Molecular Dynamics and Metropolis Monte Carlo Simulation of Lennard-Jones Argon liquid

Course Project for FK7029

Simulation Methods in Statistical Physics, Spring 2013

at Stockholm University

Author: Sebastian Arnoldt

March 11, 2013

Contents

1 Introduction

The aim of this project is to produce stable computer simulations of a Lennard-Jones Argon liquid by using Molecular Dynamics and Monte Carlo techniques. Once the simulation code runs stable, measurements of respective energies and the heat capacity are performed. Moreover the radial distribution function is computed. The foundation of this project are the somewhat vague lab instructions found in [6].

A personal side-challenge of this project is to write my first simulation using Python and trying to refute the common misconception that Python may not be used to write physics simulations. To run this code you need to install the following:

- Python 2.7.3
- SciPy (for computations)
- Numpy (for computations)
- vPython (for 3D visualization)
- matplotlib (for plotting in data analysis part)

This code has been tested on Linux Ubuntu 12.04 LTE, 32-bit. The reason for installing Python 2.7.3 (instead of the latest version) is that the package SciPy.Weave currently only runs under Python 2.7. All of the above can be installed via the Ubuntu software center. In this code I try to find the right balance between readability, taking advantage of Python features and performance. This being my first Python code, I follow community advise from stackoverflow.com and www.scipy.org, in particular the references [5] and [4].

2 Approach

When simulating a Lennard-Jones (LJ) fluid in Molecular Dynamics (MD) code from a previously written Monte Carlo (MC) simulation of the same system can be re-used and vice versa. The main difference between both implementations is that in MD we need a scheme to integrate the Newtonian equations of motion. Contrarily in MC we need a scheme to accept or reject configurations of the system under study that have been generated through random processes ([1], [2]).

Keeping this in mind, I have designed both simulations in the following manner. There are two main routines, monteCarlo.py and molDyn.py for the corresponding MD and MC simulations. Both main routines source the system parameters from config.py and the functionality from functions.py. In config.py all system parameters are stored and imported from other routines by calling *from config import**. To distinguish the parameters from regular variables, they are defined in upper case letters, for instance TEMPERATURE. The key functionalities such as generating an fcc-lattice or calculating the LJ potential are coded in functions.py.

Data analysis is performed by saving the results of the simulations to text files and analysing them in separate Python scripts, dataAnalysisMD_N.py respectively dataAnalysisMC_N.py.

Throughout this project report I use the reduced units introduced in [2, p.40]. The choice for these units is σ as unit of length, ϵ as unit of energy and m as unit of mass. From these choices the unit of temperature is ϵ/k_B and the unit of time $\sigma\sqrt{m/\epsilon}$. In the following, I denote time by the symbol τ. In reduced units, the LJ potential becomes:

$$U^{LJ}(r) = 4\left[\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right] \tag{1}$$

.

Fig. 1 below shows a conversion table between reduced units and SI units for LJ argon. The above LJ potential is used in its truncated and shifted form for all simulations below. This method requires to apply an energy correction

during the calculation of the potential energy. Following [2, pp.36] this energy correction is:

$$ecut = 4\left[\left(\frac{1}{r_c}\right)^{12} - \left(\frac{1}{r_c}\right)^{6}\right] \tag{2}$$

where r_c is the cut off radius of half length of the simulation box.

Quantity	Reduced units		Real units
temperature	$T^* = 1$	$\leftrightarrow$	$T = 119.8$ K
density	$\rho^* = 1.0$	$\leftrightarrow$	$\rho = 1680$ kg/m^3
time	$\Delta t^* = 0.005$	$\leftrightarrow$	$\Delta t = 1.09 \times 10^{-14}$ s
pressure	$P^* = 1$	$\leftrightarrow$	$P = 41.9$ MPa

Table 3.1: Translation of reduced units to real units for Lennard-Jones argon ($\epsilon/k_B = 119.8$ K, $\sigma = 3.405 \times 10^{-10}$m, $M = 0.03994$ kg/mol)

Figure 1: Conversion between reduced units and SI units, taken from [2, p.42].

3 Setup

3.1 Preliminary Setup

Initially, a system of three atoms in a three dimensional box is set-up in the reduced LJ potential. The three atoms are assigned suitable velocities in different directions and equations of motion are integrated with the Velocity Verlet Algorithm given in [3, p.203]. After a few steps, computer calculations are checked by hand. This procedure reveals the most serious programming bugs, for instance divergences in total energy. Moreover the behaviour of the atoms is simulated in 3D to observe whether the trajectories of the atoms follow the expected physical behaviour and whether the implementation of periodic boundary conditions works correctly.

3.2 Molecular Dynamics - 108 atoms

Having debugged most of the program code, a more realistic simulation is performed. To find out whether the code produces sensible results, the simulation parameters of case study 4 in [2, p.97] are used. Similar but not exactly the same results are expected, since randomness exists in the algorithm and the algorithm presented here does not follow case study 4 in [2, p.97] in all details. The simulation parameters are summarized in tab. 1 below[1].

Table 1: MD Simulation Parameters - 108 Atoms - MD. # means number.

Parameter	Symbol	Setting	Units
temperature at start	T_{start}	0.728	k_B
time step	$d\tau$	0.004	τ
# atoms	N	108	#
length of simulation box	L_{box}	5.0387	σ
number density	ρ	0.8442	$1/\sigma^3$
cut-off radius	r_c	$\frac{1}{2} \cdot L_{box}$	σ
# iterations	n	$6 \cdot 10^5$	#

[1]Note that the simulation parameters are calculated by setting the number of atoms and the lattice constant in the code. I have listed the parameters explicitly here for convenience.

The atoms are initially placed on an fcc-lattice, which closely resembles Argon in it's liquid phase [3, pp. 208]. Next, the velocity components for each atom are drawn from a Maxwell-Boltzmann distribution that is centred around zero, using the built-in functionality of Python's SciPy package. The idea behind this approach is to reach the equilibrium state within a finite calculation time. The velocities could instead be drawn from a uniform random distribution and the atoms could be placed randomly in the simulation box. Drawing the velocities from the Maxwell Boltzmann distribution and placing the atoms on an fcc-lattice, however, lets the system relax towards equilibrium faster [7].

The velocities are rescaled with the given start temperature T_{start} only once, directly after their initialization. Simultaneously the sum of all velocities is set equal to zero. For the integration of equations of motion the same Velocity Verlet integrator-scheme as in the previous section is used (3.1). The kinetic, potential and total energy are sampled during each time step and saved to a .txt file. The positions and velocities are sampled every 100th time-step and saved to a separate .text file. Velocities and positions are not sampled during the first 20 τ, i.e. the equilibration time of the simulation. Moreover, time and date of the simulation as well as the simulation parameters are recorded in each file. Subsequently, the data analysis is performed in separate Python scripts using the saved .txt files.

3.3 Metropolis Monte Carlo - 108 atoms

Having completed and debugged the MD simulation, most of the code can be re-used to perform a Metropolis MC simulation. The vital adjustment in the code is to replace the Velocity Verlet Algorithm used in the MD simulation with an acceptance-rejection rule that satisfies the detailed balance criterion. Moreover the time steps of the MD simulation are replaced with random trial moves. The code written for this Metropolis MC follows the importance-sampling algorithm described in [1, pp.114]. The simulation parameters are the same as in the above MD simulation. However, the equilibrium temperature T_{equil} of the MD simulation is calculated before running

the MC simulation using:

$$T_{equil} = \frac{\langle 2E_{kin} \rangle}{f} \tag{3}$$

, where f stands for the number of degrees of freedom. The MC simulation is then run at $T_{equil} = 1.4275$ of the MD simulation. Although the MD simulation is run in the NVE ensemble, versus the NVT ensemble in the MC simulation, some sort of comparability is hoped to be achieved by doing this. The remaining simulation parameters are the same for the MD and MC simulation.

Tab. 2 below summarizes the simulation parameters of the MC simulation. Note that the number of iterations is eight times higher in the MC simulation than in the MD simulation. This is done to improve the statistics for the MC simulation somewhat (compare 4). The positions are sampled every 100th time after the system has reached equilibrium. Moreover the ratio of $\frac{acceptance}{acceptance+rejections}$ is monitored during the simulation. The trial moves are adjusted such that this ratio stays above 0.2 and below 0.3, which is recommended by both [2] and [1]. The recorded positions, energies and the accept-reject ratio are stored in separate .txt files for data analysis. Moreover, the simulation time and date as well as the simulation parameters are recorded.

Table 2: Monte Carlo Simulation Parameters - 108 Atoms.# means number.

Parameter	Symbol	Setting	Unit
temperature	T	1.4275	k_B
trial displacement	dr	0.2 * ranf(-1,1)	σ
# atoms	N	108	#
length of simulation box	L_{box}	5.0387	σ
number density	ρ	0.8442	$1/\sigma^3$
cut-off radius	r_c	$\frac{1}{2} \cdot L_{box}$	σ
# iterations	n	$48 \cdot 10^5$	#

3.4 Monte Carlo and Molecular Dynamics - 500 atoms

Having made sure that the code works correctly in the previous set-ups, both the MD and the MC simulation are repeated. The only parameter that is changed is the number of atoms in the simulation box, which is now increased to 500. The density is held fixed, which means that the simulation box becomes larger. The start temperature for the MD simulation remains the same, however the temperature at equilibrium of the MD simulation is different: $T_{equil} = 1.5648$. This is important, since the corresponding MC simulation of 500 atoms is run at this new equilibrium temperature. Due to limitations in computation time, the MC simulation for 500 atoms is only run for $6 \cdot 10^5$ iterations (instead of 4.8 million for the MC simulation of 108 atoms). Moreover, the maximum trial displacement (DJUMP in config.py) is adjusted to keep acceptance of trial moves in the desired bounds.

4 Data Analysis

4.1 Potential and Kinetic Energy

In order to estimate the error of the potential and kinetic energy measured in both MD and MD simulation, I apply the block averaging method by Flyvbjerg & Peterson (FP) described in [2, D.3]. This method determines after how many subsequently taken data samples of a quantity the data samples become de-correlated. It also determines whether the simulation has run long enough. This is achieved by plotting the standard deviation of a block versus the number of times M the blocking operation has been applied. Once the standard deviation becomes independent of M, a plateau is reached in the plot. The corresponding value of M determines the number of time steps after which the data de-correlates. This number is 2^M. If no plateau has been reached, the simulation has not run long enough [2, pp.98, D.3].

4.2 Radial Distribution Function

To calculate the radial distribution function (RDF), I adopt Algorithm 7 suggested in [2, p.86], which is based on counting the distances of atom pairs and binning the results. In the calculations below, I divide the region of interest (half the box length) into 300 bins and calculate the RDF from 4000 snapshots of the simulated system. Snapshots are taken every 100th iteration after the system has reached equilibrium. Note that in the MD and MC simulations of 500 atoms, I only use 1000 instead of 4000 snapshots to compute the RDF[2].

Allen and Tilesley [1, p.195] suggest that a working error estimate for the RDF is to apply the block averaging technique to several points of interest in the RDF. The below analysis follows this suggestion, by analysing four points in the computed RDF with the FP block averaging method. To do this, the same code as for the energy calculations can be utilized. The four points of

[2]The reason for this is simply that I didn't have time to code the routine over to C and it runs terribly slow in Python.

interests are chosen to be the peaks, minima or the tail of the corresponding RDF.

To check the calculated RDF for consistency with the above energy measurements, I calculate the potential energy per atom from eq.(4.5.1) in [2, p.100]:

$$\frac{U}{N} = 2\pi\rho \int_0^\infty \mathrm{d}r r^2 u(r) g(r) \tag{4}$$

, where u(r) is the Lennard Jones potential (including the cut-off energy correction) and g(r) is the RDF. To calculate the integral, I simply sum over all contributions the integrand gives on the interval [0, half the box length]. I thereby introduce an error that could be reduced by incorporating methods of numerical integration. However, for this crude consistency check the simple summation is sufficient.

4.3 Heat Capacity

In the NVT ensemble, the heat capacity C_V^{NVT} is related to the variance of the total energy U as:[3]

$$C_V^{NVT} = \frac{var(U)}{T^2}$$

.

Following [1, p.51], the variance in the total energy U can be divided into contributions of potential and kinetic energy. Since those contributions are not correlated, in the NVT ensemble the heat capacity is related to the fluctuations in the potential energy V:

$$var(V) = T^2 \left(C_V^{NVT} - \frac{3}{2} N \right)$$

.

Rearranging the latter equation gives the heat capacity per atom in the NVT ensemble:

[3]Note that I have set $k_B = 1$ here, to be consistent with the above introduced MD units.

$$\frac{C_V^{NVT}}{N} = \left(\frac{var(V)}{NT^2} + \frac{3}{2}\right) \tag{5}$$

Applying a similar transformation, [1, p.53] gives the following relationship between fluctuations in the kinetic energy K respectively potential energy V and heat capacity in the NVE ensemble:

$$var(K) = var(V) = \frac{3}{2}NT^2\left(1 - \frac{3N}{2C_V}\right)$$

.

Again this is rearranged to give heat capacity per atom:

$$\frac{C_V^{NVE}}{N} = \left[\frac{2}{3} - \frac{4var(K)}{9T^2N}\right]^{-1} \tag{6}$$

.

In the data analysis, the heat capacity is determined from the variance in K respectively V in sufficiently long samples. These samples must be significantly longer than the correlation time. I achieve this by sampling both, MD and MC simulations, for sufficiently long time. Then I cut the very long sample into smaller samples, where each smaller sample is significantly longer than the corresponding number 2^M found in the above FP method (4.1). The heat capacity is then determined from the mean of the computed heat capacities of the smaller samples. The error of the heat capacity is the corresponding standard deviation of the heat capacities of the smaller samples.

To check the validity of this method, I run a second MD simulation of 108 atoms, with a minimally different starting temperature $T_{start} = 0.738$ (instead of $T_{start} = 0.728$). I then calculate the heat capacity from the measured values $\frac{\Delta E}{\Delta T}$ and compare them to the values from the above calculation. In this calculation, I use Gaussian error propagation to compute the error of the heat capacity.

5 Results

The results of each of the simulations are summarized in the plots below. Values of computed quantities are summarized in tables at the end of each subsection.

5.1 Molecular Dynamics - 108 atoms

Fig. 2 and fig.3 below show the evolution of kinetic, potential and total energy during the MD simulation of 108 atoms. Fig. 2 shows the energies during the first 20 τ of the simulation, i.e. the equilibration time suggested by [3, chpt. 8]. From this it is apparent that 20 τ is indeed a sufficiently long equilibration time for this MD simulation. In fig. 3 the long time evolution of the same quantities is plotted - the interval of $6 \cdot 10^5$ steps corresponds to 2400 τ. Note that the total energy remains constant throughout the entire simulation.

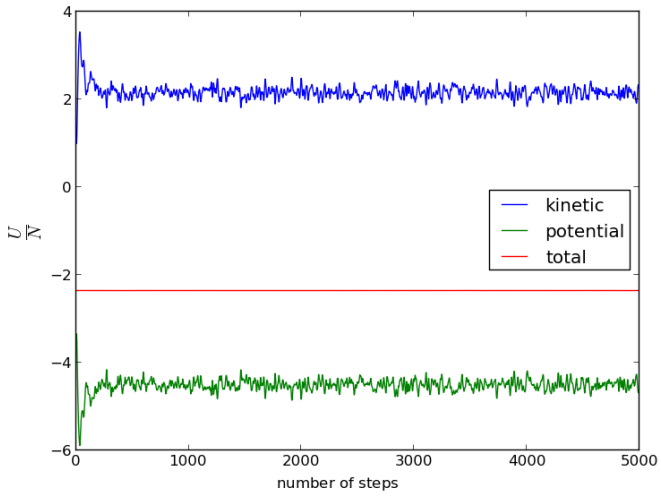

Figure 2: MD simulation, 108 atoms. Evolution of energy per atom during equilibration of the system.

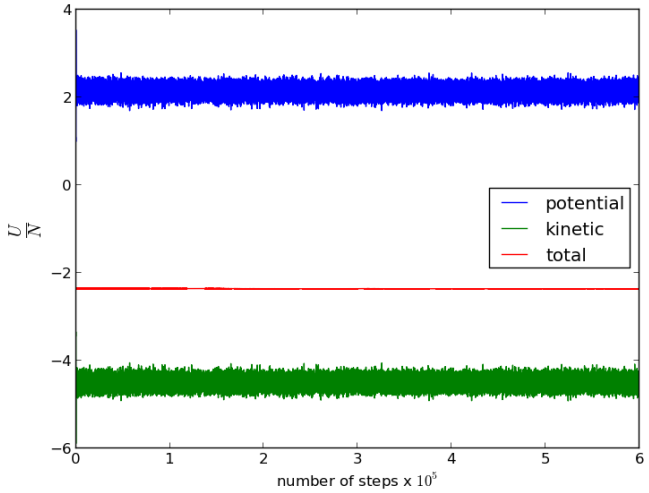

Figure 3: MD simulation, 108 atoms. Long time evolution of energy per atom.

Fig. 4 below shows the results of the FP block-averaging method for the same simulation. Both the standard deviations of the kinetic and potential energy reach the plateau after $M = 8$ blocking operations.

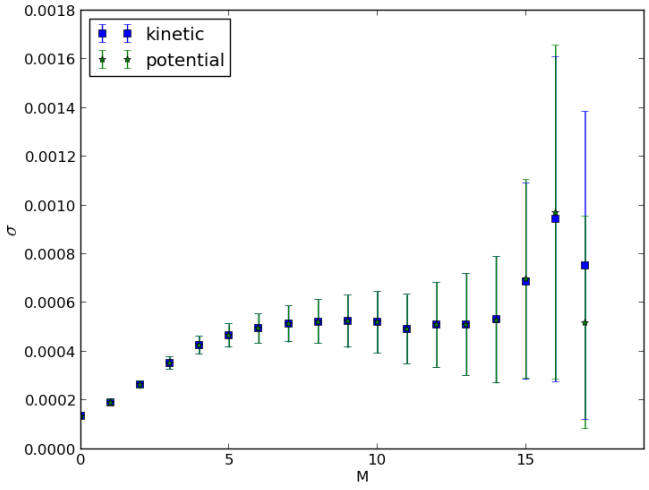

Figure 4: MD simulation, 108 atoms. Determination of standard deviation σ using the FP block-averaging method described in [2, D.3].

Fig. 15 below summarizes the results of the error calculation for the RDF. Each sub-plot shows the results of applying the FP-method to the selected data points of the RDF. The RDF itself is shown in fig. 16 below, where the RDF $g(r)$ is plotted as a function of r.

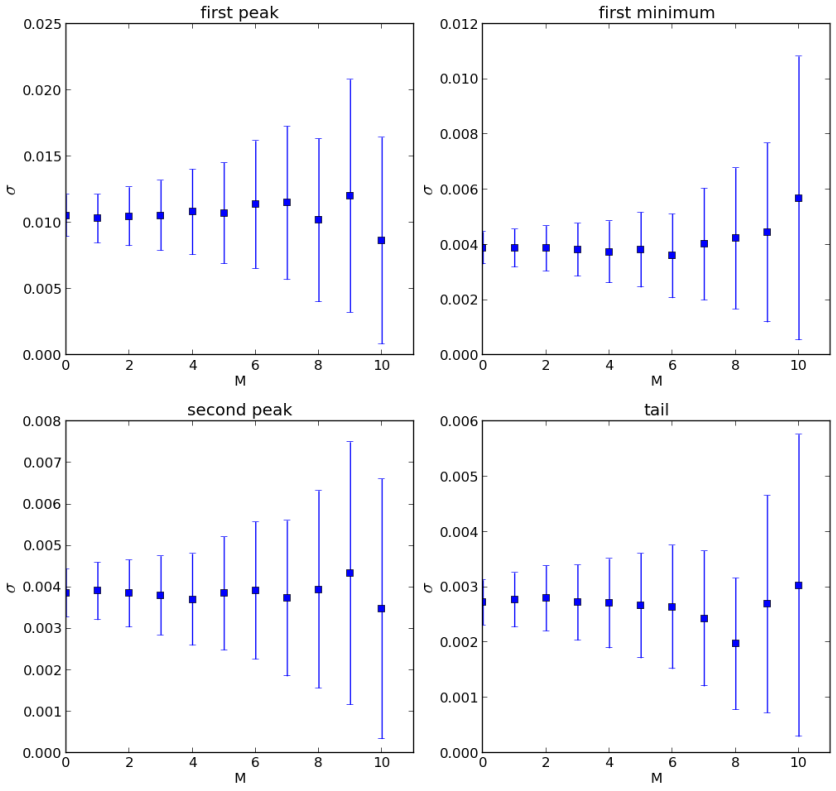

Figure 5: MD simulation, 108 atoms. Calculation of the error of selected points in the RDF $g(r)$ using the FP-method.

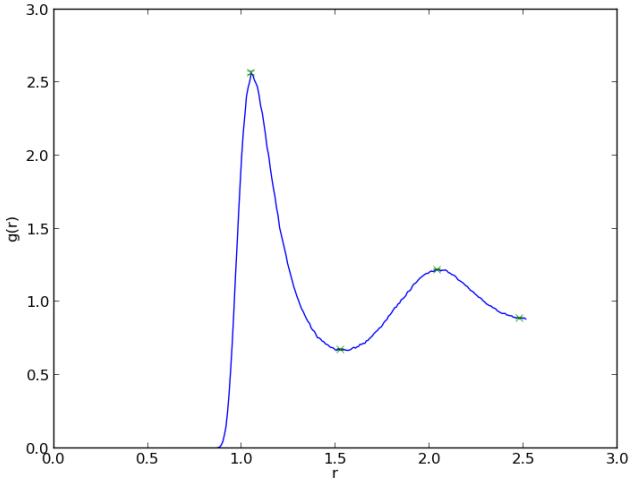

Figure 6: MD simulation, 108 atoms. Radial distribution function $g(r)$ as a function of r. The error bars are barely noticeable on the plot.

Tab. 3 below summarizes the computed values of the quantities of interest. The first column lists the method used for the calculation, the second the computed quantity, the third the values of the quantity, the fourth the corresponding error and the fifth column the corresponding units.

Table 3: MD Simulation Results - 108 Atoms.

Method Used	Quantity	Computed Value	Computed Error	Units
FP	E_{kin}	2.1412	$\pm$0.0005	ϵ
FP	E_{pot}^{avrg}	-4.5073	$\pm$0.0005	ϵ
eq. 4	E_{pot}^{rdf}	-4.5055	-	ϵ
eq. 3	T_{equi}	1.4275	$\pm$0.0004	ϵ/k_B
sampling	C_V^{NVE}/N	2.41	$\pm$0.05	k_B
changing T_{start}	C_V^{NVE}/N	2.4	$\pm$0.3	k_B

5.2 Monte Carlo - 108 atoms

Fig. 7 and fig.8 below show the evolution of the potential energy during the MC simulation of 108 atoms. Fig. 7 shows the energies during the first 10^5 steps of the simulation. In fig. 3 the long time evolution of the potential energy is plotted The accept-reject ratio is analysed from the data file: it stays between 0.2 and 0.3.

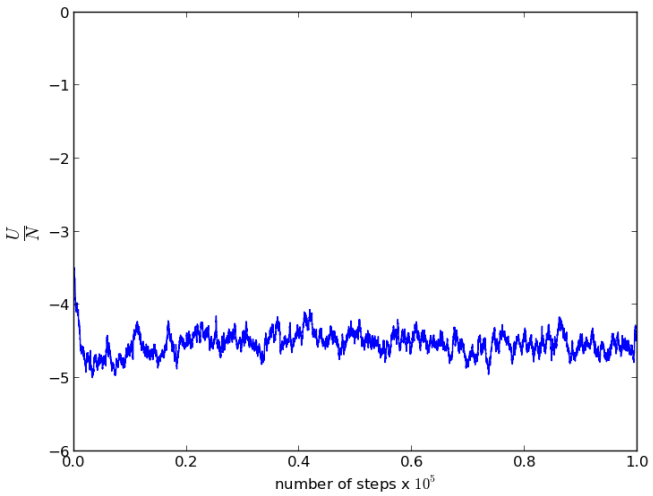

Figure 7: MC simulation, 108 atoms. Evolution of the potential energy during the first 10^5 steps.

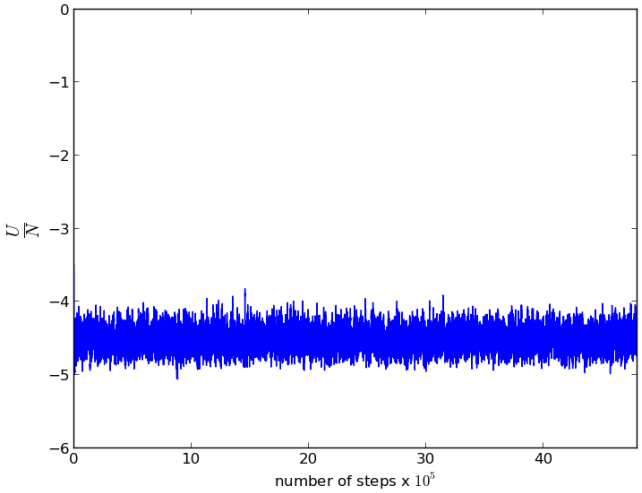

Figure 8: MC simulation, 108 atoms. Evolution of potential energy for entire simulation length.

Fig. 9 below shows the results of the FP block-averaging method for the same simulation. The standard deviations of the potential energy reaches the plateau after $M = 16$ blocking operations.

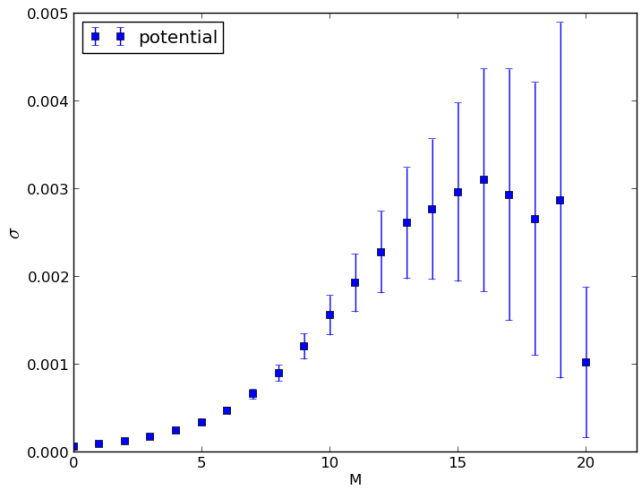

Figure 9: MC simulation, 108 atoms. Determination of standard deviation σ using the FP block-averaging method described in [2, D.3].

Fig. 20 below summarizes the results of the error calculation for the RDF. Each sub-plot shows the results of applying the FP-method to the selected data points of the RDF. The RDF itself is shown in fig. 21 below, where the RDF $g(r)$ is plotted as a function of r.

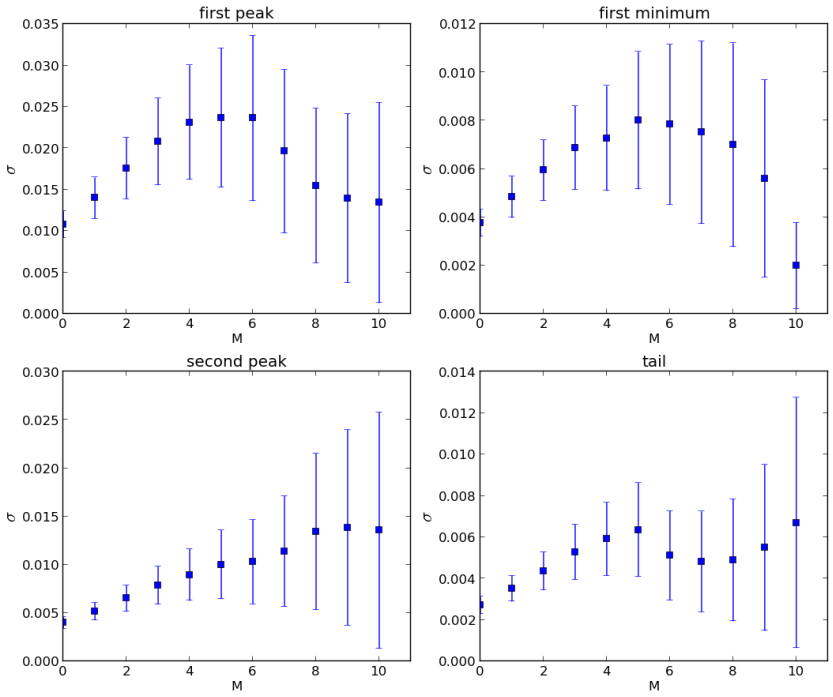

Figure 10: MC simulation, 108 atoms. Calculation of the error of selected points in the RDF $g(r)$ using the FP-method.

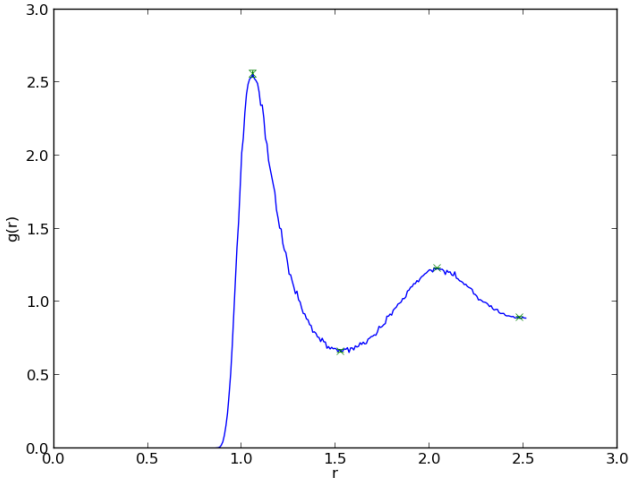

Figure 11: MC simulation, 108 atoms. Radial distribution function $g(r)$ as a function of r. The error bars are barely noticeable.

Tab. 4 below summarizes the computed values of the quantities of interest for the MC simulation of 108 atoms. The first column lists follow the same logic as tab. 3 of the last section.

Table 4: MC simulation results - 108 atoms

Method Used	Computed Quantity	Computed Value	Error	Unit
FP	E_{pot}^{avrg}	-4.5199	$\pm$0.0029	ϵ
eq. 4	E_{pot}^{rdf}	-4.5198	-	ϵ
sampling	C_V	2.40	$\pm$0.09	k_B

5.3 Molecular Dynamics - 500 atoms

Fig. 12 and fig.13 below show the evolution of kinetic, potential and total energy during the MD simulation of 500 atoms. Fig. 12 shows the energies during the first 20 τ of the simulation. In fig. 13 the long time evolution of the same quantities is plotted.

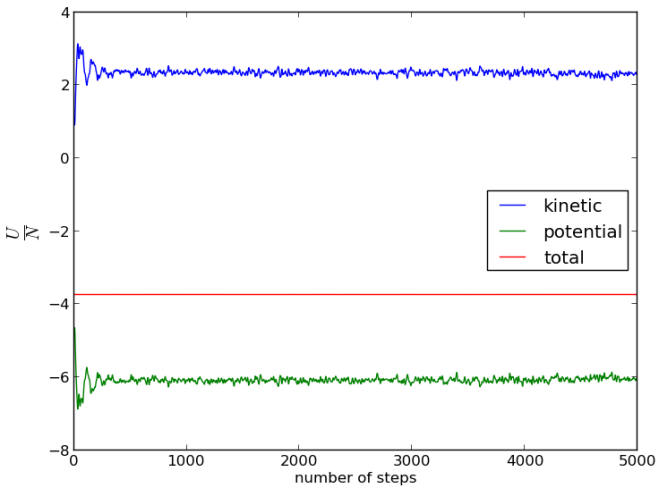

Figure 12: MD simulation, 500 atoms. Evolution of energy per atom during equilibration of the system.

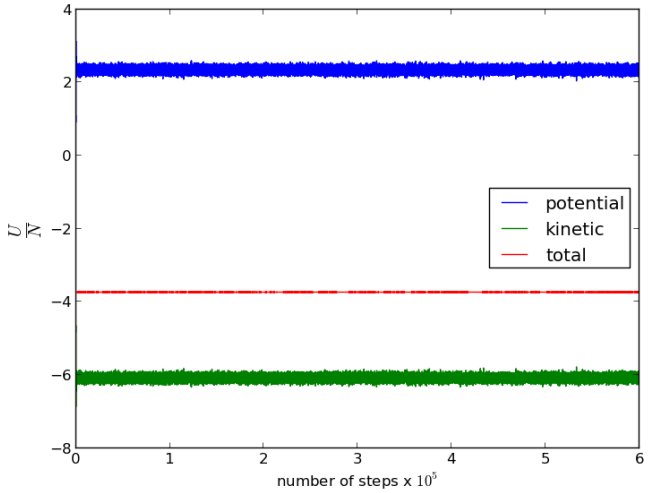

Figure 13: MD simulation, 500 atoms. Long time evolution of energy per atom.

Fig. 14 below shows the results of the FP block-averaging method for the same simulation. Both the standard deviations of the kinetic and potential energy reach the plateau after $M = 5$ blocking operations.

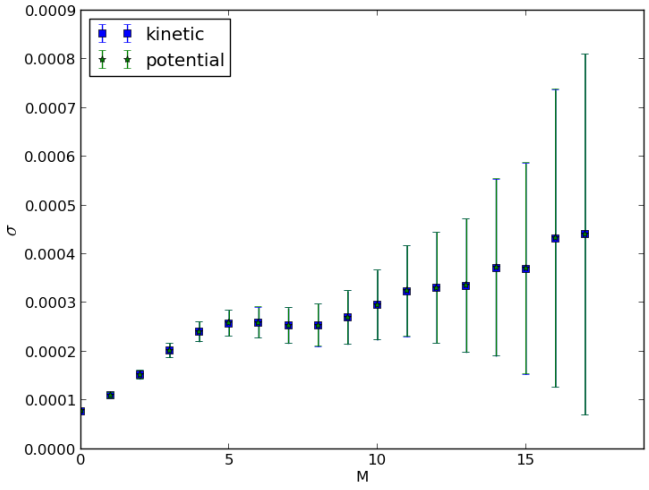

Figure 14: MD simulation, 500 atoms. Determination of standard deviation σ using the FP block-averaging method described in [2, D.3].

Fig. 15 below summarizes the results of the error calculation for the RDF. Each sub-plot shows the results of applying the FP-method to the selected data points of the RDF. The RDF itself is shown in fig. 16 below, where the RDF $g(r)$ is plotted as a function of r.

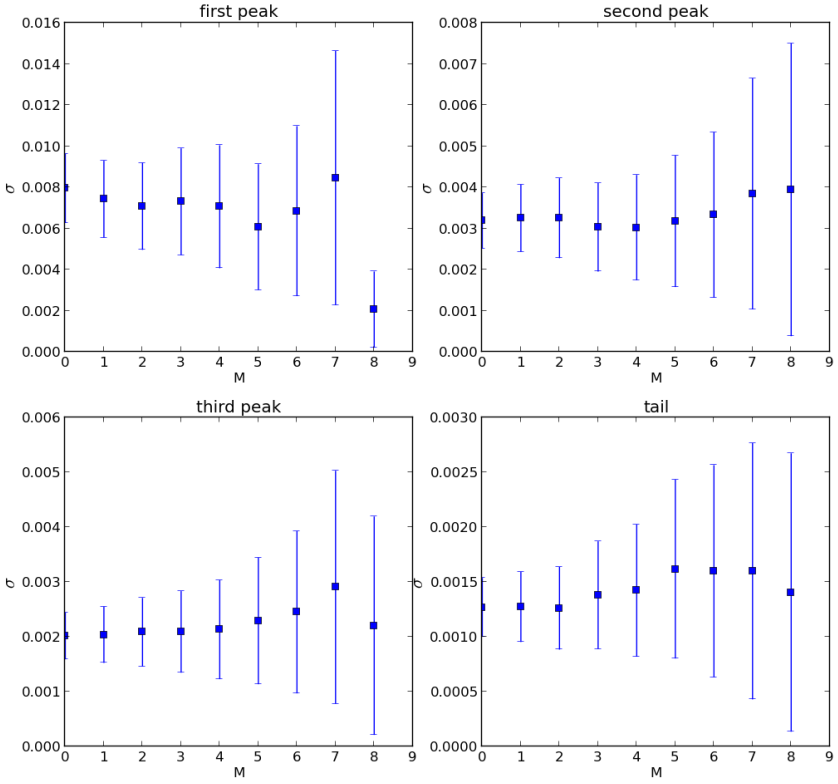

Figure 15: MD simulation, 500 atoms. Calculation of the error of selected points in the RDF $g(r)$ using the FP-method.

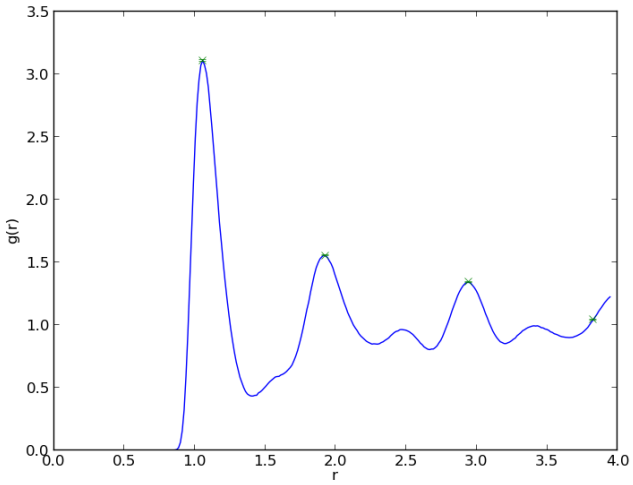

Figure 16: MD simulation, 500 atoms. Radial distribution function $g(r)$ as a function of r. The error bars are barely noticeable.

Tab. 5 below summarizes the computed values of the quantities of interest in the same fashion as in the preceding sections.

Table 5: MD Simulation Results - 500 Atoms.

Method Used	Quantity	Computed Value	Computed Error	Units
FP	E_{kin}	2.3472	$\pm$0.0003	ϵ
FP	E_{pot}^{avrg}	-6.0800	$\pm$0.0003	ϵ
eq. 4	E_{pot}^{rdf}	-6.0729	-	ϵ
eq. 3	T_{equi}	1.5678	$\pm$0.0003	ϵ/k_B
sampling	C_V^{NVE}/N	2.91	$\pm$0.07	k_B

5.4 Monte Carlo - 500 atoms

Fig. 17 and fig.18 below show the evolution of the potential energy during the MC simulation of 500 atoms. Fig. 17 shows the energies during the first 10^5 steps of the simulation. In fig. 13 the long time evolution of the potential energy is plotted. The accept-reject ratio is analysed from the data file: it stays between 0.2 and 0.5.

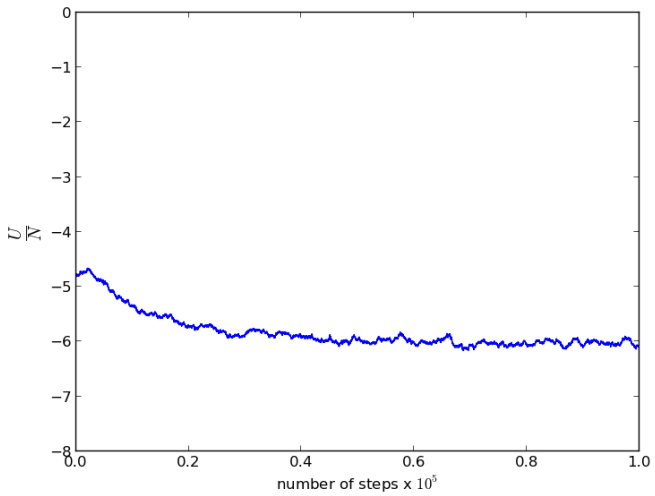

Figure 17: MC simulation, 500 atoms. Evolution of the potential energy during the first 10^5 steps.

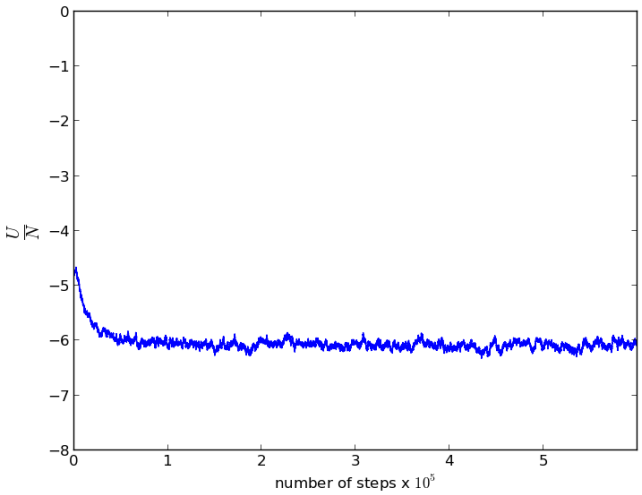

Figure 18: MC simulation, 500 atoms. Evolution of potential energy for entire simulation length.

Fig. 19 below shows the results of the FP block-averaging method for the same simulation. The standard deviations of the potential energy reaches the plateau after $M = 15$ blocking operations.

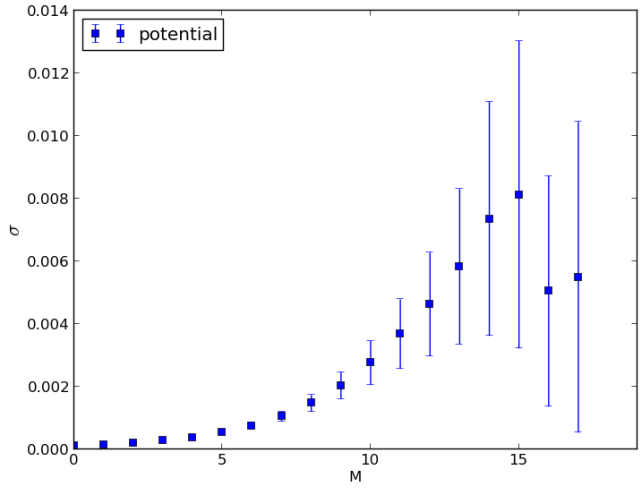

Figure 19: MC simulation, 500 atoms. Determination of standard deviation σ using the FP block-averaging method described in [2, D.3].

Fig. 20 below summarizes the results of the error calculation for the RDF. Each sub-plot shows the results of applying the FP-method to the selected data points of the RDF. The RDF itself is shown in fig. 21 below, where the RDF $g(r)$ is plotted as a function of r.

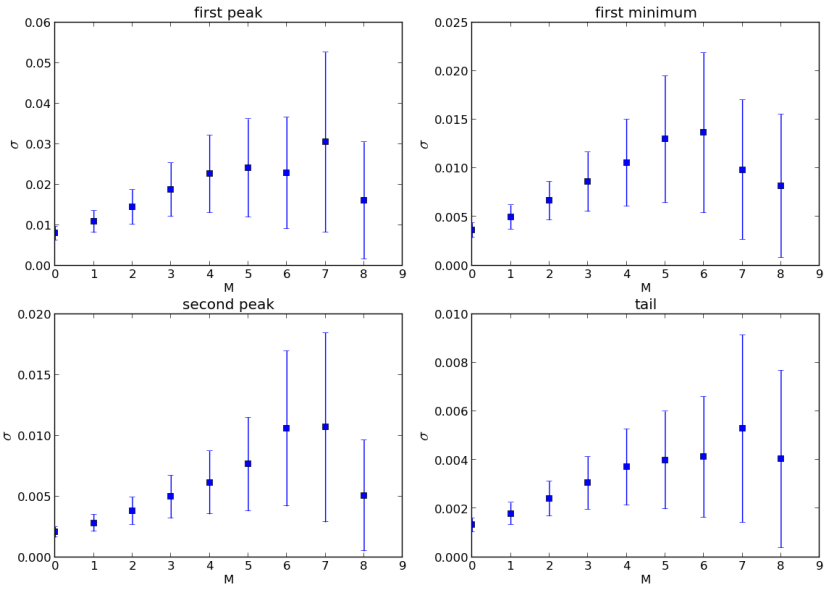

Figure 20: MC simulation, 500 atoms. Calculation of the error of selected points in the RDF $g(r)$ using the FP-method.

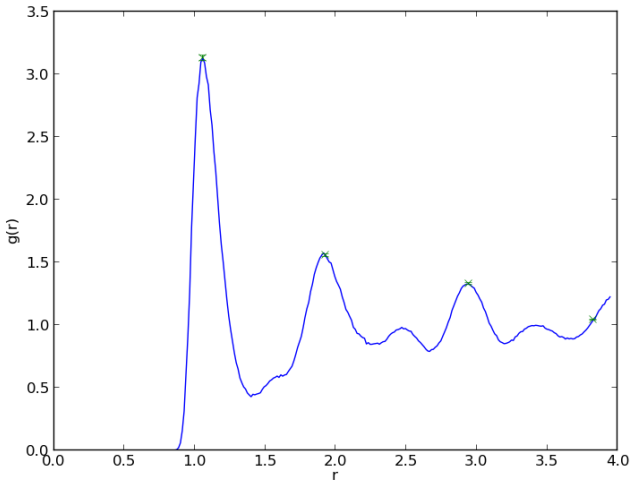

Figure 21: MC simulation, 500 atoms. Radial distribution function $g(r)$ as a function of r. The error bars are barely noticeable.

Tab. 6 below summarizes the computed values of the quantities of interest for the MC simulation of 500 atoms. The first column lists follow the same logic as tab. 5 of the last section.

Table 6: MC simulation results - 500 atoms

Method Used	Computed Quantity	Computed Value	Error	Unit
FP	E_{pot}^{avrg}	-6.0588	$\pm$0.0081	ϵ
eq. 4	E_{pot}^{rdf}	-6.0852	-	ϵ
sampling	C_V	2.8	$\pm$1.6	k_B

6 Discussion

In the MD simulation of 108 atoms the two independent methods of computing the potential energy deliver results that are in good agreement with each other. The same holds for the two independent methods used for computing the heat capacity in the same simulation. Moreover, the computed radial distribution function is in good agreement with case study 4 in [2, p.101]. This demonstrates that the MD algorithm can be trusted.

Comparing the results of the MD simulation of 108 atoms with those of the corresponding MC simulation, it is important to realize that the former simulation runs in the NVE ensemble whereas the latter in the NVT ensemble. This limits the comparability to some extend. However, the computed potential energy, heat capacity and radial distribution function of the MC simulation are in good agreement with the MD simulation. This in turn, shows that the MC algorithm is trustworthy.

Comparing MD and MC simulations further, it is apparent from the energy plots that energy fluctuations are bigger in the MC than in the MD simulation. This translates directly into an error for the potential energy that is an order of magnitude larger than the corresponding error in the MD simulation, caused by the fact that the potential energy needs longer to reach the plateau in the FP method than in the MC than in the MD simulation.

However this does not translate into a difference in order of magnitude in the errors of the heat capacity in both simulation. That may be an indication for not having chosen the sample size adequately enough, i.e. too long samples in the MD simulation and/or too short samples in the MC simulation. This has been investigated by recalculating the heat capacity in both cases with various sample sizes. However, no significant changes in the value of the error have been observed.

The results of the MD and MC simulations for 500 atoms are in good agreement with each other. However, the error in the heat capacity in the MC simulation is unacceptably high. The explanation for such a high error is the relatively short simulation time. One could argue from fig. 19 that the plateau in the FP method is not reached during the simulation. This, in

turn means that the samples used in the computation of the heat capacity are not long enough.

Another interesting aspect of discussion is the smoothing of the RDF. Through varying the number of bins and the number of samples used to calculate the RDF, different degrees of smoothing can be achieved. This topic is explored in depth in [1].

7 Conclusions

Stable Monte Carlo and Molecular Dynamics simulations of the Lennard-Jones Argon liquid are coded and produce consistent results for the quantities of interest, such as energies and heat capacity. Moreover, Python proves to be a versatile tool for crafting such simulations.

References

[1] M.P. Allen, D.J. Tildesley, 1987. *Computer Simulation of Liquids*. Oxford (UK): Oxford University Press.

[2] D. Frenkel, B. Smit, 2002. *Understanding Molecular Simulation - From Algorithms to Applications*. New York: Academic Press.

[3] J.M. Thijssen, 2007. *Computational Physics*. 2nd edition. Cambridge (UK): Cambridge University Press.

[4] SciPy. *Performance Python*. Available at `http://www.scipy.org/PerformancePython`. Accessed on 10.02.2013.

[5] StackOverflow. *Improve Nested Loop Performance*. Available at: `http://stackoverflow.com/questions/14855321/improve-nested-loop-performance`. Accessed on 13.02.2013.

[6] Stockholm University. *Programming Projects*. Available at: `http://www.syonax.net/physics/SimMethStatPhys/projects.pdf`. Accessed on 01.02.2013.

[7] University at Buffalo, Physics Department. *Statistical Physics*. Available at: `http://www.physics.buffalo.edu/phy410-505/2011/topic5/app2/index.html`. Accessed on 05.02.2013.