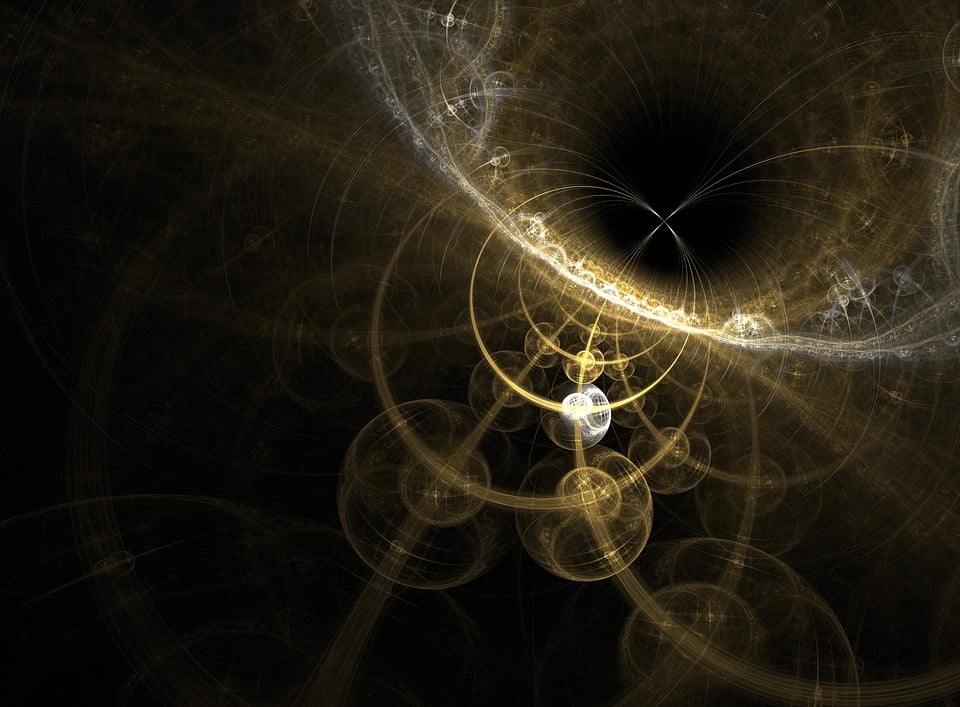
**University of Basel** Switzerland Fall semester 2022



Praktikum

Computational Physics

Report by Sascha Tran

# Information

The task of this course was to solve two large projects out of 6 given. The goal of each project was to develop a short program which solves a simple physical problem.

The projects including their instructions can be found in the following script: <https://comphys.unibas.ch/TEACH/CP/course.pdf>

**Note** that the subsection “Task” is a mere summary of the instructions. A more detailed description can be found in the script.

The aim of this report is to outline the solution of the following two projects:

* Sorting continuously growing data sets
* Determination of the melting temperature of silicon by a molecular dynamics simulation

# Sorting continuously growing data sets

## Introduction

The background and the instructions of this project can be found in the script mentioned in the beginning on page 0-18 to 0-22. All programs regarding this project can be found in folder P18.

In short, trivially inserting an element into a sorted list at the correct position using binary search has a computing time of with N being the size of the current array. This is because every time an element is to be inserted, a new array with a size one bigger than the original must be created, and all elements must be copied into the new array. For large and growing data sets this kind of algorithm can be too slow, thus this project suggests using two-dimensional arrays.

## Task and Expectation

* Implementation of the two-dimensional array including a subroutine that checks that the insertion was done correctly.
* Comparison of the performance of the trivial method with the two-dimensional method.

*It is expected that the two-dimensional method is much better than the trivial method especially with greater amount of data.*

* Examine the influence of the size of the data set of the two-dimensional array on the performance.

*It is expected that the greater the size of the initial data set is, the worse the performance of the two-dimensional array will become since it has to distribute more data at the beginning.*

* Examine the influence of the choice of the side lengths of the two-dimensional array on the performance.

*It is expected that a too big matrix will have a negative impact on the performance but also when the matrix is nearly as big as the initial length because redistribution is most likely then.*

## Results

### Implementation of the two-dimensional array

The implementation of the two-dimensional array can be found in the file *sortingData.f90.* To check that the insertion is done correctly, the subroutines *matrixToVector* and *checkVector* were created. Note that the subroutine checkVector is commented out in the main program because the performance will be measured for the further tasks.

### Performance of the trivial method with the two-dimensional method with different initial sizes of the data set

### Influence of the choice of the side lengths

***How the program works:***

The influence of the side lengths of the initial matrix with the dimension (*m* x *l)* was measured in the main program. The initial number of elements (length) was set to 1000 and 1000 insertions were done for each case.

The initial side lengths m and l were chosen as follows:

1. Initialize an array with 1000 elements and save them to a variable.
2. First set l = 100 and m = 20 (this guarantees that l x m is always greater than length = 1000).
3. Insert 1000 elements and measure the performance and the number of binary searches and save the data into a file.
4. Reset the list of elements to the initial array.
5. Keep the size of l but increase m by 10. If m is 100, increase it by 100. Redo steps 3 to 5 until m = 2100.
6. Now increase l by 100 and do the steps 3 to 5 again.
7. Do step 6 until l = 2100.
8. Redo steps 2 to 7 but with l and m switched.

***The results:***

The data for the following plots can be found in the file *mat.csv* and the plots were generated in the file *plotMat.ipynb.*

Chart, bar chart

Description automatically generated

Figure 1: Performance of inserting 1000 elements in an array that already consists of 1000 elements using a two-dimensional array with different initial sizes m x l.

Figure 1 is a heatmap displaying the performance of inserting 1000 elements into an initial matrix with different sizes m and l. Note that the colour scale is logarithmically, and the greyish squares mean that there are no data for those sizes.

The darker the colour of a square is, the longer the performance was. It can be observed that the darker squares are mostly on the right edge of the map with m being around 2000. There is also a dark square when the matrix has a dimension of 20 x 100. Additionally, it can also be observed that the height l of the matrix should be below 300 or much more above 300 especially if the width m is rather large. Otherwise, the performance is also relatively low.

Chart, table

Description automatically generated

Figure 2: Number of binary searches when inserting 1000 elements in an array that already consists of 1000 elements using a two-dimensional array with different initial sizes m x l.

Figure 2 is a heatmap displaying the total number of binary searches needed, when inserting 1000 elements with a given initial matrix size m x l. Just like the above figure the greyish squares do not consist of data.

Also, the darker the colour of a square is, the higher the number of binary searches was. It can be observed that overall, the greater initM is, the more binary searches were required.

## Discussion

### Influence of the choice of the side lengths

In Figure 1, it was observed that the worst performances were found on the right edge of the map and when the initial matrix is of dimension 20 x 100. Notably, the height m of the matrix is not completely irrelevant if the width is relatively large (1500+).

When inserting 1000 elements into a 20 x 100 matrix, it is very likely that the matrix must be redistributed (even more than once) which might be the reason why the performance is worse than a 900 x 700 matrix for example.

However, the 100 x 20 matrix does not seem to support the above thesis. In general, I expected the heatmap to be much more symmetric in terms of that there is not much of a big difference of the performance between a matrix with dimension m x l and l x m.

Since the size of the width has a much more negative impact on the performance than the height, I assumed that more binary searches could have been done in a broader matrix than its corresponding transposed matrix. In fact, Figure 2 visualizes it clearly that the width had a great impact on the number of binary searches.

1. Determination of the melting temperature of silicon by a molecular dynamics simulation

## Introduction

The background and the instructions of this project can be found in the script mentioned in the beginning on page 0-131 to 0-133. All programs regarding this project can be found in folder P131.

Briefly explained, in this project we want to find the melting temperature of silicon by running two molecular dynamics simulation. In both simulations a chunk of liquid silicon is sandwiched between crystalline silicon. The difference between these two simulations is the different initial velocities of the liquid part. One can imagine that one chunk is extremely hot and the other is only moderately hot.

## Material

This section outlines the files and software specifically required for this project.

Files[[1]](#footnote-2):

* bazant\_lib.f90
* md.f90
* cold.dat
* hot.dat

Software:

* Vesta[[2]](#footnote-3)

## Task and Expectation

### Harmonic oscillator

* Write a little program that implements the velocity Verlet algorithm (Eq. 98 slide 0-113). Test the program for a harmonic oscillator.
* Check that the total energy (potential plus kinetic) is approximately conserved and that that the trajectories are periodic.
* Observe the quality of the energy conservation as a function of the size of the time step h.

*It is expected that the quality of the energy conservation increases with smaller time steps. After all, the trajectories are much more precise with smaller time steps.*

### Melting temperature of silicone

* Add the Verlet algorithm and the calculation of the temperature T of the system using the given equation (Eq. 113) to the file *md.f90*.
* Verify that the energy is approximately conserved when running the MD program with hot.dat or cold.dat.
* Determine the melting temperature and visualize the system using V\_Sim.

*The energy should be approximately conserved since the MD simulation is mainly based on the above Verlet algorithm.*

## Results

### Harmonic oscillator

* The implementation can be found in the file *verlet.f90* and the plots were created in *plot.ipynb.*
* The simulation was held with the following initial parameters:

R = 1.0, V = 0.0, M = 1.0, h = 0.001, k = 2.0, len = 30’000 (len is the number of time steps)

With these initial parameters Figure 1 shows that the trajectory is indeed periodic (“Block 1”).

Furthermore, the total energy is approximately conserved. The absolute difference between the first and last computed energy is 5.0 ∙ 10-7 J which is nearly 0.

The computation can be found in the plot-file “Block 2”.

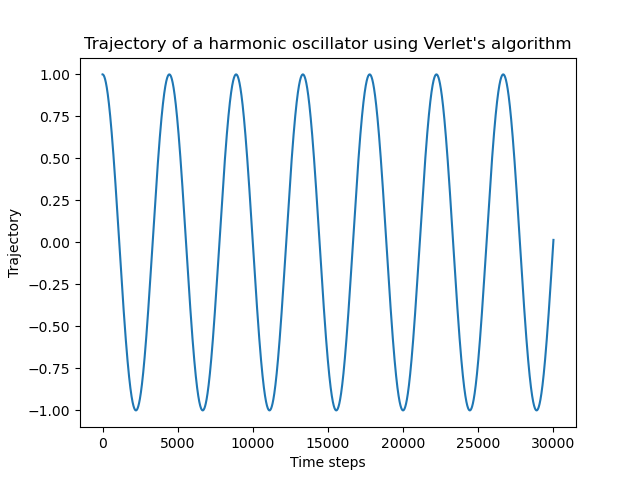


Figure 3: Trajectory of a harmonic oscillator

Table 1: Quality of energy conservation

as a function of the time steps.

|  |  |
| --- | --- |
| **Time step** | **Max. difference in total energy [J]** |
| 1 | 0.5000000 |
| 1E-01 | 0.4999997E-2 |
| 1E-02 | 0.5000000E-4 |
| 1E-03 | 0.5000000E-6 |
| 1E-04 | 0.5000004E-8 |
| 1E-05 | 0.8445467E-11 |
| 1E-06 | 0.2043921E-12 |
| 1E-07 | 0.4676259E-12 |
| 1E-08 | 0.6609158E-12 |
| 1E-09 | 0.6150636E-13 |
| 1E-10 | 0.1543210E-12 |
| 1E-11 | 0.1800782E-12 |
| 1E-12 | 0.1776357E-14 |
| 1E-13 | 0.000000 |
| 1E-14 | 0.000000 |
| 1E-15 | 0.000000 |
| 1E-16 | 0.000000 |
| 1E-17 | 0.000000 |
| 1E-18 | 0.000000 |
| 1E-19 | 0.000000 |
| 1E-20 | 0.000000 |

* Table 1 displays the difference between the greatest and smallest total energy in each simulation. The initial parameters are the same as above except for the time steps. Each row is a simulation with a different time step.

It is visible that in general, the smaller the time steps are the better the energy conservation is.

### Melting temperature of silicone

* The Verlet algorithm as well as the calculation of the temperature can be found in the file *md.f90.*
* Chart, scatter chart

  Description automatically generatedWhen running the MD simulation for 1000 time steps with cold.dat, the absolute energy difference of the last time step and the first time step is 0.02 J and with hot.dat the difference is 0.008 J. Therefore, it can be confirmed that the energy is conserved. The computation can be found in the plot-file (“Block 3”).

*Figure 4: Temperature of MD simulation using hot.dat and cold.dat.*

Table 2: Melting temperature of silicone. Temperature

computed by taking the average of the data points

of the MD simulation starting from the 100000th MD step.

|  |  |
| --- | --- |
| **Data points from** | **Melting Temperature [K]** |
| hot.dat | 1531.24 |
| cold.dat | 1533.98 |
| both files | 1532.61 |

* In Figure 2, both graphs (red and blue) have reached their equilibrium after about 100000 MD steps. To determine the melting temperature, the average of all data points of hot.dat respectively cold.dat starting at 100000 MD steps was evaluated. Additionally, the average of both results was taken which gives us a melting temperature of 1532.61 K (highlighted in Table 2). The computation can be found in the plot-file (“Block 4”).
* The following figures on the next two pages are a visualization of the system in the initial state, when the chunk was inserted, and the final state after the MD simulation. As a result of a recommendation, the visualization was performed using Vesta instead of V\_Sim.

**Visualization of the system**

**with the insertion of the hot liquid silicone**

*hot.dat*

A picture containing fabric

Description automatically generated

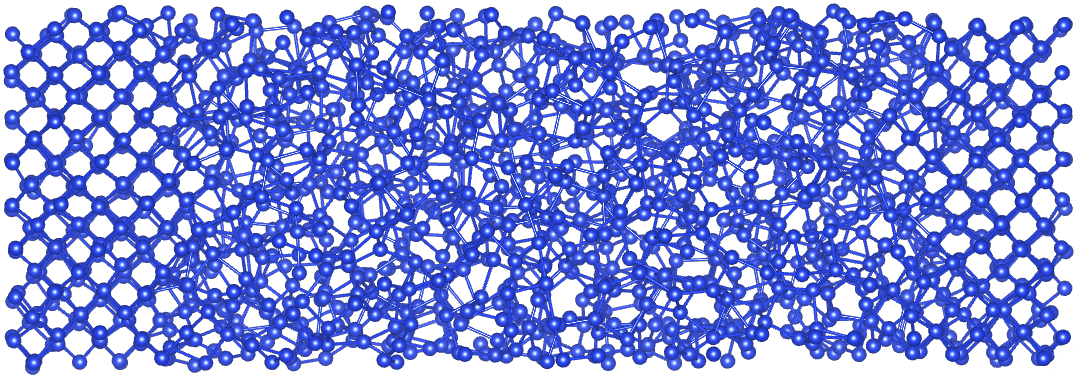


Figure 5: Initial state of the system (top) after insertion of the **hot liquid** silicone and final state of the system (bottom) at the end of the MD simulation.

**Visualization of the system**

**with the insertion of the moderately hot liquid silicone**

*cold.dat*

A picture containing fabric

Description automatically generated

A picture containing fabric

Description automatically generated

Figure 6: Initial state of the system (top) after insertion of the **moderately** **hot liquid** silicone and final state of the system (bottom) at the end of the MD simulation.

1. <https://comphys.unibas.ch/teaching.htm>, The files are found in the first .tar folder of this website. [↑](#footnote-ref-2)
2. <https://jp-minerals.org/vesta/en/> [↑](#footnote-ref-3)