Worksheet 4: Error Analysis and Langevin Thermostat

Olaf Lenz

December 2, 2014
Institute for Computational Physics, University of Stuttgart

Contents

1 General Remarks

- Deadline for the report is **Monday, 12th January 2015, 10:00 a.m.**.
- In this worksheet, you can achieve a maximum of 20 points.
- The report should be written as though it would be read by a fellow student who attends to the lecture, but doesn't do the tutorials.
- To hand in your report, send it to your tutor via email
  - Florian (`fweik@icp.uni-stuttgart.de`) (Wednesday 15:45-17:15)
  - Johannes (`zeman@icp.uni-stuttgart.de`) (Wednesday 17:30-19:00)
- Please attach the report to the email. For the report itself, please use the PDF format (we will *not* accept MS Word doc/docx files!). Include graphs and images into the report.

- If the task is to write a program, please attach the source code of the program, so that we can test it ourselves.
- The report should be 5–10 pages long. We recommend using LaTeX. A good template for a report is available online.
- The worksheets are to be solved in groups of two or three people.

On this worksheet, you will finally complete the MD simulations of the Lennard-Jones (LJ) System and do error analysis on the produced data. In the first part, you will develop functions to perform error analysis of time series data, and you will apply them to artificial datasets with a defined error. In the second part, you will first implement the Langevin thermostat for the LJ simulation, then perform some simulation runs and finally analyze the results using the error analysis functions from the first part.

All files required for this tutorial can be found in the archive `templates.tar.gz` which can be downloaded from the lecture's homepage.

2 Error Analysis

From the lecture's home page, you can download the file `janke02.pdf` of an article by Wolfhard Janke. Section 3 of this article is what the lecture on error analysis was based on, and all that needs to be done in this part of the worksheet can be learned from the article.

The data file `series.dat` contains a pickled (100000×5)-NumPy-Array with 5 time series of 100000 values each. Each time series has a different mean value and correlation time. Write a Python program that reads the data via pickle and plots the first 1000 points of the series to get a feeling for the data.

2.1 Autocorrelation Analysis

Task (3 points)

- Implement a Python function that computes the autocorrelaton function of a given time series.
- Compute the autocorrelation function of the data sets.
- Plot the autocorrelation function of the datasets.
- Plot the integrated autocorrelation function. The function should converge against the autocorrelation time.
- Can you guess the autocorrelation times from the plots?
- Why is it not useful to integrate the function over the whole interval where it can be computed to determine the autocorrelation time?

Task (3 points)

- Implement a Python function that performs automatic error analysis via autocorrelation analysis of a given time series of an observable.
- The function should compute an estimate for the autocorrelation time via the running autocorrelation time estimator (eq. 33 in Janke's article), where you cut off the integration when $k_{\max} \geq 6\hat{\tau}_{\mathcal{O},\text{int}}(k_{\max})$.
- The function should return:
  - the observable's mean value $\bar{\mathcal{O}}$
  - the error $\epsilon_{\bar{\mathcal{O}}}$ of the observable's mean value
  - the estimated autocorrelation time $\tau_{\mathcal{O},\text{int}}$
  - the error $\epsilon_{\tau_{\mathcal{O},\text{int}}}$ in the autocorrelation time
  - the effective statistics N_{eff}
- Apply the error analysis function to the sample datasets and include the results in your report. How do the results compare to your guesses from the plots?

2.2 Binning Analysis

While the autocorrelation analysis of the previous task can do its work automatically, binning analysis can only work semiautomatically, as one has to visually determine the autocorrelation time from the binning.

Task (3 points)

- Implement a Python function that computes the block variance σ_B^2 for a given block size k.
- From the block variance, one can compute an estimate for the autocorrelation time $\tau_{\mathcal{O},\text{int}} = \frac{1}{2}k\sigma_B^2/\sigma_{\mathcal{O}_i}^2$ (a.k.a. "blocking τ").
- Furthermore, one can use the block variance to estimate the error of the mean value $\epsilon_{\bar{\mathcal{O}}}^2 = \sigma_B^2/N_B$.
- Plot the blocking τ and the error estimate for $1 < k < 2000$ for the different datasets.
- Can you guess the autocorrelation time and the error of the mean value of the datasets from the plots?
- Do they match the values obtained via autocorrelation analysis?

2.3 Jackknife Analysis

Jackknife analysis for this kind of data produces identical results to the binning analysis.

Task (3 points)

- Implement a Python function that computes the Jackknife error $\epsilon^2_{\bar{\mathcal{O}}}$ of a time series for a given block size k.
- Plot the Jackknife error estimate for $1 < k < 2000$ for the different datasets.
- Can you guess the error of the mean value of the datasets from the plots?
- Do they match the values obtained via the above methods?

3 Langevin Thermostat

As we have seen on the last worksheet, even though the *velocity rescaling thermostat* is able to keep the temperature constant, this is not actually the same as simulating the canonical (N, V, T)-ensemble, as the Maxwell distribution of the velocities is destroyed.

A thermostat that does allow to simulate the canonical ensemble is the *Langevin thermostat*. In the Langevin thermostat, at each time step every particle is subject to a random (stochastic) force and to a frictional (dissipative) force. There is a precise relation between the stochastic and dissipative terms, which comes from the so-called fluctuation-dissipation theorem, and ensures sampling of the canonical ensemble. The equation of motion of a particle is thus modified:

$$m\mathbf{a}_i = \mathbf{F}_i - m\gamma\mathbf{v}_i + \mathbf{W}_i(t) \tag{1}$$

Here γ is a friction coefficient with unit t^{-1} and $\mathbf{W}_i$ is a random force acting on particle i that is uncorrelated in time and between particles, and is characterized by its variance:

$$\langle \mathbf{W}_i(t) \cdot \mathbf{W}_j(t') \rangle = \delta_{ij}\delta\left(t - t'\right) 6k_B m T\gamma \tag{2}$$

The modified Velocity-Verlet algorithm for a Langevin thermostat is as follows

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t)(1 - \Delta t\frac{\gamma}{2}) + \frac{\Delta t^2}{2m}\mathbf{G}_i(t) \tag{3}$$

$$\mathbf{v}_i(t + \Delta t) = \frac{\mathbf{v}_i(t)(1 - \Delta t\frac{\gamma}{2}) + \frac{\Delta t}{2m}(\mathbf{G}_i(t) + \mathbf{G}_i(t + \Delta t))}{(1 + \Delta t\frac{\gamma}{2})} \tag{4}$$

where $\mathbf{G}_i$ is the total force: $\mathbf{G}_i = \mathbf{F}_i + \mathbf{W}_i$.

Task (3 points)

- Derive Eq. (4) using Eq (1). Remember that, according to the Verlet algorithm (see worksheet 1), we have

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{a}_i(t) + \mathbf{a}_i(t + \Delta t)}{2m} \Delta t.$$

- Extend the Lennard-Jones simulation program that you created in the last worksheet and add the Langevin thermostat. Have a look at worksheet 1 if you do not remember how to implement the Velocity Verlet equations. If you are not confident with your own simulation program or it wasn't complete, you can find the sample solution of the last worksheet in the archive file in the subdirectory `ljsim/`.
- Did you implement the thermostat in the Python- or in the C-part of the program? Explain why you did it in the one and not the other part!

Task (2 points)

- Run simulations of $N = 1000$ LJ particles at density $\rho = 0.316$ and temperatures $T \in \{0.3, 1.0, 2.0\}$ using the Langevin thermostat ($\gamma = 0.5$).
- Determine the equilibrium mean values of the pressure P and the potential energy *per particle* E_{pot}.

4 Combining Everything: Error Analysis of Real Simulation Data

Task (3 points)

- Extend the program `ljanalyze.py` to include the error analysis functions.
- Analyze the errors of the equilibrium values of the observables that you have determined in the previous task.
- Extend the simulation runs such that the error of the observables is be less than 5%. Use any of the error estimates to determine the error.
- How many simulation steps do you need to achieve this error? What is the effective statistics of the sample?