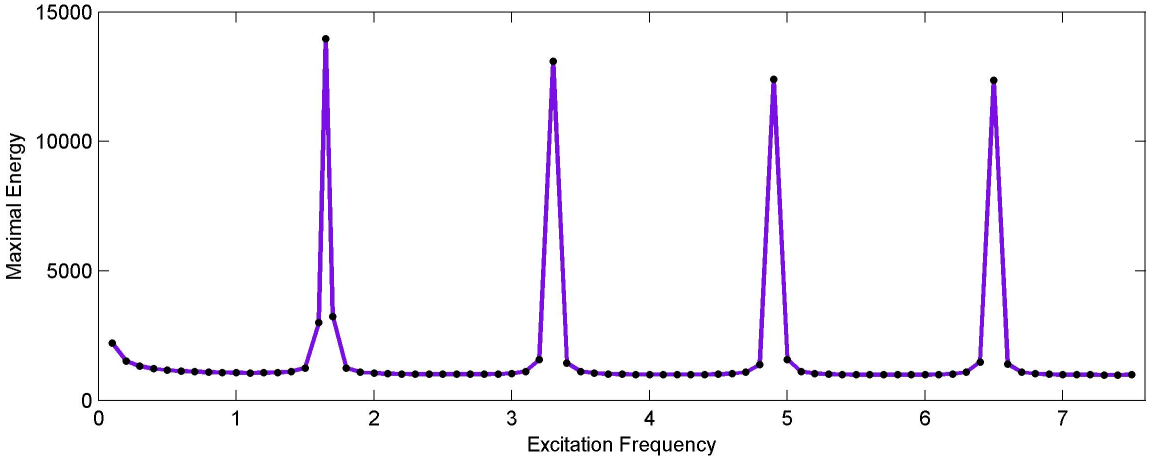

Figure 18: System Reaction (Energy) as a function of the excitation frequency. Multiples of the resonance frequency are indicated by high peaks. Since we abort our simulate after 100 seconds the values of these are not 'correct'. By increasing t_{final} one gets much higher energies. ($N = 20$, $k = 100$, spring potential)

By sizing the peaks one finds (in this special case)

$$\omega_0 \approx 1.65$$

In our model for some reason the contrary waves have slightly different frequencies which results in a beat (german: 'Schwebung') as illustrated in figure 19.

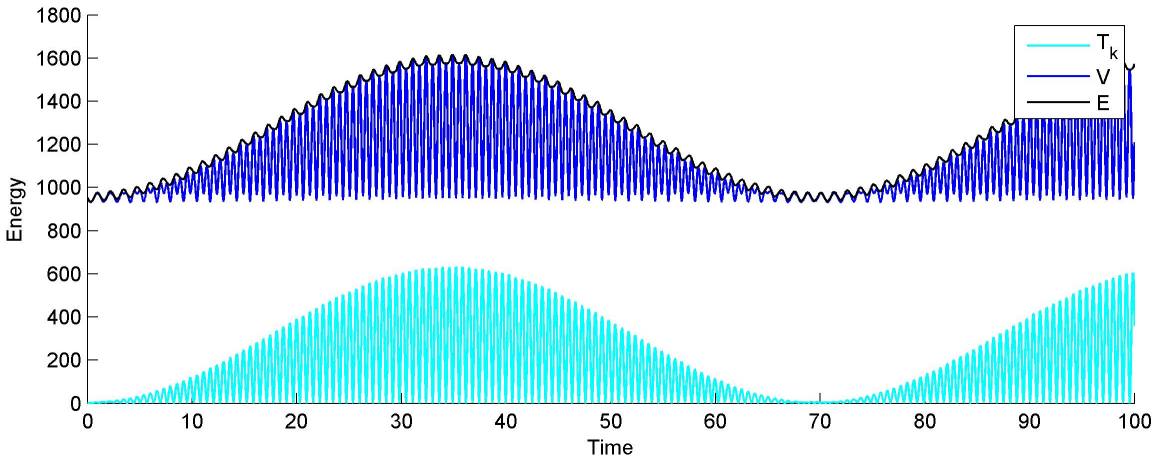

Figure 19: Beat caused by two contrary waves with slightly different frequencies ($N = 20$, $k = 100$, $\omega_{ext} = 5$, spring potential)

One could now ask questions like: What is the frequency difference of those waves and how fast does the potential/kinetic energy oscillate? Measuring the half of the period of the envelope we find

$$\frac{T_{beat}}{2} = \frac{\pi}{\omega_{beat}} \approx 70.56 \;\Rightarrow\; \omega_{beat} \approx 0.045$$

Since $\omega_{beat} = |\omega_{ext} - \omega_{refl}|$ we notice that the disparity is truly small. Furthermore this relation tells us

$$\omega_{refl} = \omega_{ext} - \omega_{beat} = 4.955$$
$$\Rightarrow \omega_{pot/kin} = \frac{\omega_{ext} + \omega_{refl}}{2} \approx 4.98$$

There are several oscillating wall effects left to investige as for instance the response in case of the coulomb potential or the 2D case (which has been implemented within this project), but at this point we relinquish to present those results.

4 Review

The project idea of modelling a solid turned out to be an extraordinary interesting topic. The simple model we implemented quickly became more and more powerful and thereby opened endless possibilities of extension and further investigation. Unfortunately this report could only presents the very basic results.
Though it was a lot of fun to participate in this project, it also was extremely time-consuming (especially considering the worth of only 40% of 6 Credit Points). There is still room for improvement in this regard. Since the timespan covered christmas and new year one had effectively less than a month to implement the model and write this report. Given other lectures and the Fortgeschrittenen Praktikum this was a tough task. In my opinion placing the project in the vacations and therefor leaving off the exam would be the best concept. Moreover it was very difficult to choose the focus of the report, since the instruction was very vague in terms of marking. Nevertheless the resulting model invites to do further studies, which I will surely do.

References

[1] Project instructions - *Computational Physics and Simulations with MATLAB, Goethe-Universität Frankfurt*

[2] Einführung in die Festkörperphysik, Prof. Dr. Müller, Physikalisches Institut