Delft University of Technology

COMPUTATIONAL PHYSICS

AP3082

Molecular Dynamics Simulation of Argon Atoms

Project 1

Authors:
Coen de Jong (4361598)
Peter-Jan Derks (4494997)
Vincent van de Kerkhof (4386302)

March 27, 2019

Abstract

In the field of computational physics, molecular dynamics is a computer simulation method for studying the physical properties of atoms. In this report, we present a molecular dynamics simulation of Argon atoms programmed in Python. Our simulation obeys physical laws and is in accordance with previous work. Using our simulator we calculate the pair correlation function and the compressibility factor of Argon for different temperatures and densities. By analyzing the pair correlation function for different temperatures and densities we have successfully found circumstances under which Argon is a solid, liquid and gas.

Keywords: *Molecular dynamics, Computational physics, Pair correlation function Argon, Compressability factor*

Contents

1 Introduction

The goal of this project is to model a molecular dynamic simulations with argon particles. These kinds of simulations are created to have a more in-depth understanding of how these molecular particles interact with each other and obtain and proof theory. Phase transitions for instance are very easy to observe in these kinds of simulations. This is modelled using the Lennard-Jones potential, which is a good approximation for these kind of simulations. This due to the fact that there are no dominant quantum effect occurring in the size of the system being modelled. Argon is used in the simulation as there is a large amount of studies covering Argon atoms, thus having an extensive amount of data for comparison the validity of the simulation.

The research is focused on determining the kinetic and potential energy, temperature, compressibility factor and pair correlation. And comparing this to the data provided in the paper provided by Jos Thijssen.[2]

2 Theory

2.1 Interaction of Argon atoms

The goal of this project is to simulate a phase transitions in Argon gas. the first step to do this, is to look at how single Argon atoms interact with each other. Argon atoms do not interact via Coulomb interactions, as argon atoms have a neutral charge density. This means that the main interaction is a result of displacements between the nucleus and the electron cloud of an Argon atom. This gives rise to a small dipole moment for each atom. These dipoles interact with each other resulting in an attractive or repulsive force. This can be modelled by applying a Lennard-Jones potential to the system as this simulates the attractive and repulsive forces on a dipole atom. This is quite an accurate model as there are not chemical reactions between the argon Atoms themselves. The formula for the Lennard-Jones potential is:

$$U(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) \tag{1}$$

σ is the characteristic length, which is for Argon atoms 3.405 Angstrom, $\epsilon = 119.8K \cdot k_b$ and r is the absolute length between two particles. The $1/r^6$ term describes the interaction between dipoles, and $1/r^{12}$ is a repulsive term which comes from the core of the atoms.

In the simulation of the motion of Argon atoms, the normal SI units lead to different magnitudes in a lot of numbers. This can result in round-off errors and is not convenient to work with. Therefore the Lennard-Jones potential is rewritten to some natural units. The distance between particles r for example, can be written as $\tilde{r} = r/\sigma$. But also the whole potential can be divided by ϵ. Taking both transformations into account results in the following dimensionless Lennard-Jones potential.

$$\tilde{U}(\tilde{r}) = U(\tilde{r})/\epsilon = 4(\tilde{r}^{-12} - \tilde{r}^{-6}) \tag{2}$$

With this dimensionless potential, the force acting on each particle can be calculated using Newtons second law of motion.

$$m\frac{d^2\vec{\tilde{r}}_i}{dt} = \sum_{i,j}\vec{F}(\vec{r}_{ij}) = -\nabla U(\vec{r}) \tag{3}$$

Where r_i is the location of particle i, r_{ij} is the absolute difference between to particles i and j and $\nabla U(r) = \frac{dU}{d\tilde{r}}\frac{\vec{\tilde{r}}}{\tilde{r}}$ the gradient of the Lennard-Jones potential. which is given by:

$$\frac{dU}{d\tilde{r}}\frac{\vec{\tilde{r}}}{\tilde{r}} = (-48\tilde{r}^{-14} + 24\tilde{r}^{-8})\vec{\tilde{r_{ij}}} \tag{4}$$

The following results were obtained by substituting equation 4 in equation 3.

$$m\frac{d^2\vec{\tilde{r}}_i}{dt} = (-48\tilde{r}^{-14} + 24\tilde{r}^{-8})\vec{\tilde{r_{ij}}} \tag{5}$$

From this equation, the motion of the particles can be calculated. This is however an approximation since the differential equation can not be solved exactly. Different methods can be used to solve the equation of motion numerically. In this project the Euler method and the Verlet algorithm are used.

2.2 Constants

The following units were chosen to accommodate an easy overview of calculations and as to use the same units in literature and the data provided in the papers.

$$K_B = 1.38 * 10^-23\ j * k^{-1}$$

$$\epsilon = 119.8 * kb = 1.65 * 10^{-21}\ j$$

$$\sigma = 3.405 * 10^{-10}\ m$$

$$M_{argon} = 6.6 * 10^{-26}\ kg$$

K_B is the Boltzmann constant, ϵ the scaling unit for energy, σ is the scaling constant for distance and M_{argon} is the mass of the argon atoms in kg.

2.3 Units and unit-less calculations

The entire simulation is done with unitless variables. This is done as it has a multitude of advantages over not doing so.

1. it prevents rounding errors in the memory, as there is a limit to the number one can represent with the traditionally used memory
2. speeds up calculations, as the formulas become less demanding on the cpu.
3. gives insight on which parts of the equation can be neglected as the can be scaled up or down with their corresponding units

as these provides higher accuracy due to less rounding errors, and higher performance due to simplification of the mathematical formulas. The following unitless variables were used during the simulation and have the following units:

The symbol with a ∼ are dimensionless units

$$E = \epsilon \tilde{E} \tag{6}$$

$$L = \sigma \tilde{L} \tag{7}$$

$$t = \sqrt{\frac{m\sigma^2}{\epsilon}} \tag{8}$$

$$T = 119.8\tilde{T} \tag{9}$$

2.4 Virial theorem

The virial Theorem was used in order to calculate the compressibility of the system.

$$\frac{\beta P}{\rho} = 1 - \frac{\beta}{3N}\left\langle \frac{1}{2}\sum_{i,j} r_{ij}\frac{dU}{dr_{ij}} \right\rangle \tag{10}$$

The brackets are there to indicate it is a time average value over time. This is done to have a value which can be easily compared to the data supplied in the book of Jos Thijssen.

2.5 Pair Correlation Function

A good measure for the behavior of many moving particles is the pair correlation function. The pair correlation function describes how density varies as a function of distance from a reference particle.

$$g(r) = \frac{2V}{N(N-1)}\frac{\langle n(r)\rangle}{4\pi r^2 \Delta r} \tag{11}$$

The pair correlation function has specific forms in different states of matter.

3 Computational method

3.1 The infinite finite box method - minimal image convention

The goal of the simulation is to simulate an infinite number of argon atoms, but as the simulation has a finite time and a finite amount of transistors this is not possible. A periodic boundary condition is applied to the simulation as to overcome this problem. This entails that the simulation is simulating a finite box of lengths L, in which argon atoms are floating freely in the box. With the only condition that if a argon atom crosses the boundary, that it will disappear and re-emerge on the opposing boundary with the same velocity as before. However, this has some subtle influences on the system itself.

1. If a particle leaves the box, it reenters at the opposite side

2. The evaluation of the interaction between the argon atoms is influenced by the periodic boundary conditions. This is because the boundary condition introduces copies of the argon atoms themselves.

3. The only argon atoms action on a argon atom are the ones which are the closest to this particular atom. This means that one of the argon atoms might be closest at the moment a copy outside the box act on it.

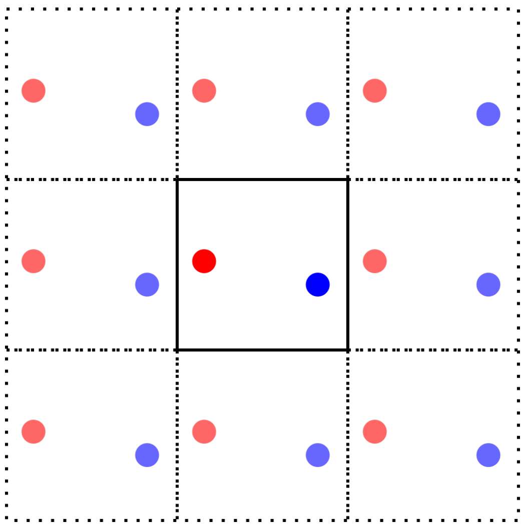

Figure 1: figure showing the boundary conditions imposed on the system. One can observe that the closest particles might not be in the initial box. This image is taken from a document of Akhmerov Anton at TUD[1]

3.2 Initial state

At the start of the simulation, a crystal lattice is created of argon atoms in a face centered cubic latice (fcc). This way the initial distribution of the argon molecules have a consistent potential energy, which is needed as otherwise there is a large possibility for the system to have an immense amount of energy stored. In figure 2 a plot of 108 particles arranged in a fcc crystal structure is shown.

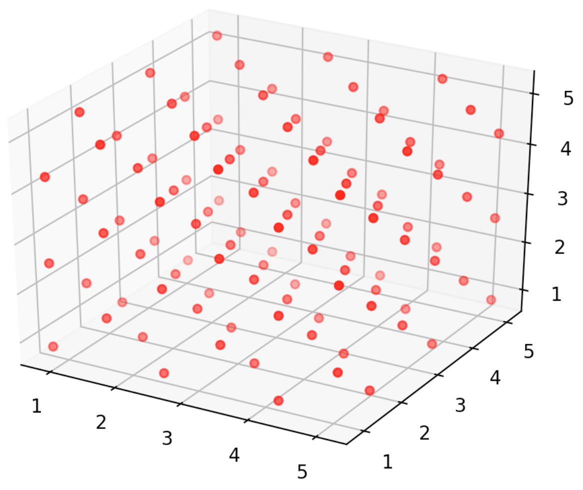

Figure 2: Plot of 108 particles arranged in fcc structure. The plot was created by running the file fcc_plot.py

The second step is to assign a starting velocity to all argon atoms. This is done by using the Maxwell distribution, which gives a kinetic energy of all atoms in a gas with temperature as only requirement.

3.3 Time evolution

The entire simulation is performed using a discretized model which uses a finite difference method. Two different kind of methods are introduced in the manual to do these kinds of simulations.[1] The Euler and the Verlet's method. The Euler's method gives you the following two equations:

$$x(t+h) = x(t) + hv(t) \tag{12}$$

$$v(t+h) = v(t) + \frac{h}{m}F(x(t)) \tag{13}$$

Where h is a very small but finite time step, which is used to obtain time integration. The reason why this method is not used in the final simulation is, because the Euler's method doesn't conserve the energy of the system in the microcanonical ensemble.

3.4 Verlet algorithm

A better method for calculating the time evolution of the system is using the Verlet algorithm, as this method conserves the total energy of the system. This is done by using the taylor expansion at $t \pm h$ and gives the following solution for the system. The info is obtained from Jos Thijssen paper on argon gas [2] and the paper of Verlet. [3]

$$x(t+h) = x(t) + hv(t) + \frac{h^2}{2m}F(x(t)) \tag{14}$$

$$v(t+h) = v(t) + \frac{h}{2m}(F(x(t)) + F(x(t+h))) \tag{15}$$

3.5 Velocity rescaling

The velocity is rescaled several times during the initial set-up as to ensure the system is stable and isn't fluctuating between high amount of kinetic and potential energy. This is done as the simulation should be done at a steady desired temperature. The temperature of the system is linked to the kinetic energy of all particles by the following equation.

$$U_{kin} = \frac{3}{2}(N-1)K_BT \tag{16}$$

U_{kin} is the total kinetic energy of all particles in the simulation, N is the number of particles, K_B is the boltzmann constant and T is the temperature.

The equipartion theorem tells us that the total amount of energy in the system is equal to the number of degrees of freedom and scales with the number of particles minus 1 and $\frac{K_BT}{2}$. Using this definition and using the fact that the simulation is using unitless variables. The following rescaling variable is used during the rescaling process.[2]

$$v_{particles,desired} = \lambda * v_{particles,current} \tag{17}$$

$$\lambda = \sqrt{\frac{(N-1)3K_BT}{m\sum_{i=1}^{N} \mid \vec{v_i} \mid^2}} \tag{18}$$

The rescaling factor is applied to the velocity of all particles a multitude of time during the initial phase. This way the system can calibrate itself to a stable system, which afterwards can be used for the simulation and to perform research.

3.6 Bootstrap

With the molecular dynamics simulation different expectation values can be calculated. However since the time the simulation runs does not go to infinite, these calculations are only approximations of the expectation value. Therefore a method is needed to estimate the errors of the results obtained from the simulation. If the data is uncorrelated, this can be done with the simple formula of the variance. But in the molecular dynamics simulation the data generated is not statistically independent. Each new configuration is dependent on its previous one. A way to overcome this problem is to implement the bootsrap method. Bootstrapping is a resampling method which takes N random data points from the data generated and computes with this set the expectation value. This procedure is repeated n times, and finally from this set of expectation values the variance is calculated with the well known formula:

$$\sigma_A = \sqrt{\langle A^2 \rangle - \langle A \rangle^2} \tag{19}$$

4 Results

In this section the results of running the code are discussed. In section 4.1 we motivate that our code correctly implements a physical system by showing that it satisfies physical laws. In section 4.2 we motivate that our code can be used to calculate observables by comparing outputs of our code to values in Verlet's paper [3]. In section 4.3 we present and discuss physical observable calculated. In section 4.4 we comment on the performance of our code and give possible improvements.

4.1 Correctness checks with physical laws

In the micro-canonical ensemble the total energy of the system is expected to be constant. Figure 7 clearly shows that the total energy is constant.

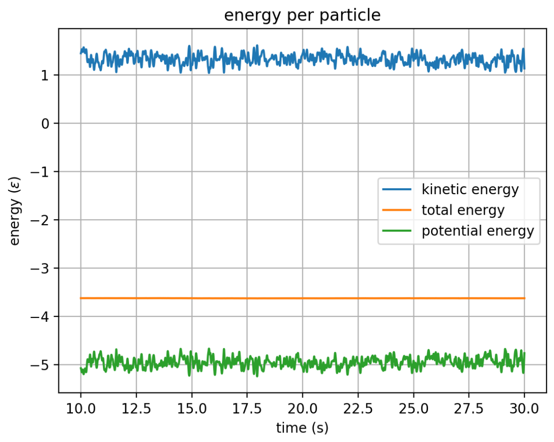

Figure 3: Energy plot created by running the file simulation.py. A system with 32 particles, temperature 1.0 (*epsilon*) and density of 0.8 .

4.2 Comparison with Verlet's paper

In the following section a comparison is made with the data obtained from the simulation and the data from the Verlet paper. The simulation is executed 32 times for different input parameters. For each simulation the values for the density, initial temperature and average temperature are given in appendix B. Also the error which is calculated with the bootstrap method is given in this table. The number of particles in the different simulations are held constant at 108 particles. In the next figure the potential energy is plotted for different simulations.

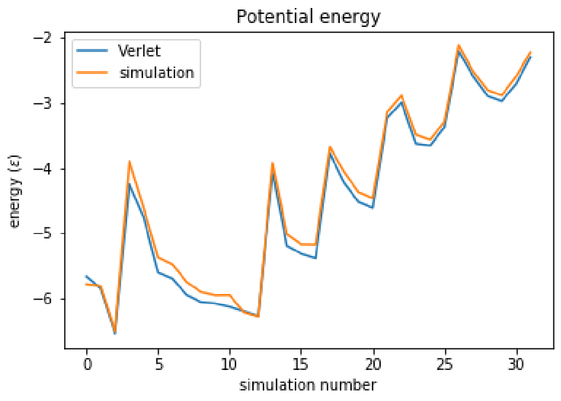

Figure 4: Potential energy per particle for different simulations. The orange line indicates the values obtained from the simulation and the blue line indicates the values from the Verlet paper. The values of the density, temperature and errors of each simulation can be seen in appendix B.

In the next figure the compressibility factor $\frac{\beta P}{\rho}$ is plotted for different simulations. In the same plot the compressibility factor of the Verlet paper is given. The data shown in this plot can also be seen in appendix B

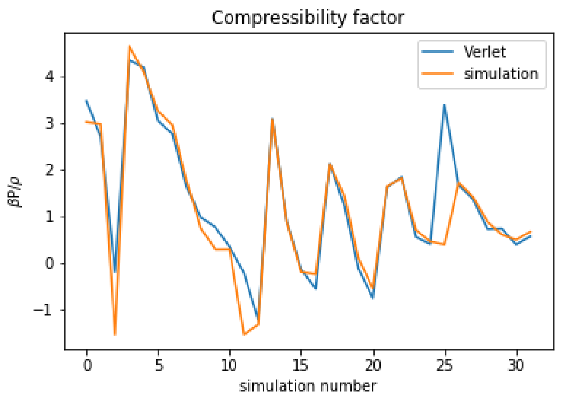

Figure 5: Compressibility factor for different simulations. The orange line indicates the values obtained from the simulation and the blue line indicates the values from the Verlet paper.

4.3 Observables

With the molecular simulation different phases are observed. This is indicated with a plot of the pair correlation function shown in the next figure. In this plot a distinction can be made between the solid, liquid and gas phase. For the solid state phase, the pair correlation functions shows characteristic peaks at distances were atoms are located. The broadening of the peaks is due to fluctuations around there lattice sites. For the gas state, the atoms do not have a regular structure but will diffuse through the whole box. In the liquid state the atoms also do not have a constant structure.

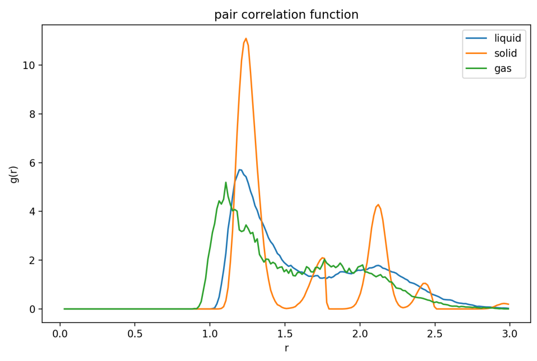

Figure 6: Pair correlation function created by running the file correlation_function_plot.py. The entire code (initial state, equilibration phase, production phase and data processing) is run for three values of input temperature and density. The orange line is the correlation function corresponding to $T = 0.5$, $\rho = 1.2$, the blue line corresponds to input $T = 1.0$, $\rho = 0.8$, the green line corresponds to $T = 3.0$, $\rho = 1.2$

In the next figure two isotherms are represented. The data is obtained by running the simulation with 32 particles and varying the density and the temperature.

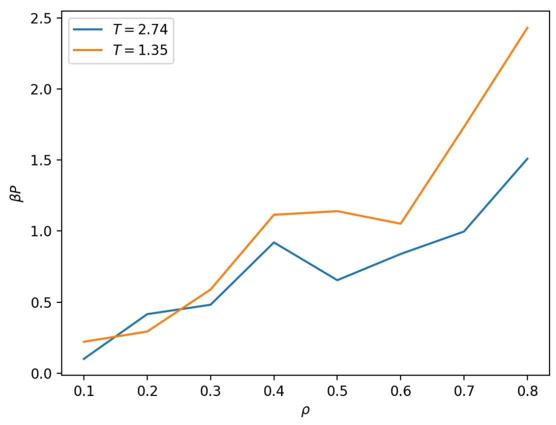

Figure 7: For the isotherms T = 2.74 and T= 1.35, βP is plotted as a function of ρ. The plot is created by running the file compressibility_density_plot.py.

4.4 Performance

Where possible we have used numpy array instead of for loops to improve the efficiency of our code. The performance was tested on a **Asus laptop, with an "intel core I7-6500 CPU at 2.5 Ghz** , the run-time is shown below:

number of particles	Run-time(sec)
32	37.62
108	101.22
256	623.43

A simple custom quadratic fit shows us that the time increases according to the following formula

$$t(n) = 52 - 0.8n + 0.01n^2 \qquad (20)$$

This means that the code would run for 6825.76 seconds for 864 particles, which is about 113 minutes. This means that the code is still valid for larger number of particles, but the time required is still a large issue, which further updates might combat.

One of the initial things done to optimize the code was using numpy arrays, instead of for-loops as this dramatically increased the speed the computer could calculate the time evolution. And by introducing mask arrays(part of numpy), it was possible to gain even more performance as it skipped all the zero elements in all the arrays.

Although one of the main problems in the code is the equilibration time as this part of the code is still slow compared to the other parts of the simulation.

5 Conclusion

In this project a molecular dynamic simulation of Argon atoms is described. The simulation can be executed for different number of particles, temperature and density. From the energy plot, it can be seen that the total energy is conserved. The time evolution of the particles is calculated using Verlet's algorithm.

The results of the simulation are in line with the data provided in the paper of Verlet's[3]. This can be observed in 4.1. The graphs shows all the measurements and values from the Verlet's paper, which are in line with the data obtained from the simulation. Although it has to be noted that the value may vary, when the system hits a unstable configuration, which seems to occasionally happen during higher temperatures.

The compressibility seems to be correct as the simulation seems to obtain similar results as the Verlet's paper and is nearly identically. Further improvements to this part of the code can be made by making sure the configuration of the system is more stable. This can be achieved by simulating the system for a larger number of particles in a larger box. This way the particles have more degrees of freedom to feel each other and are they able to distribute themselves more evenly.

The simulation is able to check if the configuration of argon atoms is a liquid, gas, or a solid. This was achieved by calculating the pair correlation function, and simulating for different temperatures and densities. This way it was possible to check if the gas was in the correct phase.

The performance of the code was checked by running the code for an increasing amount of particles and fitting a quadratic function to the runtime and the number of particles. This resulted in the following function $t(n) = 52 - 0.8 * n + 0.01 * n^2$, which means that the code would run for about 113 minutes for 864 particles, according to the fitted function. This means

That the code is still suitable for larger number of particles, but improvements can be made such that the run-time can be improved upon.

References

Akhmerov, A. (2019, sep). *Manual argon atoms.* (Computational Physics project 1 - Argon gass model)

Thijssen, J. (2007). *Computational physics* (2nd ed.). Cambridge University Press. doi: 10.1017/CBO9781139171397

Verlet, L. (1967, Jul). Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, *159*, 98–103. Retrieved from https://link.aps.org/doi/10.1103/PhysRev.159.98 doi: 10.1103/PhysRev.159.98

6 Appendix A

6.1 Individual contributions

It is challenging for us to clearly write down how we split the work, because on almost every section of the code and paper multiple people contributed. What often happened was that one of us wrote a function and someone else debugged it, made it more efficient, or changed the function such that it could be used in a 3D simulation instead of a 2D simulation. Therefore in this section we have written down who made the biggest contribution to each code section:

Coen:

- Code contributions
  - dimensionless variables
  - equilibration phase, rescaling velocity
  - histogram of distances between pairs
  - potential energy
  - correctness check using drift velocity
  - Maxwell distribution for initial state particle velocities
- report contributions
  - theory section 2.1 - 2.4
  - computational method 3.1 - 3.6
  - performance 4.4
  - conclusion

Vincent:

- Code contributions
  - 2D model (during the first few weeks of the project).
  - compressibility factor
  - virial theorem
  - csv file integration
  - running the simulation to create the table in Appendix B
- Report contributions
  - Introduction
  - Results 4.2
  - figures 5, 6

Peter-Jan:

- Code contributions
  - 2D model (during the first few weeks of the project).
  - minimal distance convention
  - overall code structure
  - pair correlation function
  - fcc lattice
  - calculation of forces between particles
- Report contributions
  - Abstract
  - figures 2,4,7,8
  - Results 4.3
  - Appendix A

6.2 Work organization

To organize our work, we discussed in the beginning of each week what each of us will work on individually. At the lectures on Wednesday morning we all looked at the progress made, and helped each other if there were problems. To prevent us working on the same sections of code we used the issue board in gitlab.

7 Appendix B

density	T_{set}	$T_{average}$	Compressibility	potential energy	T_{error}	$Compres_{error}$	U_{error}
0.88	1	0.968	3.034	-5.777	0.0007	0.006	0.001
0.88	0.94	0.941	2.991	-5.803	0.0006	0.005	0.001
0.88	0.591	0.592	-1.520	-6.500	0.0004	0.006	0.0006
0.85	2.889	3.070	4.656	-3.899	0.001	0.004	0.002
0.85	2.202	2.210	4.103	-4.616	0.001	0.005	0.002
0.85	1.214	1.241	3.264	-5.370	0.0009	0.005	0.001
0.85	1.128	1.120	2.973	-5.469	0.0008	0.006	0.001
0.85	0.88	0.850	1.775	-5.743	0.0005	0.006	0.0009
0.85	0.786	0.799	0.753	-5.894	0.0005	0.006	0.0009
0.85	0.76	0.703	0.306	-5.944	0.0005	0.007	0.0007
0.85	0.719	0.685	0.308	-5.942	0.0005	0.006	0.0007
0.85	0.658	0.660	-1.519	-6.206	0.0005	0.007	0.0007
0.85	0.591	0.633	-1.310	-6.268	0.0004	0.007	0.0006
0.75	2.849	2.753	3.083	-3.922	0.002	0.004	0.002
0.75	1.069	1.040	0.883	-5.011	0.0006	0.005	0.001
0.75	0.881	0.838	-0.179	-5.167	0.0006	0.006	0.0008
0.75	0.827	0.800	-0.227	-5.170	0.0005	0.007	0.0008
0.65	2.557	2.509	2.132	-3.680	0.001	0.004	0.002
0.65	1.585	1.600	1.461	-4.064	0.0009	0.005	0.001
0.65	1.036	1.038	0.111	-4.371	0.0007	0.006	0.0008
0.65	0.9	0.895	-0.533	-4.464	0.0005	0.007	0.0008
0.55	2.645	2.566	1.649	-3.149	0.001	0.004	0.001
0.5426	3.26	3.297	1.832	-2.894	0.001	0.004	0.002
0.5426	1.404	1.453	0.721	-3.492	0.0007	0.005	0.001
0.5426	1.326	1.308	0.474	-3.566	0.0006	0.005	0.001
0.5	1.36	1.331	0.409	-3.297	0.0008	0.005	0.001
0.45	4.625	4.839	1.731	-2.128	0.001	0.003	0.002
0.45	2.935	2.943	1.430	-2.524	0.001	0.004	0.001
0.45	1.744	1.831	0.900	-2.811	0.0008	0.004	0.001
0.45	1.552	1.558	0.620	-2.892	0.0007	0.004	0.0009
0.4	1.462	1.500	0.511	-2.605	0.0008	0.005	0.001
0.35	1.62	1.685	0.681	-2.236	0.0007	0.004	0.001