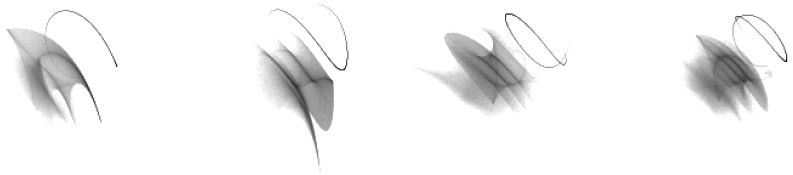
Theory, codes, and numerical simulation of heat transport in multicomponent systems

Candidate:
Riccardo Bertossa

Advisor:
Stefano Baroni

Academic Year 2021-2022



Open-source

All the codes used by and written for this work by the author are open-source and freely available.

https://github.com/rikigigi
https://github.com/sissaschool/sportran
https://github.com/aiidateam/aiida-quantumespresso
https://gitlab.com/QEF/q-e

Contents

List of published works:

- Riccardo Bertossa, Federico Grasselli, Loris Ercole, and Stefano Baroni. Theory and numerical simulation of heat transport in multicomponent systems. *Phys. Rev. Lett.*, 122(25):255901, June 2019. arXiv:1808.03341, doi:10.1103/physrevlett.122.255901

- Stefano Baroni, Riccardo Bertossa, Loris Ercole, Federico Grasselli, and Aris Marcolongo. Heat transport in insulators from ab initio green-kubo theory. In Wanda Andreoni and Sidney Yip, editors, *Handbook of Materials Modeling*, pages 809–844. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-319-44680-6_12

- Aris Marcolongo, Riccardo Bertossa, Davide Tisi, and Stefano Baroni. QEHeat: An open-source energy flux calculator for the computation of heat-transport coefficients from first principles. *Comput. Phys. Commun.*, 269:108090, December 2021. doi:10.1016/j.cpc.2021.108090

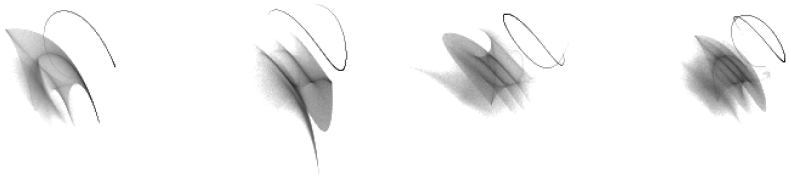
- Davide Tisi, Linfeng Zhang, Riccardo Bertossa, Han Wang, Roberto Car, and Stefano Baroni. Heat transport in liquid water from first-principles and deep neural network simulations. *Phys. Rev. B*, 104(22):224202, December 2021. doi:10.1103/physrevb.104.224202

- Loris Ercole, Riccardo Bertossa, Sebastiano Bisacchi, and Stefano Baroni. Sportran: A code to estimate transport coefficients from the cepstral analysis of (multivariate) current time series. *Computer Physics Communications*, page 108470, 2022. doi:10.1016/j.cpc.2022.108470

Abstract

Heat transport is a topic that is fundamental in many fields, from materials engineering to planetary models. The calculation of the thermal transport coefficient with the Green-Kubo theory in multicomponent fluids, especially in ab-initio simulations, had a severe data analysis issue that this work solved. In this thesis, we derive the entire theory and data analysis framework for the multicomponent Green-Kubo. Then we show the computer codes we developed, allowing the user to apply the approach previously derived. We believe that in science, replicability and reproducibility are essential requirements. Every new technique must come with an open-source and reliable implementation.

In the end, we demonstrate a significant application to superionic ammonia, fundamental to understanding the behavior of icy giant planets like Uranus and Neptune, providing an estimate for the thermal transport coefficient.



Introduction

I often find myself staring at the surface of a hot cup of tea, where lots of dancing figures appear, and the energy flows away from the tea to the air, an irreversible process like the flow of time. And then, I start thinking of the incredible complexity of the cup of tea system at the atomic level. Many different molecules, different water phases, a vast number of degrees of freedom, and the complexity of quantum interactions. How can we go from a microscopic description of the atoms in a cup of tea to the irreversible macroscopic phenomena of heat dissipation that we experience every day? Diffusion of energy emerges somehow from the statistic of an uncountable number of atoms interacting together. But to predict the evolution of the macroscopic system, knowing how the atoms move is not necessary. For example, Fourier's work on heat transport, culminating in the book *Théorie analytique de la chaleur* and the equation $\mathbf{J} = -\kappa\nabla T$, was published in 1827, well before any quantum mechanical theory was able to predict properties of materials. It says that the heat flow is a vector proportional to the temperature gradient and goes from hot to cold places. The fact that all the incredible complexity of heat transport at the microscopic scale is hidden inside a simple law and the difference between each material is described by a single number, κ, is something miraculous. κ can be measured, most of the time, with simple experiments where a difference of temperature is forced and heat flow or relaxation time are measured. But if we know the laws of the microscopic atomic interaction, why should we not be able to compute all macroscopic properties of my cup of tea starting from scratch, with only a computer and all the theory of condensed matter that is known up today?

The ability to build a functioning thermal model of a system of interest without needing to perform experiments can be used in many different fields with a broad range of applications, like planetary modeling and materials engineering. And in a few cases performing a laboratory experiment is unfeasible. When this happens, a framework for computing the thermal transport coefficient entirely ab-initio is strictly necessary, otherwise one cannot get any meaningful number to build any model and cannot advance with new knowledge in the field.

In this work, we'll focus on computing the thermal conductivity coefficient κ in systems with a band gap using a computer and the Green-Kubo theory of linear response, which can be roughly summarized in the expression

$$\kappa \propto \int_0^\infty \langle \mathbf{J}(t) \cdot \mathbf{J}(0) \rangle dt \ , \tag{1}$$

where $\mathbf{J}$ is the heat current, and the brackets indicate equilibrium ensemble averages which are accessible to molecular dynamics simulations.

Despite the familiarity and the simple phenomenological equations that describe heat flow, the microscopic definition of $\mathbf{J}$ (something that can be used in a computer experiment) is somewhat blurred since atomic energy – the microscopic thing most similar to heat – is a quantity that is not unique. Let's make a macroscopic example. Suppose we have two boys who stretch the same giant elastic. If we pretend to divide the energy of the system (the work that the boys did to stretch the elastic) into "boys' energy", we don't have any unique way of doing this. We can say that all the energy stays within a boy or the other boy. No physical principle tells us how to divide energy. If we write the system's equation of motion with all those different definitions, it makes no difference. The energy is somehow hidden inside the elastic, inside the interaction between the different parts of the system. The number of ways in which the interaction energy between any group of

atoms can be divided is infinite: the only constraint is that the sum of all atomic energies (arbitrarily defined) is equal to the system's total energy. But then, how is it possible that we experience the locality of the heat and energy at our macroscopic level? For example, we can tell how much heat is transferred inside the cup of tea by using a calorimeter. This happens because, at some point, when we move far enough from the microscopic interacting structure – farther than the length of the biggest elastic – we can group local energies of a big enough region of space. If we sum them all up, the choice of how we decided to divide the energy of the interacting boundaries of the region counts less and less, leading to a well-defined number for each region of space. And we can measure the differences between those numbers in the form of transferred heat. But let's take a step back to our microscopic world, the one we can afford with a computer.

What guarantees us that it is possible to compute a meaningful number with arbitrary quantities? A milestone in this field was the work of Marcolongo, Umari, and Baroni[50] in 2016 that shows the possibility of computing a well-defined thermal conductivity ab-initio, even though the local microscopic energy density is not unique. Note that in first principle simulations, the situation is complicated because the theory does not use anything similar to an elastic to describe interactions, but it uses a function defined in all points of the space that the system occupies. This work allowed our group to compute thermal conductivity in all systems with a band gap where there is only one type of molecule, for example, water or in a solid. Still, the theory was not yet ready for a multicomponent fluid system like molten salt. A considerable data analysis effort was made to solve many issues that a Green-Kubo theory with a density functional theory energy current has. The very big amount of energy that the pseudopotential (an arbitrary artifact) carries with it puts a lot of noise in the calculation. Still, it does not contribute to thermal conductivity. The generalization of the framework to multicomponent fluids was not clear at that time. This is the point where the work of the author starts.

The first part of the research concerned the derivation of the multicomponent generalization of all parts of the theory and the data analysis framework[7, 5]. We found a theory that is both mathematically beautiful and useful, also for the cases that were already covered by the previous theory, making the data analysis much simpler.

The second important part of the work involved computer codes. Good open-source codes (with tests and documentation) are needed so the community can use all the wisdom generated by years of experience in the field with few clicks or commands. The code for computing the DFT energy flux with Quantum ESPRESSO was renovated entirely, a fact that allowed us to merge it into the Quantum ESPRESSO Suite[48]. The code for the data analysis was updated with the new progress made by the author in the theory and was made user-friendly, with a graphical user interface and an easy-to-use command line tool[18]. We built a code to efficiently analyze the biggest trajectories. Then we built an interesting code for a semi-automatic `AiiDA` [35, 71] workflow to get easily Car-Parrinello trajectories with minimal effort. A significant amount of time was also spent on implementing features, fixing bugs, and porting part of Quantum ESPRESSO's `cp.x` code to a GPU architecture.

The last part of the work was an application of all the tools developed by the author to investigate the properties of superionic ammonia. This investigation has important applications in planetary science. In this field, we have very few alternatives to doing computer experiments to know the properties of very high-pressure and temperature systems.

1

Theory of thermal transport in insulators

In this chapter, we'll review the basics of linear response theory in molecular dynamics and the Green-Kubo expressions for the thermal transport coefficient κ in a multicomponent fluid. We'll focus on the one obtained when the mass currents $\mathbf{J}^i$ are zero, at the steady state of the experiment. This expression has some invariances that allow great flexibility in the currents that we can use.

1.1 Green-Kubo theory

1.1.1 Hydrodynamics variables

Let's take a macroscopic system at thermodynamical equilibrium. When a little perturbation is applied fluxes appear, moving atoms and energy from regions of space to other contiguous regions, trying to approach a new equilibrium state without violating conservation rules. For example, if a temperature difference is present in a solid, energy will start to flow from the hottest to the coldest part, trying to reach a new equilibrium temperature common to all parts of the system. In this case, we don't have any flow of mass, but heat flows by conduction. We can have the same situation in a fluid, bringing also a mass flow into the play. A theory that describes transport phenomena must contain densities, thermodynamic forces that can create gradients of densities, and currents. It is natural to use conserved quantities, which allows us to define local densities, local currents, and continuity equations. Then constitutive and thermodynamics relations are needed to close the equations and be able to fully describe the system and predict its behavior.

An isolated system has the following conserved quantities that we can use: total energy, total momentum, and total mass. We can write for each of those the continuity equation:

$$\frac{\partial a^i(\mathbf{r},t)}{\partial t} = -\nabla \cdot \boldsymbol{j}^i(\mathbf{r},t), \tag{1.1}$$

where a^i is the density of the conserved quantity i and is valued over all points of space and time, and $\boldsymbol{j}^i(\mathbf{r},t)$ is a vector field that is the current of the conserved quantity. The

integral of each density a^i on the system volume is an extensive conserved quantity A^i:

$$A^i[\Omega] = \int_\Omega a^i(\mathbf{r})d\mathbf{r}, \tag{1.2}$$

where we integrated over all the system volume Ω.

Inside the system, many different phenomena occur on very different timescales and different scales of length. For example, the mass density at the microscopic atomic scale has many fast and big fluctuations with peaks on atomic nuclei positions, but we are not interested in these fluctuations when looking at the system on the scale of a planet. Nor these fast fluctuations do influence the dynamics of the system at a bigger scale, in the sense that if a single atom moves in a different direction, given that all the atoms always follow the same statistics brings no change on the bigger time and space scale. So we can rethink the (1.1) in the reciprocal space, where we can easily separate the fast and the slow degrees of freedom:

$$\dot{\tilde{a}}(\mathbf{k}, t) = -i\mathbf{k} \cdot \tilde{\boldsymbol{\jmath}}(\mathbf{k}, t), \tag{1.3}$$

For small k the time derivative of the density will be smaller and smaller, and for $k = 0$ it will be zero since a is the density of a conserved quantity in an isolated system. The long wavelength components of conserved densities are what we call hydrodynamic variables. They are adiabatically decoupled from all the other fast degrees of freedom of the system.

Suppose that we want to study the evolution of the system with these variables. Then we need more expressions to link the densities to the currents, and parameters to characterize the system's properties. If the system is not far from its equilibrium configuration, we can assume a relation like the following for the time derivative of a^i:

$$\dot{a}^i(\mathbf{r}, t) = \sum_j \int d\mathbf{r}' \int dt' \Lambda^{ij}(\mathbf{r} - \mathbf{r}', t - t')a^j(\mathbf{r}', t') \tag{1.4}$$

that becomes, in the frequency domain:

$$-i\omega\tilde{a}^i(\mathbf{k}, \omega) = \sum_j \tilde{\Lambda}^{ij}(\mathbf{k}, \omega)\tilde{a}^j(\mathbf{k}, \omega) \tag{1.5}$$

Now we can ask ourselves what happens to $\tilde{\Lambda}$ when $\mathbf{k} \to 0$ by doing a multivariate expansion in $\mathbf{k}$. For the reasoning let's take the case where a^j are constants. If there is a constant term in $\tilde{\Lambda}$ not depending on k, we would have a paradoxical case where $\dot{A}^i$, the integral of a^i over the volume, depends linearly on the constant value of a^j, leading to nonsense predictions. This excludes the constant term. Then, if the system responds in the same way to perturbation in opposite direction ($\mathbf{k} \to -\mathbf{k}$), we must put to zero also the term proportional to $\mathbf{k}$ in the multivariate expansion of $\tilde{\Lambda}^{ij}$. So the first term that is not zero is the one quadratic in $\mathbf{k}$. So in general we can write

$$\tilde{\Lambda}^{ij}(\mathbf{k}) = \mathbf{k}^T \mathrm{DD}\tilde{\Lambda}^{ij}(0)\mathbf{k} + \ldots \text{higher order terms} \tag{1.6}$$

where $\mathrm{DD}\tilde{\Lambda}^{ij}(0)$ is the hessian of $\tilde{\Lambda}^{ij}$ evaluated in $\mathbf{k} = 0$. Assuming also that the system is isotropic the expansion becomes easier and we get, in the limit of small k

$$\tilde{\Lambda}^{ij}(k) = k^2\lambda^{ij} \tag{1.7}$$

Using this we can write (1.5) as:

$$-i\omega\tilde{a}^i(\mathbf{k}, \omega) = \sum_j \tilde{k}^2\lambda^{ij}\tilde{a}^j(\mathbf{k}, \omega) \tag{1.8}$$

thus we get, using the continuity equation, for the slow and decoupled hydrodynamic variables the following constitutive equation:

$$\tilde{\boldsymbol{j}}^i(\mathbf{k},\omega) = i\mathbf{k}\sum_j \tilde{\lambda}^{ij}\tilde{a}^j(\mathbf{k},\omega). \tag{1.9}$$

Here the λ^{ij}s completely describe the behavior of the hydrodynamic variables of the system. We can now write an equation for the long wavelength and stationery ($\omega = 0$) limit. In this limit, we have the quantities $\mathbf{J}^i = \frac{1}{\Omega}\int \boldsymbol{j}^i(\mathbf{r})d\mathbf{r}$, the macroscopic current, and $\mathbf{D}^i = \frac{1}{\Omega}\int \nabla a^i(\mathbf{r})d\mathbf{r}$, the macroscopic gradient, related through the equations:

$$\mathbf{J}^i = \sum_j \lambda^{ij}\mathbf{D}^j. \tag{1.10}$$

In the following, the macroscopic component of a current will be indicated as a flux.

Now we insert thermodynamics into our framework. First of all, by macroscopic we mean a property of a region of time and space that is in thermodynamical equilibrium. In this region, we can define thermodynamic variables. Let $x^i = \frac{\partial S}{\partial A^i}$ be the intensive variable conjugate to A^i, where S is the system's entropy, and $\chi^{ij} = \frac{1}{\Omega}\frac{\partial A^i}{\partial x^j}$ the corresponding susceptibility. For instance, when A^i is the energy of the system, the corresponding conjugate variable is the inverse temperature, $x^i = 1/T$, while, when A^i represents the number of particles of a given species, one has $x^i = -\mu^i/T$, μ^i being the corresponding chemical potential. The hypothesis of local thermodynamic equilibrium allows defining local values of the intensive variables, and we define *thermodynamic forces* as their average gradients: $\mathbf{F}^i = \frac{1}{\Omega}\int \nabla x^i(\mathbf{r})d\mathbf{r}$. The average density gradients are related to the thermodynamic forces through the susceptibility defined above:

$$\mathbf{D}^i = \sum_j \chi^{ij}\mathbf{F}^j. \tag{1.11}$$

By inserting this relation into Eq. (1.10), one gets:

$$\mathbf{J}^i = \sum_j L^{ij}\mathbf{F}^j, \tag{1.12}$$

where $L^{ij} = \sum_k \lambda^{ik}\chi^{kj}$. Eq. (1.12) expresses the linear relation between fluxes, the **J**'s, and thermodynamic affinities, the **F**'s, for which Onsager derived his celebrated reciprocity relations ($L^{ji} = L^{ij}$) from microscopic reversibility [57, 58, 13]. In the following section, we will link this macroscopic theory to what happens on a microscopic scale, in order to evaluate Onsager's coefficients with atomistic simulations.

1.1.2 Linear-response theory

In order to evaluate the L^{ij} phenomenological coefficients appearing in Eq. (1.12), we consider a classical system of N interacting atoms described by the Hamiltonian

$$H^\circ(\Gamma) = \sum_n \frac{1}{2M_n}(\mathbf{P}_n)^2 + V(\mathbf{R}_1, \mathbf{R}_2, \cdots \mathbf{R}_N), \tag{1.13}$$

where M_n, $\mathbf{R}_n$, and $\mathbf{P}_n$ are the masses, coordinates, and momenta of the n-th particle, $\Gamma = \{\mathbf{R}_n, \mathbf{P}_n\}$ indicates the phase-space coordinates of the entire system, and V is a generic many-body potential. Let us now suppose that the system is subject to an external

perturbation that can be described as a linear combination of the conserved densities, $\{a^i(\mathbf{r};\Gamma)\}$, as:

$$V'(\Gamma,t) = \sum_i \int v^i(\mathbf{r},t)a^i(\mathbf{r};\Gamma)d\mathbf{r}, \tag{1.14}$$

where $a(\mathbf{r};\Gamma)$ is a phase-space function whose ensemble average is the conserved density,

$$\begin{aligned} a(\mathbf{r}) &= \langle a(\mathbf{r};\Gamma)\rangle \\ &= \int a(\mathbf{r};\Gamma)\mathcal{P}^\circ(\Gamma)d\Gamma, \end{aligned} \tag{1.15}$$

$\mathcal{P}^\circ(\Gamma) \propto \mathrm{e}^{-\frac{H^\circ(\Gamma)}{k_B T}}$ is the equilibrium distribution, k_B the Boltzmann constant, and $\{v^i(\mathbf{r},t)\}$ are time-dependent fields that couple to the conserved densities and vanish at $t=-\infty$, when the system is assumed to be in thermal equilibrium at some temperature T. Of course, conserved currents are also expected values of some phase-space functions, $\boldsymbol{j}(\mathbf{r}) = \langle \boldsymbol{j}(\mathbf{r};\Gamma)\rangle$. The phase-space functions whose expected values are conserved densities/currents will be referred to as *phase-space samples* of the currents/densities. In the following, when the phase-space dependence of a conserved density/current is explicitly indicated, we will mean a phase-space sample; when it is not a phase-space average will be implied. When a phase-space sample is evaluated along a dynamical trajectory, Γ_t, the sample function will depend on time and on the initial conditions of the trajectory. Averaging with respect to the initial conditions will result in a time-dependent expected value for the conserved densities (or currents):

$$\begin{aligned} a(\mathbf{r},t) &= \langle a(\mathbf{r};\Gamma'_t)\rangle_0 \\ &= \int a(\mathbf{r};\Gamma'_t)\mathcal{P}^\circ(\Gamma_0)d\Gamma_0. \end{aligned} \tag{1.16}$$

In Eq. (1.16) the notation Γ'_t denotes somewhat pedantically that the time evolution in phase space is driven by the perturbed Hamiltonian, $H^\circ + V'$. If it were driven by H°, evidently the value of a would be time-independent. In the following, the notation Γ_t will indicate an unperturbed time evolution. As an example, the phase-space sample of the particle density can be assumed to be $n(\mathbf{r};\Gamma) = \sum_n \delta(\mathbf{r}-\mathbf{R}_n)$, the corresponding current is $\boldsymbol{j}(\mathbf{r},\Gamma) = \sum_n \delta(\mathbf{r}-\mathbf{R}_n)\mathbf{P}_n/M_n$, and a local external potential is described by: $V'(\Gamma,t) = \sum_n v(\mathbf{R}_n,t) = \int v'(\mathbf{r},t)n(\mathbf{r};\Gamma)d\mathbf{r}$. Note that sample functions are not necessarily univocally defined. Different functions whose phase-space averages coincide in the long-wavelength limit sample the same hydrodynamical variable.

According to [28], [42], and [43], the linear response of the i-th conserved current to the perturbation is:

$$j^i_\alpha(\mathbf{r},t) = \frac{1}{k_B T}\sum_j \int_{-\infty}^t dt' \int d\mathbf{r}' \left\langle j^i_\alpha(\mathbf{r},\Gamma_t)\dot{a}^j(\mathbf{r}',\Gamma_{t'})\right\rangle_0 v^j(\mathbf{r}',t') \tag{1.17}$$

$$= \frac{-1}{k_B T}\sum_{j,\beta} \int_{-\infty}^t dt' \int d\mathbf{r}' \left\langle j^i_\alpha(\mathbf{r},\Gamma_t)\partial'_\beta j^j_\beta(\mathbf{r}',\Gamma_{t'})\right\rangle_0 v^j(\mathbf{r}',t') \tag{1.18}$$

$$= \frac{1}{k_B T}\sum_{j,\beta} \int_{-\infty}^t dt' \int d\mathbf{r}' \left\langle j^i_\alpha(\mathbf{r},\Gamma_t) j^j_\beta(\mathbf{r}',\Gamma_{t'})\right\rangle_0 \partial'_\beta v^j(\mathbf{r}',t'). \tag{1.19}$$

The second line follows from the first through the continuity equation, Eq. (1.1), while the third line follows after integrating by parts with respect to $\mathbf{r}'$. The notation $\partial'_\beta = \frac{\partial}{\partial r'_\beta}$ has been used.

By integrating Eq. (1.19) all over the space, and assuming space-time homogeneity as well as isotropy, one recovers Eq. (1.12) with:

$$J^i_\alpha(\Gamma) = \frac{1}{\Omega}\int j^i_\alpha(\mathbf{r},\Gamma)d\mathbf{r}, \tag{1.20}$$

$$F^i_\alpha(\Gamma) = \frac{1}{\Omega T}\int \partial_\alpha v^i(\mathbf{r},\Gamma)d\mathbf{r}, \tag{1.21}$$

$$L^{ij}_{\alpha\beta} = \frac{\Omega}{k_B}\int_0^\infty \left\langle J^i_\alpha(\Gamma_t)J^j_\beta(\Gamma_0)\right\rangle_0 dt. \tag{1.22}$$

This completes the derivation of the Green-Kubo formula for transport coefficients, Eq. (1), from classical linear-response theory. Onsager's reciprocity relations, $L^{ij} = L^{ji}$ [57, 58], follow from Eq. (1.22) leveraging time-translational invariance, $\langle J^i_\alpha(\Gamma_t)J^j_\beta(\Gamma_0)\rangle = \langle J^i_\alpha(\Gamma_0)J^j_\beta(\Gamma_{-t})\rangle$, and micro-reversibility, $\langle J^i_\alpha(\Gamma_t)J^j_\beta(\Gamma_0)\rangle = \langle J^i_\alpha(\Gamma_{-t})J^j_\beta(\Gamma_0)\rangle$.
[5]

Einstein-Helfand expression for transport coefficients and the Wiener-Khintchine theorem

The celebrated Einstein's relation between the mean-square displacement of a diffusing particle and its velocity auto-correlation function is easily generalized to an arbitrary stochastic process and has in fact been utilized by [32] to provide an "Einstein-like" expression for transport coefficients.

Let X_t be a stationary stochastic process. One has:

$$\frac{1}{\mathcal{T}}\left\langle\left|\int_0^{\mathcal{T}} X_t dt\right|^2\right\rangle = 2\int_0^{\mathcal{T}}\langle X_t X_0\rangle\, dt - \frac{2}{\mathcal{T}}\int_0^{\mathcal{T}}\langle X_t X_0\rangle\, t\, dt. \tag{1.23}$$

In the large-$\mathcal{T}$ limit, the second term on the right-hand side of Eq. (1.23) can be neglected.

When the stochastic process is the velocity of a Brownian particle, Eq. (1.23) allows one to establish a relation between the diffusion constant of the particle, temperature, and the auto-correlation time of the velocity. When X_t is the heat flux of a macroscopic body, Eq. (1.23) allows one to estimate the thermal conductivity, as given by Eq. (1), from the asymptotic behavior of the "energy displacement" $\mathcal{D}(\tau) = \int_0^\tau \mathbf{J}(\Gamma_t)dt$.

Eq. (1.23) can be easily generalized to the finite-frequency regime, to get:

$$\begin{aligned} S_{\mathcal{T}}(\omega) &= \frac{1}{\mathcal{T}}\left\langle\left|\int_0^{\mathcal{T}} X_t \mathrm{e}^{i\omega t} dt\right|^2\right\rangle \\ &= 2\mathfrak{Re}\int_0^{\mathcal{T}}\langle X_t X_0\rangle\, \mathrm{e}^{i\omega t} dt + \mathcal{O}(\mathcal{T}^{-1}). \end{aligned} \tag{1.24}$$

This equation expresses the Wiener-Khintchine theorem [76, 37], which states that the expectation of the squared modulus of the Fourier transform of a stationary process is the Fourier transform of its time correlation function, which is usually referred to as the process *power spectral density*,

$$S(\omega) = \int_{-\infty}^{\infty}\langle X_t X_0\rangle\, \mathrm{e}^{i\omega t} dt, \tag{1.25}$$

aka the *power spectrum*. In the following the suffix $\mathcal{T}$ will be neglected for simplicity and its value assumed to be sufficiently large as to be considered infinite. More generally, when

several conserved currents interact with each other, one can define the *cross-spectrum* of the conserved fluxes as the Fourier transform of the cross time-correlation functions:

$$\begin{aligned} S^{kl}(\omega) &= \int_{-\infty}^{\infty} \langle X_t^k X_0^l \rangle \, \mathrm{e}^{i\omega t} dt \\ &= \frac{1}{\mathcal{T}} \mathfrak{Re} \left\langle \int_0^{\mathcal{T}} X_t^k \mathrm{e}^{-i\omega t} dt \times \int_0^{\mathcal{T}} X_t^l \mathrm{e}^{i\omega t} dt \right\rangle + \mathcal{O}(\mathcal{T}^{-1}). \end{aligned} \tag{1.26}$$

Eqs. (1.23) and (1.24) indicate that the transport coefficients we are after essentially are the zero-frequency value of the (cross-) power spectrum of the corresponding current(s), a fact that will be instrumental in our approach to data analysis, as explained in Sec. 2. Therefore, Eq. (1.22) can be cast into the form:

$$L^{kl} = \frac{\Omega}{2k_B} S^{kl}(\omega = 0), \tag{1.27}$$

where the Cartesian indices have been omitted for clarity.

1.1.3 Heat transport

The above treatment allows one to compute the linear response of a system at thermal equilibrium to a generic mechanical perturbation. Heat transport is determined by temperature gradients that cannot be described by any mechanical perturbation. The concept of temperature distribution implies that the system is locally at thermal equilibrium over lengths and times large with respect to atomic distances and relaxation times. Temperature affects the physical properties of a system through the Boltzmann distribution function. When the temperature is not constant, $T(\mathbf{r}) = T + \Delta T(\mathbf{r})$ ($|\Delta T| \ll T$), the effects of this inhomogeneity can be formally described by the distribution function:

$$\mathcal{P}(\Gamma) \propto \mathrm{e}^{-\int \frac{e(\mathbf{r};\Gamma)}{k_B T(\mathbf{r})} d\mathbf{r}} \tag{1.28}$$

$$= \mathrm{e}^{-\frac{H^\circ(\Gamma) + V'(\Gamma)}{k_B T}}, \tag{1.29}$$

where $e(\mathbf{r};\Gamma)$ is an energy (Hamiltonian) density, such that $\int e(\mathbf{r};\Gamma) d\mathbf{r} = H^\circ(\Gamma)$. Eq. (1.14) becomes:

$$V'(\Gamma) = -\frac{1}{T} \int \Delta T(\mathbf{r}) e(\mathbf{r};\Gamma) d\mathbf{r} + \mathcal{O}(\Delta T^2). \tag{1.30}$$

Eq. (1.30) shows that the effects of temperature inhomogeneities can be mimicked by a mechanical perturbation coupled to the temperature distribution. From Eqs. (1.12) and (1.20-1.22) we conclude that in a system where the only not trivial conserved quantity is the energy, the heat (energy) flow is coupled to temperature gradients through the constitutive equation:

$$\mathbf{J}^E = -\kappa \nabla T, \tag{1.31}$$

where the thermal conductivity $\kappa_{\alpha\beta} = L^{EE}_{\alpha\beta}/T^2$ (see Eq. (1.12)) can be expressed by a Green-Kubo relation in terms of the fluctuations of the energy flux as:

$$\kappa_{\alpha\beta} = \frac{\Omega}{k_B T^2} \int_0^\infty \left\langle J^E_\alpha(\Gamma_t) J^E_\beta(\Gamma_0) \right\rangle_0 dt, \tag{1.32}$$

and

$$\mathbf{J}^E(\Gamma) = \frac{1}{\Omega} \int \boldsymbol{\jmath}^E(\mathbf{r};\Gamma) d\mathbf{r}. \tag{1.33}$$

In order to obtain an explicit expression for the energy flux from a microscopic expression for the energy density, we multiply the continuity equation, Eq. (1.1), by $\mathbf{r}$ and integrate by parts, to obtain:

$$\mathbf{J}^E(\Gamma_t) = \frac{1}{\Omega}\int \dot{e}(\mathbf{r};\Gamma_t)\,\mathbf{r}\,d\mathbf{r} \tag{1.34}$$

$$= \frac{1}{\Omega}\int \left[\sum_n \left(\frac{\partial e(\mathbf{r};\Gamma_t)}{\partial \mathbf{R}_n}\cdot\mathbf{V}_n + \frac{\partial e(\mathbf{r};\Gamma_t)}{\partial \mathbf{P}_n}\cdot\mathbf{F}_n\right)\right]\mathbf{r}\,d\mathbf{r}, \tag{1.35}$$

where $\mathbf{F}_n$ is the force acting on the n-th atom, and $\mathbf{V}_n = \frac{\mathbf{P}_n}{M_n}$ its velocity.

The manipulations leading from the continuity equation, Eq. (1.1), to Eq. (1.35) deserve some further comments, as they imply neglecting a boundary term, $\mathbf{J}_{\partial\Omega} = \frac{1}{\Omega}\int_{\partial\Omega}(\boldsymbol{j}(\mathbf{r})\cdot\hat{\mathbf{n}})\,\mathbf{r}\,d\mathbf{r}$ (where $\partial\Omega$ is the boundary of the integration volume and $\hat{\mathbf{n}}$ the normal to it), which in general does not vanish in the thermodynamic limit and is ill-defined in periodic boundary conditions (PBC). The correct way of addressing this problem is to work with the Taylor expansion of the space Fourier transform of the continuity equation, Eq. (1.3), and to perform the thermodynamic limit at finite wavelength. The leading non-vanishing term in the Taylor expansion yields Eq. (1.34) without any boundary term in the way.

Energy flux from classical force fields

When atoms interact through a classical force field, $V(\mathbf{R}_1,\mathbf{R}_2,\cdots\mathbf{R}_N)$, an energy density can be defined in terms of local atomic energies as:

$$e(\mathbf{r},\Gamma) = \sum_n \delta(\mathbf{r}-\mathbf{R}_n)e_n(\Gamma), \tag{1.36}$$

$$e_n(\Gamma) = \frac{(\mathbf{P}_n)^2}{2M_n} + v_n(\{\mathbf{R}\}), \tag{1.37}$$

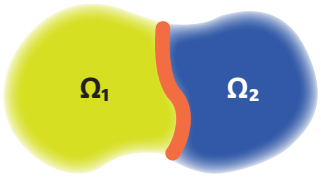
where the v_n's are a set of atomic potential energies whose sum is the total potential energy of the system, $\sum_n v_n = V$, with a short-range dependence on the coordinates of the other atoms. In the presence of long-range forces, this condition is effectively guaranteed by local charge neutrality, which we will assume throughout. By inserting Eq. (1.36) into Eq. (1.35), the energy flux can be cast into the form:

$$\mathbf{J}^E(\Gamma) = \frac{1}{\Omega}\left[\sum_n \mathbf{V}_n e_n + \sum_n \mathbf{R}_n\left(\mathbf{F}_n\cdot\mathbf{V}_n + \sum_m \mathbf{V}_m\cdot\frac{\partial v_n}{\partial \mathbf{R}_m}\right)\right]$$

$$= \frac{1}{\Omega}\left[\sum_n \mathbf{V}_n e_n + \sum_{n,m}(\mathbf{R}_n-\mathbf{R}_m)\mathbf{F}_{nm}\cdot\mathbf{V}_n\right], \tag{1.38}$$

where $\mathbf{F}_{nm} = -\frac{\partial v_m}{\partial \mathbf{R}_n}$ is the contribution of the m-th atom to the force acting on the n-th atom, $\sum_m \mathbf{F}_{nm} = \mathbf{F}_n$, and $\mathbf{F}_{nm} = -\mathbf{F}_{mn}$. When the interaction amongst atoms can be expressed in terms of two-body potentials, one has: $v_m = \frac{1}{2}\sum_n v(\mathbf{R}_n-\mathbf{R}_m)$ and $\mathbf{F}_{nm} = -\frac{1}{2}\nabla_{\mathbf{R}_n} v(\mathbf{R}_n-\mathbf{R}_m)$. Here we implicitly assumed that the interaction energy is equally partitioned between atoms m and n. In Sec. 1.2 we shall see this is not the only possible choice, with far-reaching consequences on the theory of heat transport.

The first term on the right-hand side of Eq. (1.38) is often called *convective* and the second *virial*. We feel that the wording "convective" is somewhat misleading in this context, as the convective current, as well as its contribution to heat conductivity, may not vanish even in the absence of convection.



$$E(\Omega_1 \cup \Omega_2) = E(\Omega_1) + E(\Omega_2) + W_{12}$$
$$\stackrel{?}{=} \mathcal{E}(\Omega_1) + \mathcal{E}(\Omega_2)$$

Figure 1.1: The energy of an isolated system is the sum of the energies of its subsystems (as defined when they are isolated as well) plus the interaction among them, W_{12}, whose magnitude scales as the area of the interface, depicted in red. When defining the energies of individual subsystems, $\mathcal{E}$, W_{12} has to be arbitrarily partitioned among them.

1.2 Gauge invariance of heat transport coefficients

It is often implicitly assumed that the well-definiteness of thermal transport coefficients would stem from the uniqueness of the decomposition of the system's total energy into localized, atomic, contributions. This assumption is manifestly incorrect, as any decomposition leading to the same value for the total energy as Eq. (1.37) should be considered as legitimate. The difficulty of partitioning a system's energy into subsystems' contributions is illustrated in Fig. 1.1, which depicts a system made of two interacting subsystems. When defining the energy of each of the two subsystems, an arbitrary decision has to be made as to how the interaction energy is partitioned. In the case depicted in Fig. 1.1, for instance, the energy of each of the two subsystems can be defined as $\mathcal{E}(\Omega_i) = E(\Omega_i) + \frac{1}{2}(1 \pm \lambda)W_{12}$, where $E(\Omega_i)$ are the energies of the two isolated subsystems, W_{12} their interaction energy, and λ an arbitrary constant. In the thermodynamic limit, when all the subsystems' energies are much larger than the interaction between any pairs of them, the value of the λ constant is irrelevant. When it comes to defining energy densities (*i.e.* energies of infinitesimal portions of a system) or atomic energies, instead, the magnitude of the interaction between different subsystems is comparable to their energies, which become therefore intrinsically ill-defined.

The very possibility of defining an energy current density stems from energy extensivity. The considerations illustrated in Fig. 1.1 indicate that any two densities, $e'(\mathbf{r}, t)$ and $e(\mathbf{r}, t)$, whose integrals over a macroscopic volume differ by a quantity that scales as the volume boundary, should be considered as equivalent. This equivalence can be expressed by the condition that two equivalent densities differ by the divergence of a (bounded) vector field:

$$e'(\mathbf{r}, t) = e(\mathbf{r}, t) - \nabla \cdot \boldsymbol{p}(\mathbf{r}, t). \tag{1.39}$$

In a sense, two equivalent energy densities can be thought of as different *gauges* of the same scalar field. Energy is also conserved: because of this, for any given gauge of the energy density, $e(\mathbf{r}, t)$, an energy current density can be defined, $\boldsymbol{j}(\mathbf{r}, t)$, so as to satisfy the continuity equation, Eq. (1.1). By combining Eqs. (1.39) and (1.1) we see that energy current densities and macroscopic fluxes transform under a gauge transformation as:

$$\boldsymbol{j}'(\mathbf{r}, t) = \boldsymbol{j}(\mathbf{r}, t) + \dot{\boldsymbol{p}}(\mathbf{r}, t), \tag{1.40}$$
$$\mathbf{J}'(t) = \mathbf{J}(t) + \dot{\mathbf{P}}(t), \tag{1.41}$$

where $\mathbf{P}(t) = \frac{1}{\Omega}\int \boldsymbol{p}(\mathbf{r}, t)d\mathbf{r}$. We conclude that the macroscopic energy fluxes in two different energy gauges differ by the total time derivative of a bounded phase-space vector function.

We now show that the energy fluxes of the same system in two different energy gauges, e and e', differing by a bounded total time derivative, as in Eq. (1.41), result in the same heat conductivity, as given by the Green-Kubo formula, Eq. (1.32). More generally, the Onsager coefficients coupling two fluxes, $\mathbf{J}^1$ and $\mathbf{J}^2$, do not depend on the gauge of either one of them. In fact, let $(\mathbf{J}^1)' = \mathbf{J}^1 + \dot{\mathbf{P}}$; one has:

$$\begin{aligned}\left(L^{11}\right)' &= \frac{\Omega}{2k_B}\int_{-\infty}^{+\infty}\left\langle\left(\mathbf{J}^1(t)+\dot{\mathbf{P}}(t)\right)\cdot\left(\mathbf{J}^1(0)+\dot{\mathbf{P}}(0)\right)\right\rangle dt \\ &= L^{11} + \frac{\Omega}{2k_B}\left[\left\langle\mathbf{P}(t)\cdot\dot{\mathbf{P}}(0)\right\rangle\Big|_{-\infty}^{+\infty} + 2\left\langle\mathbf{P}(t)\cdot\mathbf{J}^1(0)\right\rangle\Big|_{-\infty}^{+\infty}\right].\end{aligned} \tag{1.42}$$

The expectation of the time-lagged products in Eq. (1.42) is equal to the products of two expectations at a large time lag. As the equilibrium expectations of both a total time derivative and a current vanish, we conclude that $\left(L^{11}\right)' = L^{11}$. A slight generalization of this argument, also using microscopic reversibility as in [57, 58], allows us to conclude that $\left(L^{12}\right)' = L^{12}$ and that, in general, $\kappa' = \kappa$.

1.3 Theory of heat transport in multicomponent systems

[7]

Let us consider a homogeneous system with M components. One can be tempted to simply apply the formula that is valid in the one component case, that is

$$\kappa = \frac{\Omega}{k_b T^2}\int_0^\infty \langle \mathbf{J}^E(t)\cdot\mathbf{J}^E(0)\rangle_0\, dt \tag{1.43}$$

but this is *meaningless* in the multicomponent case if one uses the energy current. The fact that this expression does not represent anything physical can be seen very simply. The energy current in general has a part that depends on how is defined the zero of the potential, proportional to the atomic velocity. In the classical case we have:

$$\mathbf{J}^E(t) = \sum_n \mathbf{v}_n e_n + \text{other terms} \tag{1.44}$$

where $\mathbf{v}_n$ is the atomic velocity of atom n and e_n is its potential energy, which depends on the positions of all other atoms. If we change the zero of the potential energy of each component s of e_0^s, the potential energy transforms as $e_n(\Gamma) \to e_n(\Gamma) + e_0^{s(n)}$ (being $s(n)$ the component index of atom n), and the energy current transform as

$$\mathbf{J}^E(t) \to \mathbf{J}^E(t) + \sum_s \mathbf{J}^s e_0^s \tag{1.45}$$

where, given the set $I(s)$ of atomic indexes of component s, $\mathbf{J}^s = \sum_{n\in I(s)} \mathbf{v}_n$. The issue is that because each component is diffusive, in general, $\int_0^\infty \langle \mathbf{J}^s(t)\cdot\mathbf{J}^{s'}(0)\rangle_0\, dt \neq 0$, so κ would transform with a clear dependence on e_0^s, that is an arbitrary parameter. And since different e_0^s bring the same equations of motion (the description of the system is exactly the same), we conclude that if we use the one component formula in the multicomponent case, we end up with an arbitrary number. In the formula we need, somehow, to include only energy differences.

It is possible to define many slightly different heat conduction coefficients, and many definitions of heat flux are possible[29, Cap.III, §3]. One can decide to calculate the

coefficient when all the gradients of the concentrations ∇c^i are zero or when the mass currents $\mathbf{J}^i$ are zero. The first case is typical of the beginning of experiments, while the last one happens at the end when the steady state is reached [29, eq.(241) and (242), p. 280]. The first path needs the additional calculation of partial enthalpies[16, 65], and after that, the method is essentially the same as the one component case. Only a single L^{QQ} Onsager coefficient is needed, calculated with a newly defined heat current, that does not depend on the arbitrary zero of the potential energy. This is the path that a big part of the literature follows, often introducing approximation for the calculation of the partial enthalpies. But since we are interested in the steady state, we investigate the latter.

We have in total $M+1$ scalar conserved quantities: the total energy and the mass (or the particle number) for each fluid. But since the mass currents are momentum densities, in the center of mass frame the total momentum density is zero. So we have one less independent mass current, giving to us $M-1$ independent mass densities and currents. This gives M independent conserved quantities. The Onsager relations will look like this:

$$\begin{pmatrix} \boldsymbol{J}^E \\ \boldsymbol{J}^1 \\ \vdots \\ \boldsymbol{J}^{M-1} \end{pmatrix} = \left(\begin{array}{c|c} L^{EE} & L^{E\,1} \dots L^{E\,M-1} \\ \hline L^{1\,E} & \\ \vdots & \{L^{ij}\} \\ L^{M-1\,E} & \end{array}\right) \begin{pmatrix} \boldsymbol{X}^E \\ \boldsymbol{X}^1 \\ \vdots \\ \boldsymbol{X}^{M-1} \end{pmatrix} \tag{1.46}$$

$$\tag{1.47}$$

or, by grouping together the block matrices:

$$\begin{pmatrix} \boldsymbol{J}^E \\ J \end{pmatrix} = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \begin{pmatrix} \boldsymbol{X}^E \\ X \end{pmatrix} \tag{1.48}$$

Thermal conduction happens when all the mass fluxes vanish (convection is not present). We want to extract an expression with only $\boldsymbol{X}^E$ present. So the first step is to invert the expression by multiplying on the left with the L^{-1} matrix to have in the right side of the first equation $\boldsymbol{X}^E$ only. Then by setting all the mass currents $\boldsymbol{J}^i$ to zero we can get an equation with $\boldsymbol{J}^E$ and $\boldsymbol{X}^E$ only. The coefficient in front of $\boldsymbol{J}^E$ is the inverse of the thermal conductivity we are looking for. This coefficient can be written using the Shur complement to invert the matrix block by block:

$$\left(\begin{array}{c|c} (A - BD^{-1}C)^{-1} & \dots \\ \hline \dots & \dots \end{array}\right) \begin{pmatrix} \boldsymbol{J}_E \\ J \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_E \\ X \end{pmatrix} \tag{1.49}$$

So we have that the thermal conductivity is

$$\kappa = \frac{1}{(L^{-1})^{EE}} = A - BD^{-1}C = L^{EE} - L^{Ei}\{L_\star^{-1}\}^{ij}L^{jE} \tag{1.50}$$

where the implicit summation is over the mass block indexes, $(1, \dots, M-1)$ and $L_\star$ is the submatrix of the mass block only.

[20, 5]

1.3.1 Convective invariance

Let us make the convention that Latin indexes refer to the mass indexes $(1, \dots, M-1)$, and index E refers to the first one. Consider a transformation of the energy current

$$J^E \to J^E + c^i J^i \tag{1.51}$$

where to lighten the notation, repeated indexes are implicitly summed. This transformation results, for example, when the zero of the potential energy per species is shifted by a constant or when one uses the heat current with the partial enthalpies terms in place of the energy current. In general, one cannot assume that mass currents are non-diffusing, so if one adds a constant to the atomic energy of an atomic specie, the L^{EE} term of the matrix $L^{\mu\nu}$ *changes*. Applying the transformation (1.51) to Eq.(1.50) one gets:

$$L^{EE} \rightarrow L^{EE} + c^i L^{iE} + c^i L^{Ei} + c^i c^j L^{ij} \tag{1.52}$$

$$L^{iE} \rightarrow L^{iE} + c^j L^{ij} \tag{1.53}$$

$$L^{Ei} \rightarrow L^{Ei} + c^j L^{ji} \tag{1.54}$$

$$L^{ij} \rightarrow L^{ij} \tag{1.55}$$

Therefore the Schur complement, the thermal conductivity κ, transforms as:

$$\begin{aligned} &L^{EE} - L^{iE}(L_\star^{-1})^{ji}L^{Ej} \rightarrow \\ &\rightarrow L^{EE} + c^i L^{iE} + c^i L^{Ei} + c^i c^j L^{ij} - (L^{iE} + c^k L^{ik})(L_\star^{-1})^{ji}(L^{Ej} + c^l L^{lj}) \\ &= L^{EE} + c^i L^{iE} + c^i L^{Ei} + c^i c^j L^{ij} - L^{iE}(L_\star^{-1})^{ji}L^{Ej} - c^k \delta_{kj} L^{Ej} - c^k \delta_{kj} c^l L^{lj} - L^{iE}\delta_{il}c^l \\ &= L^{EE} - L^{iE}(L_\star^{-1})^{ji}L^{Ej} \end{aligned} \tag{1.56}$$

We note that these transformations are also valid for $L^{\mu\nu}(T) = \int_0^T J^\mu(0) \cdot J^\nu(t)\, dt$ and for a frequency dependent $L^{\mu\nu}(\omega)$ that can be defined via the Wiener–Khinchin theorem.

Therefore, *two different definitions of J^E, differing by a linear combination of the mass currents, lead to the same thermal conductivity κ, and to the same $\kappa(T)$ and $\kappa(\omega)$.*

This extends the previous gauge invariance principle for the single-component system. The multicomponent expression also makes useless the computation of the partial enthalpies, as they enter the heat current definition proportionally to the mass fluxes.

Also, a simpler invariance is present. We can apply the transformation on the mass fluxes only $\mathbf{J}^s \rightarrow \Lambda^{sl}\mathbf{J}^l$, where Λ is any invertible matrix. Then, since Λ^{-1} cancels out trivially with Λ in any place where it appears in the Shur complement, κ does not change. This allows us to completely forget about any factor or units in front of the mass fluxes if we are interested in κ only. *Any* non-singular linear combination of the mass fluxes can be used.

2

Data analysis

If we apply the knowledge of the previous chapter 1, as-is, to a real-world case, we can produce a number difficult to interpret. Since it is not always possible to perform arbitrary long computer simulations to estimate errors with a block analysis, a more profound understanding of the statistical properties and an advanced noise filtering technique are needed. In the literature, we can find many different methods to solve the issue of convergence of the GK integrals, like a fitting of the correlation function, a rule to truncate the integral or a rule to subtract non-diffusing currents from the energy one. This section describes techniques that solve most issues without introducing ad-hoc prescriptions and using sound mathematical derivations. The final result will be an elegant framework to compute κ and its statistical error with optimally short simulations using a technique called *cepstral analysis*, that comes from the field of audio data filtering [10]. The most important feature of this technique is that we don't need to split the trajectory into blocks to obtain the statistical error. The new multicomponent ideas developed by the author in the present chapter were used in many computational works, for example, [59, 27, 70, 60, 26, 69].

2.1 On the statistical distribution of κ's estimators

At this point of the work, we have derived a set of expressions, like (1.22) and (1.27), that together with the explicit expression of the energy current (1.35) and its specialization for classical force field (1.38) allows us to evaluate the Onsager matrix L with an equilibrium molecular dynamics simulation, and the heat transport coefficient (1.50). But to perform efficient data analysis, we first need to understand the statistical properties of the estimators we defined.

A multi-current time series can be viewed as a realization of a multivariate statistical process. The space autocorrelation functions of conserved currents are usually short-ranged because far-away atoms do not interact that much. Therefore, in the thermodynamic limit, the corresponding fluxes can be seen as sums of independent identically distributed stochastic (iid) random variables. According to the central limit theorem, their equilibrium distribution must be Gaussian. So the time series of all the independent fluxes of the system is a multivariate stochastic variable that tends to a multivariate normal distribution.

We saw in the previous chapters two different estimators for κ in the multicomponent framework. The most straightforward way is to evaluate all the integrals of (1.22) truncated at timestep t and then to project over the energy current subspace with (1.50). That means:

- sample M scalar random variables $(J^E, J^1, \ldots, J^{M-1})$ from ℓ independent gaussian processes. M is the number of independent currents, and ℓ can be, for example, the number of spatial dimensions of the system or the number of different simulations. The process is repeated N times, being N the number of timesteps. We name t the index of the discrete timestep of the simulation
- compute all the $L^{\mu\nu}$ GK integral up to timestep t and evaluate κ for that fixed value of t.

$$c^{\mu\nu}(t') = \frac{1}{N'} \sum_{s=0}^{N'-1} J^{\mu}(s) J^{\nu}(s+t') \ , \tag{2.1}$$

$$L^{\mu\nu}(t) \propto \sum_{t'<t} c^{\mu\nu}(t') \, dt \tag{2.2}$$

where we performed an average over N' timesteps, assuming the ergodicity of the molecular dynamics simulation. Given that the average covers enough uncorrelated timesteps, $c^{\mu\nu}(t')$ will be a normal random variable for each t'. When estimating $c^{\mu\nu}(t)$ in a finite-size system, it never goes to 0. From the t' where the actual expectation value $\langle c^{\mu\nu}(t') \rangle$ is zero, the sampled value $c^{\mu\nu}(t')$ will start to fluctuate randomly around zero, being nothing but the noise coming from thermal fluctuations of the finite-size system. So starting from this t' on, we can safely assume that we are integrating random numbers with a gaussian distribution and zero average. Thus $L^{\mu\nu}(t)$ will behave like a random walk, eventually reaching every possible value given a big enough t. While this behavior can be acceptable in the one-component case, where inverting a matrix is unnecessary, this can be problematic in the multicomponent case. Take as an example the two-component case. The expression is

$$\kappa(t) = L^{E\,E}(t) - \frac{\left(L^{E\,1}(t)\right)^2}{L^{1\,1}(t)} \tag{2.3}$$

If there is an insufficient statistic, the denominator $L^{1\,1}(t)$ can cross zero for a specific upper time integral value. In the general case, the L matrix can cross a singular region at any time with non-zero probability, becoming non-invertible. We can have anything from random divergences to systematic errors that depend on the variance of the random walks. But the more critical fact is that a well-defined probability density does not exist for this estimator in the multicomponent case, since the hypothetical probability density does not integrate to a finite number. From a mathematically rigorous point of view, one should never use the GK time integral to estimate the thermal transport coefficient using the multicomponent framework, while in practice, if the statistic is big enough, it can be acceptable.

An example of this bad behavior is presented in figure 2.1. One simple solution is to run longer molecular dynamics simulations, but this could not be possible if, for example, we have to deal with costly ab-initio molecular dynamics simulations. The only way to estimate the error using the GK integral technique is to perform a block analysis. The Einstein-Helfand expression (1.23) behaves better, having at the denominator a non-negative quantity.

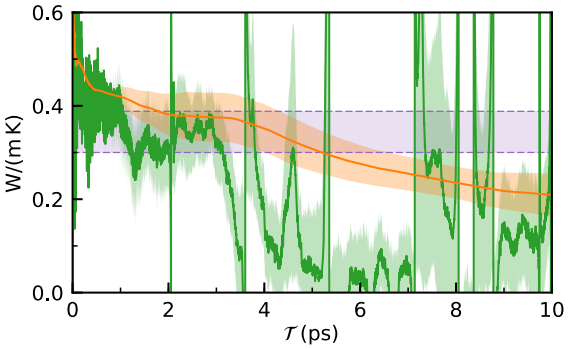


Figure 2.1: Figure taken from [7]. Bad behavior is presented, caused by a not sufficient amount of statistics. The data comes from a classical 100ps simulation of a liquid mixture of water and ethanol flexible molecules. The convergence of $\kappa(t)$ is never achieved with the GK formula, but random divergences appear (green line). The smoother orange line is calculated with the Einstein-Helfand formula, which can improve the plain GK integral result and has better statistical properties.

The next easier estimator that one could compute is the one derived from the Wiener-Khinchin theorem, discretizing Eq.(1.26). We do a discrete Fourier transform [1] and we evaluate everything in the frequency domain. We look for the result at $\omega = 0$ (ω and t are discrete indexes from the integration timestep in molecular dynamics). So we define, given the number of steps N and the timestep size ϵ:

$$\tilde{J}^{\mu}(\omega) = \sqrt{\frac{\epsilon}{N}} \sum_{t=0}^{N-1} e^{-\frac{2\pi i}{N}\omega t} J^{\mu}(t) \tag{2.4}$$

$$S^{\mu\nu}(\omega) = \tilde{J}^{\mu *}(\omega)\tilde{J}^{\nu}(\omega) \tag{2.5}$$

to have $L^{\mu\nu} = S^{\mu\nu}(\omega = 0)$. The $\sqrt{\epsilon/N}$ factor is needed to make the $1/\mathcal{T}$ (with $\mathcal{T} = N\epsilon$) and the dt (that now is called ϵ) factors of (1.26). In the frequency domain, the matrix is well-behaved. Take, for example, the same two-component case of before with ℓ, the number of independent time series, being 3:

$$\kappa(\omega) \propto \left|\tilde{\boldsymbol{J}}^{E}(\omega)\right|^2 - \frac{\left|\tilde{\boldsymbol{J}}^{E*}(\omega) \cdot \tilde{\boldsymbol{J}}^{1}(\omega)\right|^2}{\left|\tilde{\boldsymbol{J}}^{1}(\omega)\right|^2} \tag{2.6}$$

In this case, we still have the random variable at the denominator, but the denominator never becomes zero.

A little note: when everything is relatively simple, as it is up to now, it makes no sense to complicate the notation to distinguish between *estimators* of a quantity and the quantity itself, its theoretical value. But now, as we'll see, with complex matrix distribution the distinction between the estimator and its theoretical value becomes fundamental, as many factors that depend on the particular distribution enters the estimator but of course not its theoretical value. Failing to distinguish the two would bring wrong results. For this reason, now we will label all the estimators with $\hat{\cdot}$ (a little hat).

[1] We do this because we have been taught since our childhood that when we don't know what to do, we have to do a Fourier transform.

We can start with the estimator for $S^{\mu\nu}(\omega)$ – ω being a discrete index. We have ℓ independent processes (for example, one for each spatial dimension) and M independent fluxes that enter the Onsager relations. So the estimator is

$$ {}^{(\ell M)}\hat{S}^{\mu\nu}(\omega) = \frac{1}{\ell}\sum_{p=1}^{\ell} J_p^{\mu}(\omega)\, J_p^{\nu *}(\omega) \tag{2.7} $$

where we performed an average over the independent process. We will need the labels ℓ (number of processes) and M (number of independent fluxes and size of the matrix) later.

We see that $\hat{S}^{\mu\nu}(\omega)$ is an unbiased estimator of the cross-spectrum. Its average is the theoretical value: $\left\langle {}^{(\ell M)}\hat{S}^{ij} \right\rangle = S^{\mu\nu}(\omega)$ for each frequency. In the large-N limit, the real and imaginary parts of $\tilde{J}^i(\omega)$ are normal deviates that are uncorrelated for $\omega \neq \omega'$. We conclude that the cross-periodogram is a random matrix distributed as a complex Wishart deviate [25, 24]:

$$ {}^{(\ell M)}\hat{S}(\omega) \sim \mathcal{CW}_M\left(S(\omega), \ell\right). \tag{2.8} $$

The notation $\mathcal{CW}_M(S, \ell)$ in Eq. (2.8) indicates the distribution of the $M \times M$ Hermitian matrix ${}^{(\ell M)}\hat{S}^{ij} = \frac{1}{\ell}\sum_{p=1}^{\ell} X_p^i X_p^{j*}$, where $\{X_p^i\}$ ($p = 1, \cdots \ell$, $i = 1, \cdots M$) are ℓ samples of an M-dimensional zero-mean normal variate whose covariance is $S^{ij} = \langle X^i X^{j*} \rangle$.

Similarly to the real case, a Bartlett decomposition [41][9, Proposition 7.9] holds for complex Wishart matrices [52], reading:

$$ {}^{(\ell M)}\hat{S} = \frac{1}{\ell}\mathcal{S} R R^{\top} \mathcal{S}^{\dagger}, \tag{2.9} $$

where "$\top$" and "$\dagger$" indicate the transpose and the adjoint of a real and complex matrix, respectively; $\mathcal{S}$ is the lower triangular Cholesky factor of the covariance matrix, $S = \mathcal{S}\mathcal{S}^{\dagger}$, and R is a real $M \times M$ lower triangular random matrix of the form

$$ R = \begin{pmatrix} c_1 & 0 & 0 & \cdots & 0 \\ n_{21} & c_2 & 0 & \cdots & 0 \\ n_{31} & n_{32} & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{M1} & n_{M2} & n_{M3} & \cdots & c_M \end{pmatrix}, \tag{2.10} $$

where $c_i^2 \sim \chi^2_{2(\ell-i+1)}$ (a χ^2 distribution with $2(\ell - i + 1)$ degrees of freedom) and $n_{ij} \sim \mathcal{N}(0, 1)$ (the normal distribution). We stress that R is independent of the specific covariance matrix and only depends upon ℓ and M. In particular, it is independent of the ordering of the fluxes J^i. This distribution is valid for $\omega \neq 0 \wedge \omega \neq N/2$ since here the matrix is real. For the real case $c_i^2 \sim \chi^2_{\ell-i+1}$, without the factor 2. To know the distribution of κ we must compute the distribution of the inverse, that is

$$ {}^{(\ell M)}\hat{S}^{-1} = \ell (\mathcal{S}^{\dagger})^{-1} (R^{\top})^{-1} R^{-1} \mathcal{S}^{-1}, \tag{2.11} $$

Note that the product of lower (upper) triangular matrices is lower (upper) triangular, and the inverse of a lower (upper) triangular matrix is lower (upper) triangular. The determinant is equal to the product of the diagonal terms.

By expressing the M, M matrix element (where the matrix product is very easy to compute) of the inverse of ${}^{(\ell M)}\hat{S}$ in Eq. (2.9) as the ratio between the corresponding minor

and the full determinant, and using some obvious properties of the determinants and of triangular matrices, we find that:

$$\left({}^{(\ell M)}\hat{S}^{-1}\right)^{MM} = \ell \left(S^{-1}\right)^{MM} \frac{1}{c_M^2} \tag{2.12}$$

We are interested in the inverse of the right hand side without the ℓ factor:

$$\frac{\ell}{\left({}^{(\ell M)}\hat{S}^{-1}(\omega)\right)^{MM}} = \frac{1}{(S^{-1}(\omega))^{MM}} c_M^2, \tag{2.13}$$

As the ordering of the fluxes is arbitrary, a similar relation holds for all the diagonal elements of the inverse of the cross-periodogram. Note that the expectation value of c_M^2 is the number of degrees of freedom of the χ^2 distribution. So, to get as an expectation value a pure S^{-1} matrix element without other factors we have to divide the previous expression by that number. We conclude that the generic estimator for $\kappa(\omega) \equiv \frac{1}{(S^{-1}(\omega))^{11}}$ is:

$$\hat{\kappa}(\omega) \equiv \frac{\ell}{\langle c_M^2(\omega)\rangle} \frac{1}{\left({}^{(\ell M)}\hat{S}^{-1}(\omega)\right)^{11}} = \frac{1}{(S^{-1}(\omega))^{11}}\, \xi(\omega) \equiv \kappa(\omega)\xi(\omega), \tag{2.14}$$

where $\xi(\omega)$ are independent random (with respect to ω) random variables, with expectation value 1, distributed as

$$\xi(\omega) \sim \begin{cases} \frac{1}{\ell - M + 1}\, \chi^2_{\ell - M + 1} & \text{for } \omega \in \{0, \frac{N}{2}\}, \\ \\ \frac{1}{2(\ell - M + 1)}\, \chi^2_{2(\ell - M + 1)} & \text{otherwise.} \end{cases} \tag{2.15}$$

and

$$\langle c_M^2(\omega)\rangle = \begin{cases} \ell - M + 1 & \text{for } \omega \in \{0, \frac{N}{2}\}, \\ \\ 2(\ell - M + 1) & \text{otherwise.} \end{cases} \tag{2.16}$$

where in the case $\omega = 0, \frac{N}{2}$ we have half the degrees of freedom because of the reality of the Fourier transforms. Note the extra factor that is needed to get a correct result, which depends on the input statistics. A real-world example of $\hat{\kappa}(\omega)$ is provided in fig. 2.2.

We conclude this section by remarking, once again, that the most straightforward method, calculating first the averaged time integrals of all the autocorrelation functions and then inverting the matrix has the issue of random walks. On the contrary, the frequency domain approach has the advantage to have well-defined statistics, but we have two issues that prevent us to use this result "as is", without doing block averages. Each discrete ω is statistically independent of the nearby $\omega + 1$, and the estimator is not consistent (its variance does not vanish in the large N limit). One can take advantage of the fact that we know the exact statistical properties in the frequency domain to build an ad-hoc filter that analytically estimates the result's variance, assuming the continuity of the expectation value $\kappa(\omega)$. This is the goal of *cepstral analysis*.

2.2 Cepstral analysis

[19, 7] We start from the form of the estimator:

$$\hat{\kappa}(\omega) = \kappa(\omega)\xi(\omega) \tag{2.17}$$

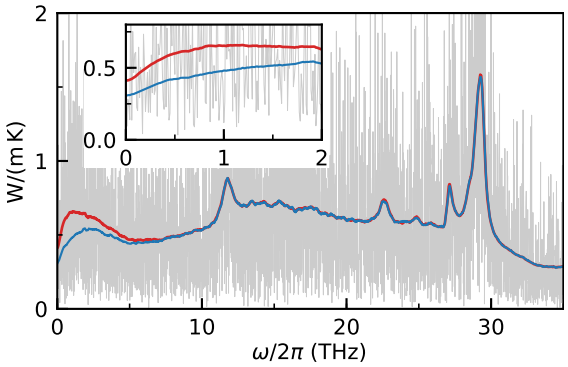


Figure 2.2: The grey discontinuous line is an example of $\hat{\kappa}(\omega)$ values for the liquid water-ethanol mixture, each discrete ω value is uncorrelated from the next one. The continuous line is a running average for guiding the eye. The higher red line is the one obtained with the J^E flux only, and the lower blue line is obtained with the correct multicomponent expression.

as in (2.14) and (2.15). The simpler idea that we can try to filter out the multiplicative noise is to perform a moving average over ω, averaging all neighbor frequencies up to a small enough range. This approach has many issues. The first one is that we are interested in the limit $\omega \to 0$ in the low-frequency region, and here we have fewer available points to perform the average (the spectrum is an even function). Then the finite size of the averaging window adds a systematic error since $\kappa(\omega)$ is not constant there. Nevertheless, the moving average can be used for visualization purposes of the spectrum, as a guide for the eye.

But looking carefully at $\hat{\kappa}(\omega)$, one can think to take the logarithm to transform the multiplicative noise into an additive one, that can be filtered out easily. Then we can take a new discrete Fourier transform, and in the new "frequency-frequency" space filter out the noise that is present at high "frequency-frequency" (labeled with $\bar{t}$), where the value of the signal that we want to get is zero. Filtering, in this case, is equivalent to deciding a cutoff, putting to zero everything after the cutoff, and back transforming to the frequency space. By doing the logarithm we get:

$$\log(\hat{\kappa}(\omega)) = \log(\kappa(\omega)) + \log(\xi(\omega)) \tag{2.18}$$

By performing the discrete Fourier transform we get the so-called *cepstrum*:

$$\hat{C}(\bar{t}) = \frac{1}{N}\sum_{\omega=0}^{N-1}(\log(\kappa(\omega)) + \log(\xi(\omega)))\,\mathrm{e}^{-2\pi i\frac{\omega\bar{t}}{N}} \tag{2.19}$$

$$= C(\bar{t}) + \frac{1}{N}\sum_{\omega=0}^{N-1}\log(\xi(\omega))\mathrm{e}^{-2\pi i\frac{\omega\bar{t}}{N}} \tag{2.20}$$

$$= C(\bar{t}) + \frac{1}{N}\left[\log(\xi(0)) + (-1)^{\bar{t}}\log(\xi(N/2)) + 2\sum_{\omega=1}^{N/2-1}\log(\xi(\omega))\cos\left(2\pi\frac{\omega\bar{t}}{N}\right)\right], \tag{2.21}$$

where $C(\bar{t}) = \frac{1}{N}\sum_{\omega=0}^{N-1}\log(\kappa(\omega))\mathrm{e}^{-2\pi i\frac{\omega\bar{t}}{N}}$, the cepstrum of the true quantity. It is a real even function because $\hat{\kappa}(\omega)$ is real and even. The explicit symmetry is written in (2.21)

for an even number of points N, which is necessary when computing statistical properties. Otherwise, one can fail to consider a random variable and its symmetrical, entirely dependent, copy as the same random variable. According to a generalized central-limit theorem for the Fourier transform of independent and identically distributed (iid) stochastic variables [4, 62], for $N \to \infty$ the *cepstral coefficients*, $\hat{C}(\bar{t})$, are iid normal variates whose expectation value is

$$\begin{aligned}\langle \hat{C}(\bar{t})\rangle &= C(\bar{t}) + \frac{1}{N}\sum_{\omega=0}^{N-1}\langle \log(\xi(\omega))\rangle \mathrm{e}^{-2\pi i \frac{\omega \bar{t}}{N}} \\ &= C(\bar{t}) + \frac{1}{N}\sum_{\omega=0}^{N-1} L_0(\omega)\mathrm{e}^{-2\pi i \frac{\omega \bar{t}}{N}} \\ &= C(\bar{t}) + \bar{L}_0\delta_{\bar{t}0} + O(1/N),\end{aligned} \tag{2.22}$$

where we defined the expectation value of $\log(\xi(\omega))$ as

$$L_0(\omega) = \begin{cases} \psi^{(0)}((\ell - M + 1)/2) - \log((\ell - M + 1)/2) & \text{for } \omega \in \{0, \frac{N}{2}\}, \\ \\ \bar{L}_0 \equiv \psi(\ell - M + 1) - \log(\ell - M + 1) & \text{otherwise.} \end{cases} \tag{2.23}$$

and $\psi^{(0)}$ is the digamma function. [2] We can make the $\bar{t} = 0$ and $\bar{t} = N/2$ values of the sum in the (2.22) expression almost equal to $\bar{L}_0$ with an error of order $1/N$ ignoring the $\omega = 0$ and $\omega = N/2$ differences. We can do the same for the other values of $\bar{t}$, obtaining zero with an error of order $1/N$. Since N, the number of samples, is huge, this is a safe approximation. Similarly, we can compute the variance. Here it is crucial to carefully take care of the symmetry of $\xi(\omega)$ and the fact that $\xi(\omega) \equiv \xi(N - \omega)$ (they are the same, not independent, random variable), as it is written in (2.21). The variance of a single $\log(\xi(\omega))$ is:

$$\sigma_0^2(\omega) \equiv \mathrm{var}(\log(\xi(\omega))) = \begin{cases} \psi^{(1)}((\ell - M + 1)/2) & \text{for } \omega \in \{0, \frac{N}{2}\}, \\ \\ \bar{\sigma}_0^2 \equiv \psi'(\ell - M + 1) & \text{otherwise,} \end{cases} \tag{2.24}$$

Where $\psi^{(1)}(x)$ is also called the trigamma function. As before, we can approximate $\sigma_0^2(0)$ and $\sigma_0^2(N/2)$ to $\bar{\sigma}_0^2$ making an error $O(1/N)$ in the computation of the variance, so we get:

$$\mathrm{var}(\hat{C}(\bar{t})) = \begin{cases} 2\frac{\bar{\sigma}_0^2}{N} & \text{for } \bar{t} \in \{0, \frac{N}{2}\}, \\ \\ \frac{\bar{\sigma}_0^2}{N} & \text{otherwise,} \end{cases} \tag{2.25}$$

Unfortunately, the expectation value of the inverse Fourier transform of $\hat{C}(\bar{t})$ does not represent $\log(\kappa(\omega))$ because of the additional $\bar{L}_0$ factor:

$$\left\langle \sum_{\bar{t}=0}^{N-1} \hat{C}(\bar{t})\mathrm{e}^{2\pi i \frac{\bar{t}\omega}{N}} \right\rangle = \bar{L}_0 + \sum_{\bar{t}=0}^{N-1} C(\bar{t})\mathrm{e}^{2\pi i \frac{\bar{t}\omega}{N}}.$$

[2] The polygamma functions are defined as $\psi^{(n)} \equiv \frac{\mathrm{d}^{n+1}}{\mathrm{d}z^{n+1}} \log\Gamma(z)$ where $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,dt$, $\Re(z) > 0$ is the gamma function.

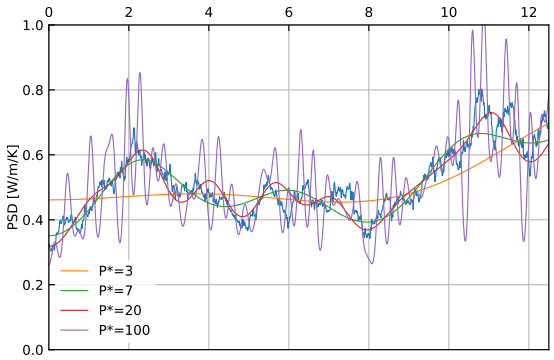


Figure 2.3: Example filtered multicomponent spectrum with a variable number of cepstral coefficients for the water-ethanol system. When the coefficients are not enough ($P^* = 3$), the filtering procedure can produce systematic errors. When the number of coefficients is too high ($P^* = 100$), noise is present inside the filtered spectrum.

so the estimator whose expectation value is $\log(\kappa(\omega))$ is

$$\hat{\mathcal{L}}(\omega) = -\bar{L}_0 + \sum_{\bar{t}=0}^{N-1} \hat{C}(\bar{t}) \mathrm{e}^{2\pi i \frac{\bar{t}\omega}{N}} \tag{2.26}$$

The assumption that $\kappa(\omega)$ is a smooth function implies that the number of significant cepstral coefficients $\hat{C}(\bar{t})$ in the cepstral expansion of Eq. (2.19) is hopefully *small*, allowing us to implement here a low-pass filter. We thus indicate by P^* a small integer such that

$$C(\bar{t}) \approx 0 \qquad \text{for} \qquad P^* \leq \bar{t} \leq N - P^*. \tag{2.27}$$

Since we are interested in the $\omega = 0$ component of (2.26) and Eq. (2.27), we must consider the stochastic variable

$$\hat{\mathcal{L}}(0) = -\bar{L}_0 + \hat{C}(0) + 2 \sum_{\bar{t}=1}^{P^*-1} \hat{C}(\bar{t}), \tag{2.28}$$

whose expectation value and variance are

$$\begin{aligned} \langle \hat{\mathcal{L}}(\omega) \rangle &= \log\left(\kappa(\omega)\right) \\ \mathrm{var}\left(\hat{\mathcal{L}}(\omega)\right) &= \frac{4P^* - 2}{N} \sigma_0^2. \end{aligned} \tag{2.29}$$

where the variance is computed by summing all independent gaussian random variables $\hat{C}(0)$ and $\hat{C}(\bar{t})$ whose variance is written in Eq.(2.25). $\hat{\mathcal{L}}(\omega)$ is thus an *unbiased estimator* for $\log(\kappa(\omega))$, and hence, of the log of thermal conductivity, we are after. At fixed P^*, the variance of $\hat{\mathcal{L}}(\omega)$ vanishes in the $N \to \infty$ limit, thus making the estimator consistent. Of course, any finite choice of P^* introduces a bias, which is a decreasing function of P^*, while the statistical error is an increasing function of it. Its optimal value is the one that makes the bias of the order of the statistical error. By adopting this value in the $N \to \infty$ limit both the bias and the statistical error can be made arbitrarily small.

In figure 2.3, we can see how the filter behaves with a variable number of cepstral coefficients. An estimate of the "optimal" value for P^* can be obtained with any of the many *model selection* techniques available in the statistics literature [15]; in our applications, we choose to adopt the Akaike information criterion (AIC) [1, 2, 19], even if other more sophisticated, and possibly more efficient, methods could be devised.

2.3 Non-diffusing currents decorrelation

The cepstral filtering procedure is more efficient if $\kappa(\omega)$ is smoother because the description of the function requires fewer cepstral coefficients causing the variance to be lower. But often happens that due to the arbitrary choice of the atomic potential energy, $\kappa(\omega)$ has a huge magnitude on most of the frequency and has a big fall near $\omega = 0$, creating an almost intractable function in the region where the physical meaning of $\kappa(\omega)$ resides. We can see examples of this behavior in [50, 49], where the fall sometimes amounts to 3 orders of magnitude. Here Marcolongo et al. showed that some parts of the energy current are not necessary to perform the calculation but are very high in power. Since it is always possible to add a flux $\mathbf{J}^i_{\text{n.diff.}}$ that does not diffuse (that means that it satisfy $\int_0^\infty \langle \mathbf{J}^i_{\text{n.diff.}}(0)\mathbf{J}^i_{\text{n.diff.}}(t)\rangle \, dt = 0$) in the energy current definition without changing κ, the first obvious approach was to try to remove them. Our work improved the original procedure, making it more straightforward, and it is immediately generalized to multicomponent fluids. By adding many independent $\mathbf{J}^i_{\text{n.diff.}}$ to the list of fluxes (and thus increasing M), we do not change the physical value of κ at $\omega = 0$. Still, we change all the other values at finite ω, lowering them because of the projector-like properties of the multicomponent formula. This algebraically removes all the contributions by the $\mathbf{J}^i_{\text{n.diff.}}$ currents. It is possible to use this technique also in the time domain. The result is that the oscillatory behavior of $\kappa(t)$ (where t is the upper time limit of all Green-Kubo integrals) is reduced, improving convergence properties.

3

Computer codes

In this chapter, we will review all the codes produced by the author to create a framework that can calculate real-world numbers. We start from `QEHeat`, a code to calculate the DFT energy flux, then we go to `SporTran`, to apply cepstral analysis on the data produced by `QEHeat`, and then a code that uses `AiiDA` to perform reproducible Car-Parrinello simulation in a semi-automatic way. All the codes are released to the open-source domain, and all good programming practices are implemented (documentation, test, well-commented code). The codes developed by the author and described in the chapter were used, particularly `SporTran`, in the works [59, 27, 70, 60, 26, 61, 69, 46].

3.1 QEHeat, an energy current code for DFT

[48]

The original expression of the adiabatic DFT energy flux, from [47, 50], was for a long time implemented only in a private branch of the Quantum ESPRESSO code. The code was highly experimental, and a significant amount of bash scripting was present to glue many parts of the calculation together. The code had the issue of the unnecessary utilization of a massive amount of disk I/O to transfer data from one piece of the calculation to the other, which causes multiple issues in highly parallel file system architecture with a very heavy utilization typical of HPC centers. So during the first pandemic lockdown, the author performed a massive rewriting of the code, creating a compact and easy-to-use tool in the Quantum ESPRESSO suite of computer codes that avoids all the unnecessary utilization of the file system.

`QEHeat` can be easily interfaced to read a dynamical trajectory generated with a code of choice and compute the MUB flux for the corresponding steps. `QEHeat` is already delivered with a user-friendly interface for the `cp.x` program of the Quantum ESPRESSO[22, 21, 23]. The combination of `QEHeat`, `cp.x` and the post-processing tool `SporTran`[18], designed to perform the statistical analysis needed to evaluate transport coefficients, provides a convenient framework to compute the heat conductivity of extended insulating systems—be they crystalline, amorphous, or liquid—entirely from first principles. We warn that the MUB flux is an *adiabatic* energy flux. This refers to the fact that electrons are supposed to populate the ground state during time evolution, which is the case for insulators with a finite band gap.

We note that other computational efforts have been proposed to evaluate microscopic expressions for the energy flux based on DFT. For example, Kang et. al [36] evaluate an atomic decomposition of the total energy and then use a two-step procedure to take into account periodic boundary conditions when evaluating the energy flux. Carbogno et. al [12] neglect the convective contribution to the energy flux and use a DFT-based expression for the virial component. This approach is developed as an approximation suited to the description of thermal transport in solids. The open-source distribution of `QEHeat` will make it easier to compare the computational advantages of the various expressions which, after the assessment of the principle of gauge invariance, are being developed by the community. Finally, the evaluation of the energy flux provided by `QEHeat` involves a finite difference evaluation of electronic properties, e.g. electronic densities or potentials, which are performed entirely in-memory, storing the results of different total energy calculations into ad-hoc data structures. Therefore, `QEHeat` provides as well to developers a modular and easily extendable framework to evaluate time derivatives of electronic properties via a finite difference approach.

3.1.1 Overview

`QEHeat` serves the purpose of evaluating the energy flux $\boldsymbol{J}$ at the DFT level of theory. A formal expression can be obtained by integrating by parts the continuity equation $\dot{\epsilon} = -\nabla \cdot \boldsymbol{j}$ as [5]:

$$\boldsymbol{J} = \int_{\Omega} d\boldsymbol{r} \dot{\epsilon}(\boldsymbol{r}) \boldsymbol{r}, \tag{3.1}$$

where $\epsilon(\boldsymbol{r})$ is the energy density of the system, whose integral is its total energy. Strictly speaking, Eq. (3.1) is ill-defined in periodic boundary conditions (PBC), which are commonly adopted in molecular simulations. In order to compute it explicitly within PBC, one has to first recast it in a boundary-insensitive form, and Ω can then be replaced with the volume of the simulation cell in Eqs. (1-3.1). Once this is done, the DFT energy flux can be cast into the MUB form, called here $\boldsymbol{J}^{MUB}$, and discussed in detail in later sections.

`QEHeat` computes the MUB energy flux as a function of the atomic positions, $\{\boldsymbol{R}_s\}$ and velocities, $\{\boldsymbol{V}_s\}$, *i.e.* for any selected snapshot of an ab-initio molecular dynamics (AIMD) trajectory. Despite the complexity of the resulting formula for the energy current, from a practical point of view the use of `QEHeat` relies on a limited number of additional input parameters with respect to a standard Quantum ESPRESSO DFT computation. These are reported in the `energy_current` input namelist, which is shown in figure 3.1. The meaning of all the keywords is explained in more detail in section 3.1.4. The only additional parameters are `eta` and `n_max`, controlling the Ewald summations, which appear only in classical contributions to the energy current, and `delta_t`, a time-discretization parameter, used to perform numerical derivatives. The default values should work for most systems.

3.1.2 DFT energy flux

In this section we recall the expression of MUB energy flux [50] and some of the notation used. For a more extensive and detailed study of the implementations of the many components of the MUB current the reader is referred to [48, Appendix B]. $\boldsymbol{J}^{MUB}$ is expressed as a sum of five components:

$$\boldsymbol{J}^{MUB} = \boldsymbol{J}^{KS} + \boldsymbol{J}^{0} + \boldsymbol{J}^{n} + \boldsymbol{J}^{H} + \boldsymbol{J}^{XC}, \tag{3.2}$$

```
    &energy_current
    delta_t = 1.000,
    file_output = 'current_hz',
    eta = 0.100,
    n_max = 5,
    trajdir = 'traj/cp',
    first_step = 1,
    vel_input_units = 'CP'
/
```

Figure 3.1: Example of the energy_current namelist. delta_t is the time used for numerical derivatives. eta and n_max are the parameters used to converge Ewald sums. trajdir is the prefix of the trajectory files. In this example, the program reads the files "traj/cp.pos" and "traj/cp.vel". first_step tell the program the first step id to compute. The step ids are part of the trajectory file format. After this namelist the full pw.x input is required. A full example of the input and the documentation of the keywords can be found at https://gitlab.com/QEF/q-e/-/tree/master/QEHeat/Doc.

where

$$\boldsymbol{J}^{KS} = \sum_{v} \left(\langle \varphi_v | \hat{\boldsymbol{r}} \hat{H}^{KS} | \dot{\varphi}_v \rangle + \varepsilon_v \langle \dot{\varphi}_v | \hat{\boldsymbol{r}} | \varphi_v \rangle \right), \tag{3.3}$$

$$\boldsymbol{J}^{0} = \sum_{s\boldsymbol{L}} \sum_{v} \left\langle \varphi_v \left| (\hat{\boldsymbol{r}} - \boldsymbol{R}_s - \boldsymbol{L}) \left(\boldsymbol{V}_s \cdot \nabla_{s\boldsymbol{L}} \hat{v}^0 \right) \right| \varphi_v \right\rangle, \tag{3.4}$$

$$\boldsymbol{J}^{n} = \sum_{s} \left[\boldsymbol{V}_s e_s^0 + \sum_{t \neq s} \sum_{L} (\boldsymbol{R}_s - \boldsymbol{R}_t - \boldsymbol{L}) \left(\boldsymbol{V}_t \cdot \nabla_{t\boldsymbol{L}} w_s \right) - \sum_{L \neq 0} \boldsymbol{L} \left(\boldsymbol{V}_s \cdot \nabla_{s\boldsymbol{L}} w_s \right) \right] \tag{3.5}$$

$$\boldsymbol{J}^{H} = \frac{1}{4\pi e^2} \int \dot{v}^H(\boldsymbol{r}) \nabla v^H(\boldsymbol{r}) d\boldsymbol{r}, \tag{3.6}$$

$$\boldsymbol{J}^{XC} = \begin{cases} 0 & \text{(LDA)} \\ -\int n(\boldsymbol{r}) \dot{n}(\boldsymbol{r}) \boldsymbol{\partial} \epsilon^{GGA}(\boldsymbol{r}) d\boldsymbol{r} & \text{(GGA).} \end{cases} \tag{3.7}$$

In the following, these components are referred to as the *Kohn-Sham*, *Zero*, *Ionic*, *Hartree* and *Exchange-Correlation* fluxes, respectively. For the insulating systems of interest in this work, the ionic degrees of freedom completely define the state of the system and the electrons populate the ground state, according to the adiabatic approximation. Each time derivative, indicated with the usual *dot* operator, has then to be understood from the implicit dependence on the atomic positions. We note that a complete understanding of the different components of the MUB energy flux is not needed to perform, as a user, a thermal-conductivity calculation. Here and in the following, we indicate with $\boldsymbol{L}$ the lattice vector. We stress that periodic boundary conditions (PBC) are assumed here over the unit (simulation) cell. This implies that Kohn-Sham orbitals and energies are sampled at the Γ point of the Brillouin cell. The notation $\nabla_{s\boldsymbol{L}}$ is a shorthand for the gradient with respect to displacement of the atom at location $\boldsymbol{R}_s + \boldsymbol{L}$. A summation over s runs over all atoms belonging to the simulation cell. Unless otherwise specified, carets indicate quantum-mechanical operators, as in $\hat{H}^{KS}$ or $\hat{\boldsymbol{r}}$. Following is a brief report of the definition of the most important terms according to their physical meaning. The

ionic energy e_s^0 is the sum of the kinetic energy, $\frac{1}{2}M_sV_s^2$, and w_s the classical electrostatic interaction between the s-atom and all other atoms in the system. The electronic degrees of freedom are instead described by the instantaneous Kohn-Sham Hamiltonian, $\hat{H}^{KS}$, and its eigenvalues and eigenvectors, the Kohn-Sham energies, ϵ_v, and orbitals, φ_v. A summation over v runs over all occupied orbitals. Several quantities, like the electronic number-density distribution, $n(\boldsymbol{r})$, and Hartree potential, $v^H(\boldsymbol{r})$, are implicit functions of the wave-functions. $\hat{v}^0$ is the total external atomic pseudo-potential, describing the interaction between electrons and nuclei. For a more detailed analysis see [48, Appendix B.2]. The symbol ϵ^{GGA} stands for the generalized gradient approximation (GGA) exchange-correlation local energy per particle and its derivative with respect to density gradients is indicated with $\boldsymbol{\partial}\epsilon^{GGA}$, which is a vector whose component along direction $i \in \{x, y, z\}$ is given by $\partial\epsilon^{GGA}(n, \nabla n)/\partial(\nabla_i n)$. In the present version of `QEHeat` only the local density approximation (LDA) and the generalized gradient approximation given by the PBE [63] functional are implemented.

The formulas reported are compatible with PBC. Thus, they can be implemented for periodic systems, where some of the summations need to be extended to all the periodic replicas of the atoms. Nevertheless, the computation of the various contributions to the energy current, Eqs. (3.3-3.7), is plagued by the occurrence of several divergences, arising from the long range character of the Coulomb interaction. As it is the case for the total energies, atomic forces, and stress, the individual electronic, ionic, and electron-ion contributions diverge and it is only their sum that is regular in the thermodynamic limit. In order to regularize the individual components of the MUB flux, we compute all the relevant terms by screening the Coulomb interaction with a Yukawa cutoff, $\frac{1}{x} \to \frac{e^{-\mu x}}{x}$. In [48, Appendix B], we check explicitly that the singular contributions to the various terms cancel each other in the $\mu \to 0$ limit, so that they can be consistently and safely neglected and do not appear in the final formulas reported in [48, Appendix B].

3.1.3 Code structure

We start by describing the strategy implemented to compute numerical derivatives of quantities appearing in Eqs. (3.3), (3.6), and (3.7), like $\dot{n}$ and $\dot{v}^H$, since they require special treatment. Quite generally, one needs to evaluate terms of the type $\dot{f}(\{\boldsymbol{R}_s(t)\})$, where the function f can be a scalar function, which depends on time only through the set of the instantaneous ionic positions $\{\boldsymbol{R}_s(t)\}$, evolving according to Hamilton's equations of motion. `QEHeat` implements a finite-difference scheme, using by default a symmetric numerical differentiation formula:

$$\dot{f}(\{\boldsymbol{R}_s\}) \approx \frac{f(\{\boldsymbol{R}_s + \boldsymbol{V}_s dt/2\}) - f(\{\boldsymbol{R}_s - \boldsymbol{V}_s dt/2\})}{dt} \tag{3.8}$$

The small parameter dt is an input of the computation. In such a scheme quantities that are not differentiated are evaluated at time t, so three wave-functions are required to be kept in memory at the same time. `QEHeat` performs therefore for each step two additional self-consistent-field (SCF) DFT calculations, using the same DFT solver of the Quantum ESPRESSO distribution, at slightly displaced positions, i.e. $\{\boldsymbol{R}_s - \boldsymbol{V}_s dt/2\}$ and $\{\boldsymbol{R}_s + \boldsymbol{V}_s dt/2\}$ along the AIMD trajectory. The wave-functions of the previous calculation are used as a starting point for the next one, which requires much fewer iterations to converge. We note that `QEHeat` gives the user also the possibility to use a non-symmetric differentiation scheme, which is shown in [48, Appendix E]. This scheme is computationally cheaper. Nevertheless, for differentiable functions, the order of convergence of

the symmetric scheme is quadratic in dt, whereas the non-symmetric one is linear. Accordingly, the stability is improved with the default symmetric scheme. We recommend therefore the latter and use it for all calculations here presented.

The trajectory data is managed by the Fortran derived data type `cpv_trajectory` defined in the file `cpv_traj.f90`, while the orbitals and the associated atomic position are managed by the derived type `scf_result`, implemented in `scf_result.f90`. The most relevant subroutine that acts on this object is `scf_result_set_from_global_variable`, which copies the eigenfunctions, the eigenvalues, the potential, and the atomic positions from the QUANTUM ESPRESSO's global variables to the instance of `scf_result`. The results for each of the three (or two) wave-functions that are required by the computation routines are stored in the variable `scf_all`, defined in the main program routine. Global variables are avoided as much as possible.

The code starts by reading the input "namelists": first the `energy_current` namelist, then all the `pw.x` namelists. Then it calls all the `pw.x`-related initialization routines. After eventually reading the previously generated output file that allows the program to set the correct starting timestep, it enters the main loop over the input trajectory timesteps. The trajectory files have the same format of QUANTUM ESPRESSO's `cp.x` code output files.

The most important routines where the above-mentioned data structures are used are the following:

- `SUBROUTINE current_zero` (module `zero_mod`)
  Carries out the computation of 3.4. This routine is called in the middle of the computation using the same timestep t of the positions stored in the input trajectory, so that the result does not depend on dt.

- `SUBROUTINE current_ionic` (module `ionic_mod`)
  Computes all parts of 3.5, and it is called as `current_zero` at the same timestep of the input trajectory

- `SUBROUTINE current_hartree_xc` (module `hartree_xc_mod`)
  Computes (3.6) and (3.7). Since a numerical derivative is needed, this routine reads the wave-functions from the global type `scf_all` and it is run at the end of all necessary `run_pwscf` calls.

- `SUBROUTINE current_kohn_sham` (module `kohn_sham_mod`)
  Computes (3.3). As `current_hartree_xc`, it needs all the wave-functions calculated by the DFT solver for this step.

- `SUBROUTINE run_pwscf`
  Uses QUANTUM ESPRESSO's routines to solve the DFT problem for the atomic positions stored in the global array `tau`. Equivalent (but the starting wave-function and potential, that can be the last computed one) to a standard call to the `pw.x` program with the input stripped of the `ENERGY_CURRENT` namelist. The result is stored in the QUANTUM ESPRESSO's global arrays (`evc`)

- `SUBROUTINE prepare_next_step`
  This routine is used to change the global array `tau` to `tau` + `vel`·dt·`ipm`, where `ipm` is the argument of the subroutine that can be -1,0,1. After doing that it calls the necessary routines to prepare the potential for `run_pwscf`.

The 4 modules, one for each part of the MUB current, are completely independent of one another. The structure of the main loop over the trajectory's time steps is summarized in algorithm 1.

Algorithm 1 Workflow of `all_currents.f90`.

1: QUANTUM ESPRESSO initialization (plane waves, pseudo-potentials,...)
2: Reading of Restart
3: **for** each snapshot **do**
4: call `run_pwscf` with positions displaced at $t - dt/2$
5: call `run_pwscf` with non-displaced positions at t
6: call `current_zero`, evaluate currents derived from the pseudo-potential
7: call `current_ionic`, evaluate the electrostatic and kinetic Ionic current
8: call `run_pwscf` with positions displaced at $t + dt/2$
9: call `current_hartree_xc`, evaluate Excange and Hartree currents
10: call `current_kohn_sham`, evaluate Kohn-Sham current

Steps 6 and 7 do not require any finite differences, while steps 9 and 10 do. Step 10 is the most expensive.

As with every big computational code, an extended test suite is needed to safeguard the correctness of the calculation after every source code modification. We implemented small tests that are able to run on a single core of a cheap laptop that check against changes in the numerical output of many parts of the code, using the standard QUANTUM ESPRESSO's test suite framework.

To conclude the section we want to do some remarks on the code and its interactivity with other typical *ab-initio* simulations tools. In principle the wave-functions computed on-the-fly by `cp.x` during the AIMD run could be used, but we preferred to implement a workflow where the computation of the currents is completely decoupled from the AIMD engine, thus the wave-functions are always recomputed by `pw.x`. The chosen approach allows the user to run the calculation in post-processing mode, thus using the preferred code to generate the dynamics, not to be limited to those in the QE packages. This way it allows, also, a trivial and powerful per-snapshot parallelization.

3.1.4 Code usage

Input description

The input is organized in a traditional FORTRAN namelist input file, similar to the input files of many QUANTUM ESPRESSO's programs, and an optional trajectory file (that is a file for the atomic velocities and a file for atomic positions) if the user wants to compute the energy current for more than one snapshot with a single run. A full example of the input can be found at https://gitlab.com/QEF/q-e/-/tree/master/QEHeat/examples. Before running `QEHeat` it is necessary to obtain velocities and positions from a different code. If the QUANTUM ESPRESSO's `cp.x` program is used for this purpose, its output trajectory files can be recycled as input trajectory files without any modification. The program's mandatory input is organized into an `ENERGY_CURRENT` namelist and all the usual `pw.x` namelists. We remind the user that, up to the present version, only norm-conserving pseudopotentials and the PBE exchange-correlation functional are supported. At the end of the input file the `ATOMIC_VELOCITIES` card is required. In the `IONS` namelists the value `ion_velocities = 'from_input'` is required, since the program must read the

atomic velocities to compute the energy current. An extensive input description can be found in the file `INPUT_ALL_CURRENTS.html` in the `Doc` folder of the code repository. Here we remark on the most important parameters of the `ENERGY_CURRENT` namelist:

- `delta_t` : time in PW's atomic unit used to compute all the numerical derivatives like the one in Eq. (3.8);
- `trajdir` : prefix of the cp-formatted trajectory. Optional: if not set, only the positions and the velocities of the input file are read;
- `n_max` : the number of periodic images along the directions of each basis cell vector to converge Ewald sums. This fixes the range of $\mathbf{L}$ in Eq. (3.5) ;
- `eta` : convergence parameter of the Ewald sums needed in the computation of $\boldsymbol{J}^n$, for more details see [48, Appendix B.3]

An example of the namelist is provided in fig. 3.1. An additional output file is written and updated at the end of each step in the folder where the program is run. All the currents are printed in a column format, ready to be analyzed by an external post-processing tool.

`CONTROL`'s `conv_thr` and `ENERGY_CURRENT`'s `delta_t` have a profound link and influence heavily each other, and despite we think the default value of `delta_t=1.0` is safe enough, they must be carefully tested, verifying that the standard deviation of the result is low enough.

The standard deviation of the output energy current can be estimated by repeating the same calculation for every step, many different times, setting for each repetition a random starting potential and a random starting wave-function. The input options `re_init_wfc_1 = .true.` together with `n_repeat_every_step = 20`, for example, do 20 repetitions of every timestep, resetting the starting wave-functions/potential before the first SCF calculation. The `pw.x`'s input option `startingwfc = 'random'` is suggested, to obtain a faithful error estimation. If more reinitializations are desired, the options `re_init_wfc_2` and `re_init_wfc_3` can control the randomness of the starting WFC and potential of every of the 3 (or 2) wave-functions needed to perform the numerical derivatives, as explained in section 3.1.3. Note that when the wave-function is reinitialized from scratch, the computation time raises since more SCF cycles are required to reach the target convergence threshold. When `n_repeat_every_step` is greater than 1, an additional column formatted output file with the averages and the standard deviations is produced.

3.1.5 Implementation checks: Finite systems translating at constant speed

The Green-Kubo current associated with a localized energy density $\epsilon(\boldsymbol{r}, t)$ rigidly translating with constant velocity $\boldsymbol{v}$, is equal to $E^{tot} \times \boldsymbol{v}$. One possible way to show this is to consider $\epsilon(\boldsymbol{r}, t) = \epsilon(\boldsymbol{r} - \boldsymbol{v}t, 0) \equiv \epsilon^0(\boldsymbol{r} - \boldsymbol{v}t)$,thus $J_a = \int \dot{\epsilon} r_a d\boldsymbol{r} = -v_b \int (\partial_b \epsilon^0) r_a d\boldsymbol{r} = v_a \int \epsilon^0 d\boldsymbol{r} = E^{tot} v_a$. Note that we used the fact that ϵ^0 can be taken identically equal to zero at the boundary of the integration volume, to remove boundary contributions from the integration by parts. The identity requires therefore the energy density to be localized and this condition can be mimicked in PBC considering a large enough supercell. We used this property to check the correctness of our implementation for each individual current in 3.2. We simulate a single Argon atom and a water molecule at equilibrium, both translating at a constant speed. We then compare $\boldsymbol{J}^{MUB}$ output from `QEHeat` and $E^{tot}\boldsymbol{v}$,

	parameter	units
INPUT	dt	$\tau_{a.u.}$
	eta	$1/a_0^2$
	velocities	$a_0/\tau_{a.u.}$ (CP units can be specified)
	positions	a_0
OUTPUT	energy current	$Ry \cdot a_0/\tau_{a.u.}$
	electronic density current	$a_0/\tau_{a.u.}$
	center of mass currents	$a_0/\tau_{a.u.}$

Table 3.1: Units used for the input and the output, where $Ry = 2.1799 \cdot 10^{-18}J = 13.606eV$, $a_0 = 5.2918 \cdot 10^{-11}m$, $\tau_{a.u.} = 4.8378 \cdot 10^{-17}s$ are the Rydberg units of energy, the Bohr radius and the time unit in Rydberg atomic units. The program assumes the input velocities to be in Rydberg atomic units, the standard for `pw.x`, unless specified otherwise with vel_input_units='CP' in the energy_current namelist. In that case, it assumes Hartree atomic units, the standard for `cp.x`. `cp.x`'s unit of time is $2.4189 \cdot 10^{-17}s$.

where E^{tot} is evaluated using an independent computation from the QE code. TheAs discussed, the resulting currents need to be equal only in the limit of large cells, where boundary effects can be neglected, i.e. the energy density is truly localized, and under tight convergence criteria. In Fig. 3.2, we report the ratio between the computed and theoretical values as a function of the cell parameter, showing that the correct limit behavior is recovered. For this calculation, we used a cutoff of $120Ry$ and econv $= 10^{-14}Ry$. Additionally, in Appendix C we perform the same test using a large cell parameter but removing individual current components (i.e. the {XC,IONIC,ZERO,KOHN} components) from the total energy flux. Since removing each component changes the difference with respect to the limiting theoretical value, this proves the correct implementation of each individual current.

In the same figure, using the same approach, we tested the electronic density current as well. In the infinite cell limit the electron density current of a system translating at a constant speed $\boldsymbol{v}$ is $\boldsymbol{J}^{el} = N_e\boldsymbol{v}$ where N_e is the number of electrons. It is possible to see in figure 3.2 that the correct limit is obtained, validating the implementation of the code.

3.2 `SporTran`, a data analysis code for Green-Kubo transport coefficients

[18]

`SporTran` is designed to perform a complete cepstral analysis of a possibly multivariate current time series generated by EMD, without the need of implementing, or even fully mastering, all the details of the protocol. Moreover, an easy-to-use and multi-platform Graphical User Interface (GUI) is provided, allowing the inexperienced user to input all the needed parameters and visualize all the intermediate results in an interactive and easy-to-understand way. In addition, the package is equipped with a command-line interface, permitting one to run the program on headless computers or easily embed it in scripts. Finally, a flexible Python API is provided, giving more advanced users complete control over all the parameters and intermediate steps of the protocol.

The `SporTran` package requires at least Python 3 and the numerical library `numpy` [31] for its core functionalities. In addition to this, the complete package requires `scipy`, `matplotlib`, `markdown2` and `pillow`. It can be installed by cloning the GitHub repository

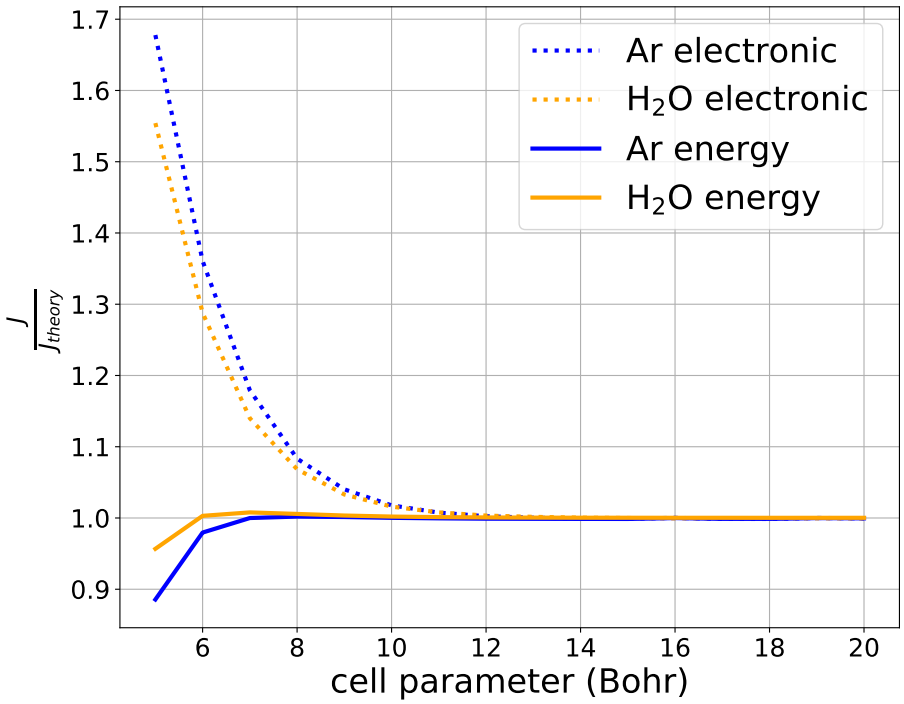


Figure 3.2: Proof that the behavior of the code is correct for systems translating at equilibrium and constant velocity. In this setting, the output of QEHeat can be exactly compared with the known theoretical value, indicated with J_{theory}, in the large cell limit. Indeed the plot shows that the ratio J/J_{theory} goes to one increasing the simulation cell. This test can be performed for J equal to the electronic density current (dotted), which is used to calculate a part of the energy current, and for the energy current itself (not dotted). Tests are performed for a single relaxed water molecule (H_2O) and a single argon atom (Ar). The electronic current should be, in the infinite cell limit, $\boldsymbol{J}^{el}_{\text{theory}} = N_{el} \times \boldsymbol{v}$, where N_{el} is the number of electrons and the energy flux should be, in the infinite cell limit, equal to $\boldsymbol{J}^{ene}_{theory} = E^{tot} \times \boldsymbol{v}$, E^{tot} being the total energy.

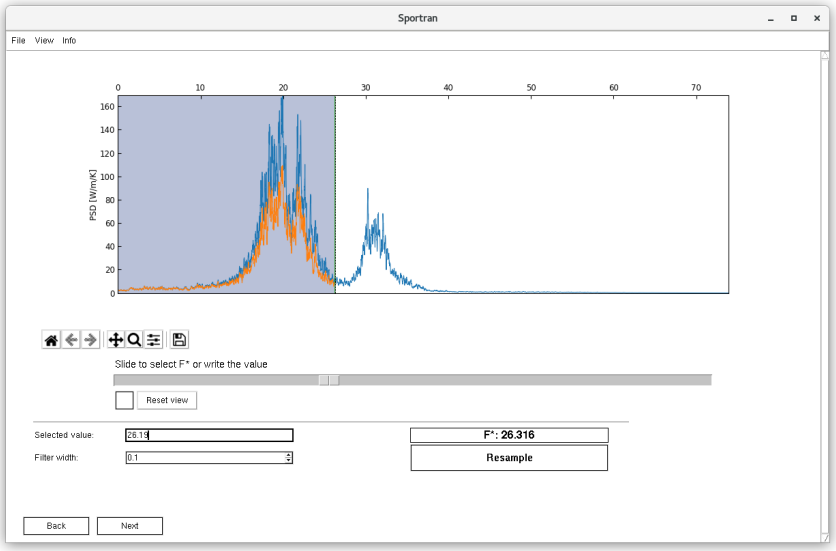


Figure 3.3: Selection of the Nyqvist frequency, f^*. The figure shows the power spectrum of the energy current of a silica glass sample provided in the examples.

[17] or by using the `pip` utility (`pip install sportran`).

3.2.1 Graphical User Interface

The GUI is designed to smoothly drive the user from the raw data to the final result. The input can be a simple column-formatted text file, with a text header for every column, or a NumPy/pickle binary file containing a dictionary. The user selects the file format and is then guided through a few simple steps to load the file, select the desired currents, set the physical constants, and determine the value of the Nyqvist frequency, f^*, as well as the optimal number of cepstral coefficients, P^*. The latter step is actually performed automatically through the Akaike information criterion [1, 2], whose suggestion can however be manually adjusted by the user. In Fig. 3.3 we display a screenshot of the selection of the Nyqvist frequency.

3.2.2 Command-Line Interface

The command-line interface is designed to easily embed the complete analysis in scripts. All the parameters have to be specified as command line arguments (see Tab. 3.2). The program outputs a PDF file with several plots, and textual or binary files containing the raw results of the calculation. Those files can be easily used for further analysis.

3.2.3 Python API description

Here we provide an example of the usage of the code as a Python library. In this example, which is also provided in the git repository [17] and documentation, we perform the analysis of the energy current time series of a molten salt, sodium chloride. We stress that the procedure is the same for any type of transport coefficient calculation. The data is contained in a plain text file that was extracted from the output of a LAMMPS [64] simulation, formatted as follows:

`--input-format {table,dict,lammps}`	input format
`-k MAINFLUXKEY`	the header of the J_0 flux column
`-j ADD_CURRENTS`	header of additional current (optional)
`-C {electric,heat,stress}`	the type of current
`-u {real,metal,qepw,GPa...}`	units
`-t TIMESTEP`	timestep in fs
`--VOLUME VOLUME`	volume of the system in Å^3
`--TEMPERATURE TEMPERATURE`	temperature of the system in K
`-r`	resample the time series with the specified FSTAR
`--FSTAR FSTAR`	maximum frequency to analyse in THz
`-w PSD_FILTERW`	size of the moving average filter used in the plots, for visualization purposes (THz)
`--help`	show the complete help
`--list-currents`	list all the currents and units implemented in the code
`INPUTFILE`	input file

Table 3.2: List of important command line parameters.

```
Temp  c_flux[1] c_flux[2] c_flux[3] c_vcm[1][1] c_vcm[1][2] c_vcm[1][3]
1442.7319  250.86549 20.619423 200.115 -0.15991832 -0.071370426 0.020687917
1440.8060  196.22265 82.667342 284.3325 -0.13755206 -0.071002931 -0.011279876
...
```

Note the first line and the LAMMPS-like notation to define vector components. Besides the fluxes, there can be additional columns with scalar data, *e.g.* the temperature. In this particular case, the first vector quantity, named `c_flux`, is the energy current, while the second vector quantity, named `c_vcm[1]`, is the velocity of the center of mass of the sodium atoms. This file contains time series that can be generated by any MD code. The API provides a generic module to read this kind of data file:

```
import sportran as st
jfile = st.i_o.TableFile('./examples/data/NaCl.dat', group_vectors=True)
jfile.read_datalines(
       start_step=0, NSTEPS=0, select_ckeys=['Temp', 'flux', 'vcm[1]'])
```

The `select_ckeys` argument must be set with the names of the desired columns. The code will automatically read them as Cartesian components of a vector-valued process. Please refer to the documentation for additional input formats.

We are now ready to initialize a `Current`-type object. `Current` is an abstract class that serves as a template for different types of currents. The `GenericCurrent` class, derived from `Current`, defines a generic current time series: in order to estimate the transport coefficient, only the time step and the κ_{scale} factor must be defined. To streamline the user workflow, a few other specialized subclasses are available: `HeatCurrent` (thermal conductivity), `ElectrictCurrent` (electrical conductivity), `StressCurrent` (viscosity), each corresponding to different types of transport processes. For each current type several units are available,[1] that simply define the κ_{scale} factor. In the case of this example:

```
DT_FS = 5.0            # time step [fs]
TEMPERATURE = np.mean(jfile.data['Temp'])  # mean temperature [K]
VOLUME = 40.21**3      # volume [A^3]
```

[1] A list of units for each current subclass can be printed: *e.g.* `print(sp.current.HeatCurrent.get_units_list())`.

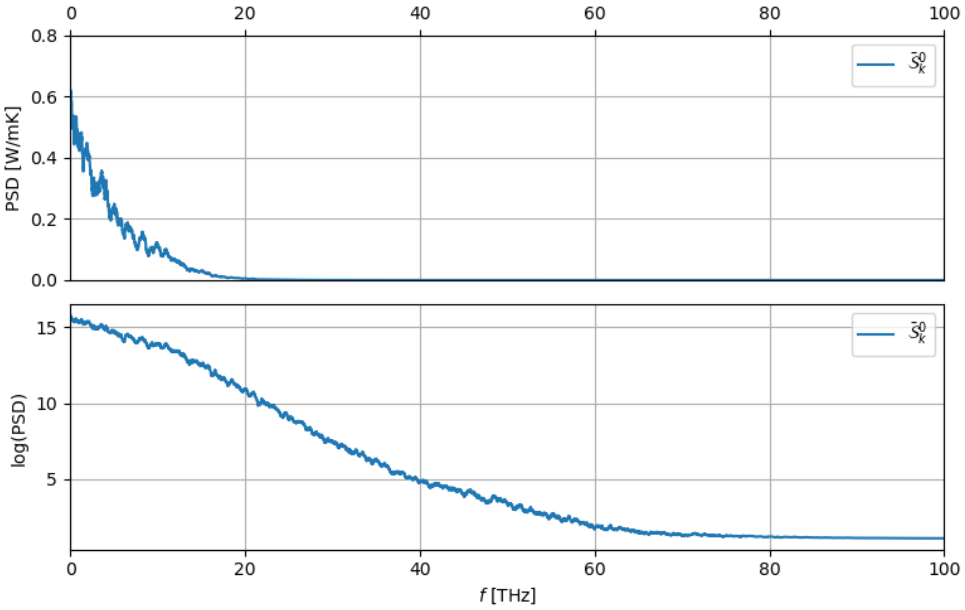


Figure 3.4: Plot of $\bar{\mathcal{S}}^0_k$ and $\log \bar{\mathcal{S}}^0_k$ generated using the class method `HeatCurrent.plot_periodogram()`.

```
j = st.HeatCurrent([jfile.data['flux'], jfile.data['vcm[1]']],
        UNITS='metal', DT_FS=DT_FS, TEMPERATURE=TEMPERATURE, VOLUME=VOLUME)
```

`metal` are the units used to compute the heat flux (as defined in LAMMPS [64]). The input parameters needed to define each `Current` subclass are listed in the code documentation.

We are now in the position to compute the sample $\bar{\mathcal{S}}^0_k$. Remember again that in the univariate case this is simply the periodogram of the original time series. In order to plot $\bar{\mathcal{S}}^0_k$ and its logarithm we can use the following function:

```
ax = j.plot_periodogram(PSD_FILTER_W=0.4, kappa_units=True,
                        label=r'$\bar{\mathcal{S}}^0_k$')
```

`PSD_FILTER_W` defines the width in THz of a moving average filter used for visualization purposes. The result is shown in Fig. 3.4. Since we are interested in the zero-frequency value of $\bar{\mathcal{S}}^0_k$, we resample the time series in order to decrease the Nyqvist frequency to f^* and focus on the lower part of the spectrum. We do this as follows:

```
FSTAR_THZ = 14.0
jf, ax = j.resample(fstar_THz=FSTAR_THZ, plot=True, freq_units='thz')
ax.set_xlim([0, 20])
```

The resulting plot is shown in Fig. 3.5. We are now ready to perform the cepstral analysis. This is as simple as calling the function:

```
jf.cepstral_analysis()
```

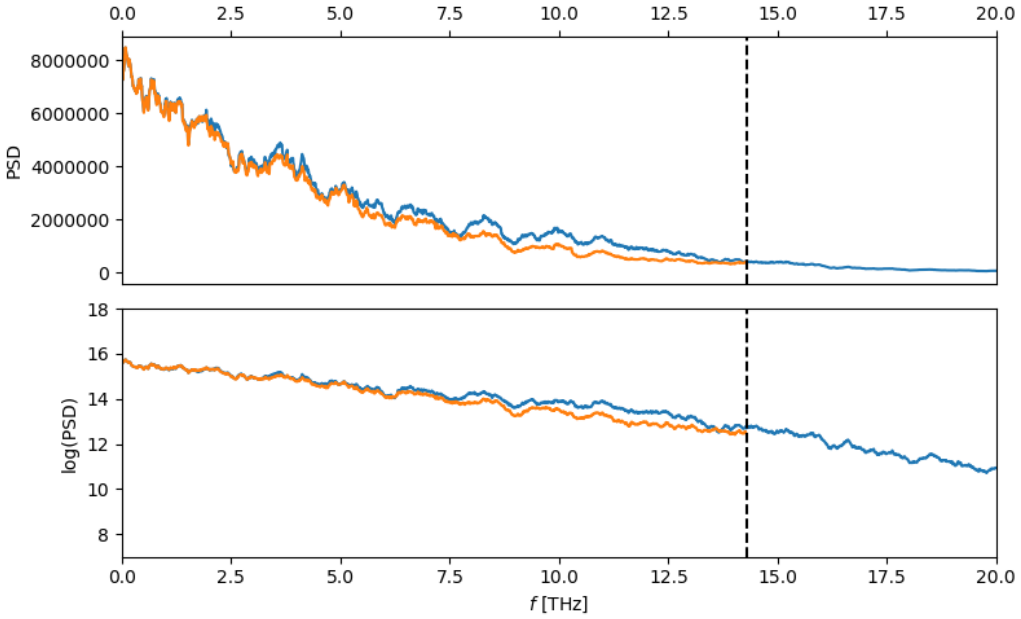


Figure 3.5: Resampled time series. The blue and orange lines represent $\bar{\mathcal{S}}_k^0$ of the original and the resampled time series, respectively.

The result provided is the following:

```
-----------------------------------------------------
CEPSTRAL ANALYSIS
-----------------------------------------------------
AIC_Kmin  = 3  (P* = 4, corr_factor = 1.000000)
L_0*   =          15.158757 +/-   0.056227
S_0*   =     6824108.702608 +/- 383697.095268
-----------------------------------------------------
kappa* =           0.498310 +/-   0.028018  W/mK
-----------------------------------------------------
```

For additional details please refer to the Jupyter notebook examples provided with the source code, or to the documentation.

All the steps and plots of this workflow are implemented in the command line tool and in the graphical user interface, which streamlines the execution for the end user.

Extending `SporTran` and code details

We remark again that the core part of the code is process-agnostic, *i.e.* it does not depend on the type of transport coefficient and can be seen as a way to compute *any* Green-Kubo integral from a flux time series. However, when performing an actual calculation, the ability to set the physical units can be very practical. For this reason, `SporTran` is designed in a way that makes it easy to add new custom units and transport coefficients that can be expressed as in Eq. (1.50), where physical coefficients such as temperature and volume are multiplied. This enables the user to input all the required parameters in a simple and friendly manner through all the available interfaces, and to get back the

result in the preferred units. If the following procedure is used, all the built-in interfaces (GUI and CLI) will detect the new parameters and request them when needed, thanks to the introspection features of the Python language.

Adding a new unit is very easy: it is sufficient to go to the folder `sportran/current/units`, open the module corresponding to the desired current type (heat, electric, stress, ...) and add a function named `kappa_scale_mynewunitname`, similarly to the other functions found therein. The code will list this new unit called `mynewunitname` in every user interface and in the help function.

In order to define a new type of current and set its units, the following operations are required. First, in the folder `sportran/current` add a new module (*e.g.* `mycurrent.py`), in which a subclass of `sportran.current.Current` (*e.g.* `MyCurrent`) is defined. Then:

- Define the following class attributes:
  - `_current_type`: the name of the current. This will be used by the user interfaces to look for the available units corresponding to this current (*e.g.* `mycurrent`).
  - `_input_parameters`: a set of parameters that this class needs, *e.g.* `{'DT_FS', 'UNITS', 'TEMPERATURE', 'VOLUME'}`.
  - `_KAPPA_SI_UNITS`: a string describing the units of the transport coefficient, *e.g.* `W/m/K`.
- Define a `_builder` method (property). This is a method returning a dictionary of all the parameters needed to rebuild an identical current object.
- Define the units: create a new module in the folder `sportran/current/units` and name it after `_current_type` (*e.g.* `mycurrent.py`). Add here all the desired units as functions called `kappa_scale_myunit`, as previously explained. The input parameters of these functions must be the same listed in the `_input_parameters` class attribute.

The code has extensive documentation written with Sphinx available at https://sportran.readthedocs.io and a complete test suite that the user is strongly advised to run before using the code on a new machine, or after modifying the code. Examples are included in the package, both for the single-component and the multi-component case in the form of Jupyter notebooks and command-line scripts. Any issue reports or contributions to this code are encouraged and can be submitted to the GitHub page [17].

3.3 Automatic workflows with `cp.x` and `AiiDA`

[35, 71]

During computational work usually the simulations depend on each other. It is possible to produce a reproducible and verifiable result by hand only when the number of simulations is small, and this task requires a big amount of tedious work. `AiiDA` purpose is to manage all the data produced by a simulation and all the inputs of a simulation on a remote machine and keep track of all the dependencies for you. The full data provenance graph is stored in a relational database, so it is easy to retrieve any old simulation simply by querying the database. Then it is possible to take advantage of these facilities to code automatic workflows, that manage many simulations automatically, performing all of them using the same methodology and getting all the results in your data analysis

```
400K/
    61/
    62/
    63/
    in.61
    in.62
    in.63
    submit-cg-61.job
    resubmit-61.job
    submit-verlet-62.job
    submit-nose-63.job
    MEMENTO
    .
    .
500K/
    .
    .
    .
```

Figure 3.6: Example messy folder structure, that we try to avoid with AiiDA . Reconstructing the exact sequence of commands and getting the same result can be hard or impossible. For example the user can modify some input file and use it again, and no trace of the old file is left. Looking for a particular calculation can be difficult as well.

framework. This is an important step towards coding our knowledge in computer codes, especially in a field where doing a simulation is a sort of cooking recipe, with some fixed guidelines but no unique way of getting a correct result.

3.3.1 AiiDA overview

AiiDA has 3 principal components: a code that is called verdi that runs in the background of the user's computer interacting with the remote machines, a relational database where all the graph of all the computation is stored, and a repository of files where the outputs and inputs that are too big for the database are kept. The AiiDA 's python library manages all operations. The user needs to interact only with verdi, and with the built-in python object-relational mapping (ORM) interface to the database. All the system is built around the idea of the Node object, that is a node in the provenance graph of a computation. A Node has inputs and outputs links that point to other Nodes. It is possible to attach to a Node extra data, such as reference to the big files repository or extra comments. In the python library, the generic Node object is specialized, for example, to an input parameter, a trajectory, a computation, or a remote computer code. A Node can be sealed (for example, after a computation is finished). After this event, the node's inputs, outputs, and contents cannot change, but only new extra comments can be attached. When a graph is built with the AiiDA interface, nodes not waiting for something to finish will be automatically sealed. Thus any provenance graph built with AiiDA will represent a completely reproducible calculation, given that the executable codes on the remote computing resource are kept there.

To interact with the external computing resources, a specific code Node is needed for each different computer, and for each different code. Calculations Nodes are created by plugins provided by the AiiDA developers. Usually, they are organized in a part that validates the inputs and prepares the input files, consuming input nodes, and a part that parses the output generating new AiiDA output Nodes.

The author contributed actively to the AiiDA Quantum ESPRESSO plugin[2], implementing the support for the newer versions of the cp.x code, new parsing features, a

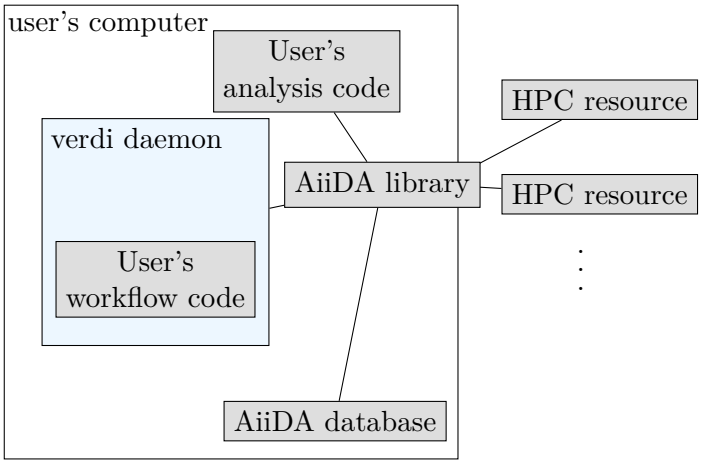
[2]See https://github.com/aiidateam/aiida-quantumespresso and the pull request https://github.com/aiidateam/aiida-quantumespresso/pull/455

Figure 3.7: Example `AiiDA` graph of a reproducible computation. The complexity is very high, and keeping track of the dependencies without an automated tool is impossible.

refactoring of many parts that allowed the code to enhance its flexibility (needed for the new features), and many new tests to the test suite. This work resulted in the modification of 17 source files and roughly a thousand lines of code.

All the operations on the remote filesystem and all communications with the queue manager of the high-performance computing resources are handled transparently by the `AiiDA` library. The remote computers and the remote codes need to be configured only once when they are used for the first time.

With all this machinery, it is possible to write a code that manages the sequence of computation of other codes, called *workflow*. All the processes of performing a complicated calculation become (semi-) automatic. An important feature is that the workflow code does not need to know anything about folders, databases, remote filesystems, lowest level details of how to build the input of a managed code, but it will interact only with the code plugin and the `AiiDA` interface to the database and repository, implementing only the highest level logic of the computation.



3.3.2 Small review of first principle molecular dynamics

[11, 51] Here we'll review the basic theory of Born-Oppenheimer (BO) and Car-Parrinello (CP) first principle molecular dynamics to understand what the semi-automatic CP code does.

In the BO method, we assume that nuclei are classical particles and that the evolution of the electrons is adiabatic. This means that the forces on the nuclei depend only on the atomic positions and not on the time derivative of the electronic wavefunctions. So a basic algorithm is to solve the following equations of motion:

$$M_I \ddot{R}_I = -\nabla_I \min_{\psi_0} \langle \psi_0 | H_e | \psi_0 \rangle \tag{3.9}$$

$$E_0 \psi_0 = H_e \psi_0 \tag{3.10}$$

where H_e is the Kohn-Sham hamiltonian and the electronic state ψ_0 is found by solving the Kohn-Sham equations at every time. To do a molecular dynamics simulation the equations are integrated over discrete time steps with the Verlet algorithm. Forces are calculated with the Hellmann-Feynman theorem after a complete minimization of the DFT energy, done at each integration time step. This can be applied for each case where the adiabatic approximation is valid, but it may not the be most efficient one. The biggest issue is the difficulties to perform a well-converged diagonalization of the hamiltonian, so that good forces are obtained and the constant of motion of the system (kinetic energy of the nuclei plus DFT energy) is conserved. This is not always easy to obtain since small non-convergence errors tend to generate drifts in the $K + V$ time series.

The Car-Parrinello (CP) method tries to overcome these difficulties and can be a good choice in some cases. The basic idea is to treat the electron wave function as a classical field that moves in the DFT potential. If the field starts near the energy minimum of the potential and its temperature is low enough, applying Newton's equation to it will let it evolve following the minimum of the DFT potential, resulting in a dynamical minimization procedure that does not need to diagonalize the KS hamiltonian at each step of the dynamic. The famous CP lagrangian:

$$\mathcal{L} = \sum_v \frac{1}{2}\mu \int_\Omega d^3r |\dot{\psi}_v|^2 + \sum_I \frac{1}{2} M_I \dot{R}_I^2 - E_{DFT}[\{\psi_v\}, \{R_I\}] + \sum_{i,j} \Lambda_{ij} \left(\int_\Omega d^3r \psi_i^* \psi_j - \delta_{ij} \right) \tag{3.11}$$

where we have a fictitious electronic classical kinetic energy, a fictitious electronic mass μ, and lagrangian multipliers Λ_{ij} used to keep the wavefunctions orthogonal during the dynamics. It can be shown that in the limit of small μ, solving the equation of motion of this lagrangian is equivalent to doing BO dynamics on the ions. The equations of motion for the system are:

$$\mu \ddot{\psi}_v(r,t) = -\frac{\delta E}{\delta \psi_v^*}(r,t) + \sum_j \Lambda_{vj} \psi_j \tag{3.12}$$

$$M_I \ddot{R}_I = -\nabla_{R_I} E_{DFT} \tag{3.13}$$

The idea is that the equations should remain decoupled: the fastest modes ω_n^{max} of the second equation should be slower than the slowest ω_e^{min} of the first, so that there is no energy transfer and the electronic wavefunctions can stay near the ground state, oscillating fast around it, for all the simulation. This decoupling can be controlled by the parameter μ thanks to the relation [51]:

$$\omega_e^{min} \propto \sqrt{\frac{E_{gap}}{\mu}} \,. \tag{3.14}$$

Since E_{gap}, the energy difference between the lowest unoccupied and the highest occupied state, depends on the physics of the system, the only parameter that we can tune is μ. The lower the μ the higher the electronic frequencies, and the better the decoupling. But you will have to use a smaller timestep to integrate the equations of motion. Doing a good simulation that conserves well enough the physical energy of the system and the constant of motion of the CP lagrangian is a matter of finding a good balance between a not-too-small integration timestep and a not-too-big μ, in systems where $E_{gap} > 0$.

The fact that we assign an arbitrary mass to the electronic wavefunctions, with the only purpose of letting them move with newton equations following the minimum of the DFT functional, also has consequences on the system's inertia. If, in a classical system, one adds something with a mass bounded to something else with a potential, one cannot expect that this blob with a higher mass moves with the same inertia as before. The practical effect of this on the simulation is that, on average, the forces calculated on the ions will be systematically lower by a factor with respect to the forces calculated with the Hellmann-Feynman theorem and a full minimization of the DFT functional, because, intuitively, part of the missing force is needed to accelerate the electrons. It is important to check for this error and try to correct it when we calculate quantities that depend on the mass of the ions, such as diffusion coefficients. At the same time, static properties like the pair correlation function should not depend on the mass of the electrons in a well-formed simulation. The effect on the forces is investigated in deeper detail in [68, 67]. The leading order additional ionic mass term in systems where electrons are localized around ions, with no distortions to the charge of the ions as they move in the field of the other ions, can be estimated with the following[68]:

$$\Delta M_I = \frac{2}{3}\frac{m_e}{\hbar^2}\sum_j \mu\langle\phi_j^I\Big| - \frac{\hbar^2}{2m_e}\nabla_j^2\Big|\phi_j^I\rangle \tag{3.15}$$

where the sum is performed over the orbitals of the ion I, and the terms in the sum are the quantum kinetic energy term of the wavefunction. This correction can be understood as additional inertia due to ions carrying around a rigid cloud of massive electrons. This term vanishes in the limit $\mu \to 0$.

Parallel transport gauge wavefunction derivative

In general, this would be a way of computing wavefunction derivative by fixing the gauge[3]. In our case, it will be used just to conveniently initialize the velocities of the electrons since the CP dynamic of the electrons is completely fictitious. In practice it is a good choice: the fictitious kinetic energy of the electrons keeps oscillating around the value that we compute with this derivative for an equilibrated system.

Let

$$P(t) = \sum_v |\psi_v(t)\rangle\langle\psi_v(t)| \tag{3.16}$$

be the projector over the occupied electronic states at time t, $v = 1, \ldots, N_b$ where N_b is the number of bands. Then we can write a generic occupied state as

$$|\phi(t)\rangle = P(t)\,|\phi(t)\rangle \tag{3.17}$$

deriving the expression we have:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\phi(t)\rangle = \frac{\mathrm{d}P(t)}{\mathrm{d}t}|\phi(t)\rangle + P(t)\frac{\mathrm{d}}{\mathrm{d}t}|\phi(t)\rangle \tag{3.18}$$

The parallel transport gauge is the one where we put $P(t)\frac{\mathrm{d}}{\mathrm{d}t}|\phi(t)\rangle = 0$ (the derivative has no components over the occupied manifold), so we define the wavefunction derivative to be

$$\frac{\mathrm{d}}{\mathrm{d}t}|\phi(t)\rangle \equiv \frac{\mathrm{d}P(t)}{\mathrm{d}t}|\phi(t)\rangle \tag{3.19}$$

This has many technical advantages in the numerical computation of the wavefunction derivative. It is not possible to numerically differentiate the wavefunction computed with two independent minimizations of the ground state because in general, the phase choice of the DFT solver is arbitrary. Therefore initializing a CP simulation with a $\dot{\phi}$ computed in this way is not possible, we would get nothing but random numbers. On the contrary, the projector $P(t)$ does not depend on the gauge, so it can be differentiated numerically without any particular issue other than choosing a good time interval for computing the difference. The alternative to using this method can be to initialize the electronic velocities to zero, but this will result in a larger kink to the wavefunction, ending up with more kinetic energy after the system equilibrates.

3.3.3 cp.x code

The work on the `cp.x` code was focused on the code paths used for this study, mainly in the initialization part of the simulation. The biggest work was done on the conjugate gradient minimization routine.

The conjugate gradient routine, located in the `CPV/src/cg_sub.f90` file of the official Quantum ESPRESSO distribution, has the main purpose of initializing the CP simulation, and it is able to compute the wavefunction derivative in the parallel transport gauge. The parallel transport derivative routine is common with the code `QEHeat`, and is in the module `Modules/wave_gauge.f90`. Numerically, $\frac{\mathrm{d}P(t)}{\mathrm{d}t}|\phi(t)\rangle$ becomes

$$\frac{\mathrm{d}P(0)}{\mathrm{d}t}\left|\phi_i^0\right\rangle \sim \frac{1}{\mathrm{d}t}\sum_v \left(|\phi_v^+\rangle\langle\phi_v^+| - |\phi_v^-\rangle\langle\phi_v^-|\right)|\phi_i^0\rangle \ . \tag{3.20}$$

In the code, this is equivalent to the computation of two matrices filled with scalar products of wavefunctions calculated at three different timesteps. If needed, also two timesteps can be used. The gauge of the resulting wavefunction will be the one of $|\phi^0\rangle$.

The wavefunction derivative is computed every time the CG routine is called and dynamics is running. At the moment this is the only way to initialize wavefunction velocities to a reasonable value different from zero.

The possibility to call the routine in the middle of a Verlet run was implemented using `cp.x`'s autopilot module. Following the documentation, now it is possible to run a full electronic minimization, with a wavefunction velocity calculation, on the fly when the code is running by simply writing a file called `pilot.mb` inside the simulation's folder with the following content:

```
PILOT
  NOW : electron_dynamics = 'cg'
  NOW + 1 : electron_dynamics = 'verlet'
ENDRULES
```

Then we ported all the relevant code paths for norm-conserving pseudopotentials of the CG algorithm to the GPU architecture using OpenACC, and we fixed the following small issues:

- wavefunction velocities were not updated correctly when changing the timestep
- initialization of the velocities of the ions from a Boltzmann distribution was not working
- missing tests in the test suite for the CG algorithm
- missing documentation of the output files, small documentation updates

[22, 21, 23]

3.3.4 Car-Parrinello `AiiDA` workflow

The workflow has a big variety of inputs listed in table 3.3. A part is related to the choice of the code and the parameters for the resource manager of the HPC system. Then there are the parameters of the DFT functional: wavefunction cutoff, pseudopotential, and others depending on the calculation one wants. Then the workflow is structured so that the choice of the Car-Parrinello simulation parameters (electronic mass and timestep) does not need to be made explicitly. You provide a reasonable range of timesteps and masses, and an acceptable Car-Parrinello/Hellmann-Feynman force ratio. The code will try to find an efficient choice of the simulation parameter by doing some guesses and by seeing how the simulation behaves. In the inputs, you provide the initial condition of the system and the points in the T-P diagram that you want to simulate. Then the code performs all the steps that would be made by hand and are usually boring. The user is left only with the task of inspecting the products of the simulations. The code performs the following automatically.

Initialization and choice of the parameters

This code does not assume that the initial configuration is equilibrated. For example, the initial state can come from a force field simulation, or an arbitrary randomized set of positions and velocities. So the first thing that the code does is a few steps of Born-Oppenheimer dynamics, so if the initial state is far from equilibrium configuration the simulation can warm up a little bit, allowing faster atomic movements without failing at the very beginning. If we used the CP dynamics in those very early steps, we easily got very high electronic temperatures from the beginning. Those steps are performed with a direct conjugate gradient minimization of the electronic state. At the end of those steps, the wavefunction velocity is computed within the parallel transport gauge.

We start the CP dynamics with a very small timestep, to allow further (preliminary) equilibration, and a very small electronic mass, that should work for every system. Now the task is to find a reasonable value for the fake electronic mass and the timestep to perform a more efficient simulation. The code generates a variety of trajectories with different electronic masses and the same low timestep value, and then, if some of them succeed, starting from the last point of each of them, a range of trajectories with different timestep values, chosen from the input parameters. After all those simulations are finished, we pick the ones that are successfully finished and we run on many snapshots the PW.x code, computing the forces. The last step is to pick all the PW runs that are successful and compute the CP/Hellmann-Feynman force ratio with the corresponding frame of the CP trajectory. We get histograms like the one of figure 3.8. By looking at the statistics of the data, we can decide if the simulation fits the required accuracy, does not fit it, or is simply garbage (that is the case if the distribution spreads too much). If

`cp_code`	node with the CP code to use
`pw_code`	node with the PW code to use
`structure`	initial structure, or trajectory to start from
`ecutwfc`	cutoff for the DFT energy functional
`pseudo_family`	pseudopotentials
`target_force_ratio`	the code will choose the fictitious electronic mass to have a CP/Hellmann-Feynman force ratio close to this parameter
`additional_parameters_cp`	additional parameters to pass to all CP calculations
`emass_list`	list of fictitious mass to try at the beginning of the parameter finding algorithm
`max_slope_min_emass`	minimum acceptable fictitious mass
`dt_start_stop_step`	range of timesteps to try at the beginning of the parameter finding algorithm
`number_of_pw_per_trajectory`	number of frames that will be used to compute the forces that will be compared to the CP ones in the parameter finding
`nve_required_picoseconds`	number of picoseconds of the final NVE trajectory, after all the equilibration
`nstep_initial_cg`	number of Born-Oppenheimer steps to do at the very beginning of the simulation, starting from the initial structure

`thermobarostat_points`	list of dictionaries with the following format:
	`{"temperature_K": 2800,`
	`↪ "pressure_KBar": 1000,`
	`↪ "equilibration_time_ps": 1.0,`
	`↪ "thermostat_time_ps": 1.0}, ...`
	The simulation will loop over those T,P points and try to equilibrate

`temperature_tolerance`	if the difference between the average temperature in the NVE equilibration and the asked temperature is more than that, repeat the procedure for this point
`pressure_tolerance`	if the difference between the average pressure in the NVE equilibration and the asked pressure is more than that, repeat the procedure for this point

Table 3.3: List of the input parameters of the CP workchain

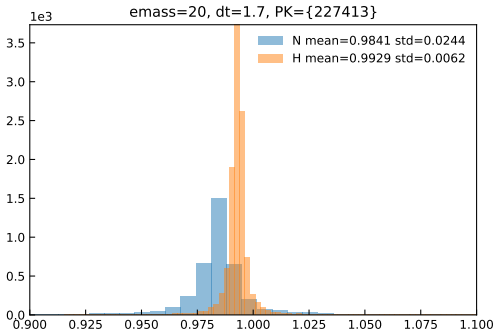


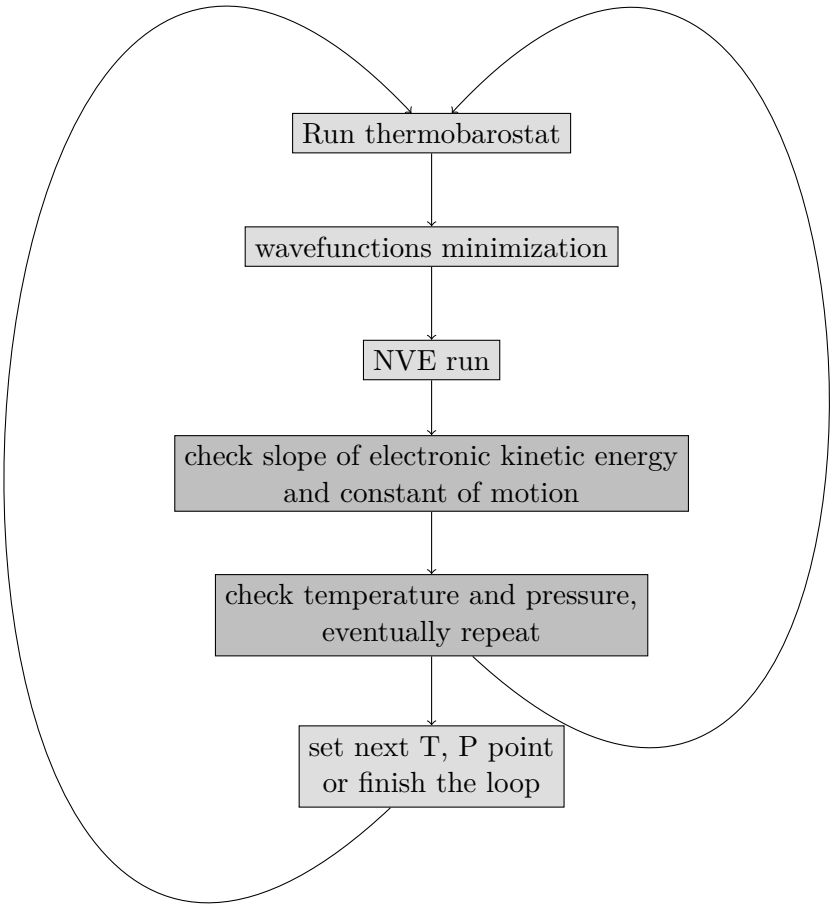
Figure 3.8: Example histogram of the ratio between the Car-Parrinello forces and the forces computed with a standard self-consistent DFT calculation. Atomic units.

no simulation fits the requirement, the electronic fictitious electronic mass is lowered by a factor, then everything is repeated. At some point, this loop terminates, and the code chooses the higher electronic mass between all the simulations that are accurate enough. This is our best guess when the trajectory is a few timesteps long.

Now the code picks the last frame of the chosen trajectory and, if asked to, runs many benchmark trajectories of a few hundred molecular dynamics steps to select the most efficient available parallelization options for the cp.x code. This works by trying a few values of band groups and task groups and seeing how much time is spent on one molecular dynamic step. The ionic mass optionally is adjusted using the average force ratio (thus estimating empirically the correction of eq 3.15, to correct the leading term of the finite fictitious electron mass effect. Now we are ready to start the simulation.

Thermobarostatation of the trajectory

At this point, we enter the workflow loop over the number of P-T points. First, the thermobarostat is run. The code `cp.x` is repeatedly run till the length of the trajectory specified in the input is reached. After this, a CG minimization of the electronic ground state and a wave function velocity computation is performed. Then a small NVE simulation is run, as required in the input. After this, the code checks the last NVE trajectory and computes the slope of the electron's kinetic energy and the constant of motion. The electronic fictitious mass and the timestep are lowered if they are not within the limits. The ratio of the forces between the is computed again and if asked the ionic mass is adjusted again with the new factor. Then a new CG is made, and everything loops over all the P-T points requested in the input. In the end, a longer NVE trajectory is computed optionally.


Run thermobarostat
wavefunctions minimization
NVE run
check slope of electronic kinetic energy and constant of motion
check temperature and pressure, eventually repeat
set next T, P point or finish the loop

3.4 Code for efficient data analysis of huge trajectories

[6] `Analisi` is a multi-platform tool written in C++ and python for computing averages on molecular dynamics trajectory. The tool was written to analyze the biggest trajectory, so many levels of parallelization are implemented. The structure is such that adding new computation is simple, and every calculation added can use very easily the MPI framework to compute block averages without writing a single line of MPI code since each block is independent. The tool comes with a command line interface and a python interface and has an extensive test suite that is continuously run on Linux, Mac, and Windows operative systems. The test suite includes unit tests, tests of the command line interface functionality, tests of the conversion tools between different molecular dynamic trajectory formats, and tests of the python interface inside jupyter notebooks. A binary package on `https://anaconda.org/conda-forge/analisi` is also available on every supported operative system, for x86, arm, and power9 processor architectures. Some of the calculations implemented are:

- Van Hove correlation functions[72] and its particular case, the pair distribution function $g(r)$
- angular density expanded in spherical harmonics correlation function
- Steinhardt (averaged) descriptors[44]
- SANN first neighbor algorithm[73]
- mean square displacement
- multi-component Green-Kubo / Einstein-Helfand time domain integrals
- vibrational spectrum
- atomic density calculation

3.4.1 Structure overview

The high-level structure is shown in figure 3.9. The final user can interact with the tools from a simple command line interface that takes a binary LAMMPS trajectory file for input, or from python using the code as a library. In the latter case, the library can take advantage of the ubiquitous NumPy array format, using them directly as inputs. The python library approach can better integrate the tool, for example, in an `AiiDA` framework or beautiful jupyter notebooks. The command line interface on the other side can be used very easily on supercomputers to deal with the heaviest trajectory analysis or to get a fast plot of the requested calculation with Gnuplot.

Some parts of the code are common to all calculation types like MPI communications for doing block averages, multithreading boilerplate code, and the interaction with the binary trajectory file. These part are interfaced with an abstraction layer that allows changing their implementation completely, keeping the interface fixed, without touching the other parts of the code. In the code, there is heavy use of the template feature of the C++ programming language, and we completely avoid the virtual class approach, where the code decides at runtime which particular implementation of the abstraction to choose.

Each calculation that the tool can do is mapped to a C++ class. This layer is found after the two parallelization levels of figure 3.9. The class has to implement a specific

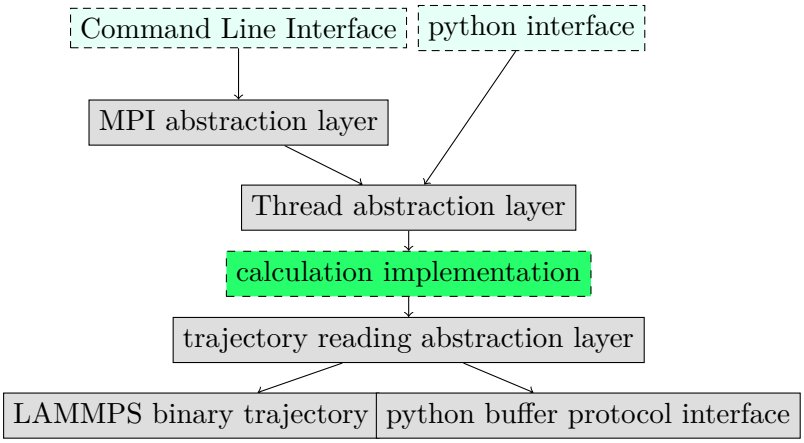


Figure 3.9: High-level organization of the code of the `Analisi` toolbox. To implement a new calculation the advanced user must write the related class and write some boilerplate code in the command line interface and in the python interface to expose the code to the outside world, touching the parts marked with the dashed box in the diagram.

interface so it can be used correctly from the upper layers, specifically from the classes `BlockAverage` and `CalculateMultiThread`. It must implement the following functions:

- `void calculate(int)` the function that is called to compute the quantity in a block, with the timestep index as the only argument
- `int nExtraTimesteps(int)` used to know how much timestep does the class need to access after the end of the block, with the number of the block as argument
- `reset(int)` called before `calculate` with the size of the block as argument
- algebraic operations on the data, that can be borrowed from the class `VectorOp` if the new calculation is derived from there
- optionally, the class can derive from `CalculateMultiThread` and implement the function `calc_single_th(...)` with a range of arguments that depends on the strategy of parallelization. Many strategies are implemented. In this case, `calculate(int)` is implemented by `CalculateMultiThread`.
- to interact with the python interface, the class can implement some function to describe the shape of the data array

The calculation class can use the provided trajectory reading abstraction layer to use, without any additional effort, both the Lammps binary and the python interface. The trajectory reading layer interacts with the higher level block average layer to load the data before each call to `calculate(int)`. The advanced user therefore can use only the following functions to access the trajectory data:

- `positions_data()` returns a pointer to the beginning of the contiguous position array, where the memory is arranged putting side by side the 3d vectors coordinate like x1 y1 z1 x2 y2 z2 ... . Note that the address may change after a new reading of the trajectory if a size change is necessary. Only the trajectory data of the current block is loaded into the memory.

- `get_natoms()` returns the number of atoms
- `get_nloaded_timesteps()` return the number of timesteps loaded into the positions and velocity arrays (equals to the block size plus the extra timesteps wanted by the calculation returned by the function `nExtraTimesteps`)
- `velocity_data()` returns a pointer to the beginning of velocities data, in the same format as the position one.
- `positions(int iatom, int itimestep)` return a pointer to the position of iatom at timestep itimestep. Note that timestep indexes are absolute, 0 is the first timestep of the whole trajectory.
- `velocity(int iatom, int itimestep)` return a pointer to the velocity of iatom at timestep itimestep.
- `d2_minImage(int i, int j, int itimestep, int jtimestep)` minimum image convention of the square distance between specified pair of atoms at the specified timestep. The cell vectors at timestep i are used for applying the PBC.

Many other functions can be used, and they can be directly read from the code in the file `lib/include/basetrajectory.h`.

4

Application to high PT phases of ammonia

In the nearer future, before potential space missions, some progress is envisioned in the following fonts. Further improvements in EOS calculations and experiments of volatile materials such as water, ammonia and methane, their mixtures, as well as their mixtures with rock or with hydrogen (and helium), are essential. Quote from *The interiors of Uranus and Neptune: current understanding and open questions* Ravit Helled and Jonathan J. Fortney [33]

4.1 Superionic Ammonia

Ammonia is a candidate material for the inner layers of planets like Uranus and Neptune, and its properties at high pressure and temperature (hundreds of gigapascals and thousands of kelvins) are yet not fully understood. Understanding the high-pressure phase diagram is fundamental to excluding certain materials or better understanding the planets' possible dynamics.[34, 33]

In this chapter, we investigate the thermal properties of the superionic phase of ammonia. Since its structure is not yet established, it is necessary to understand its most stable atomic configurations before computing transport properties. Our results add some new understanding of its structure but also leave some questions unanswered. Nevertheless, we provide an estimate for the thermal conductivity coefficient.

Ammonia has a very rich high-pressure phase diagram, presenting two solid phases that are well known experimentally, a plastic phase and superionic phases whose structure is less clear, in a region of the PT space approximately between 10-100GPa and 400-3000 K. The density of the material here is around $2g/cm^3$ and is compatible with the estimation of the radial density profile of gas giant planets, estimated from the gravitational field moments.

We are interested in its thermal properties because it is possible to measure experimentally the energy dissipated by the planet by looking at its internal luminosity and the surface temperature. Knowing how much energy is dissipated makes it possible to build a model for the thermal history of the planet[66]. A crucial ingredient for those models is how energy flows from the innermost layers of the planet to the outer surface.

Experimental work on high-pressure materials is available. They are usually performed with a diamond anvil cell (DAC) and high-power lasers. The experimental setup allows a sample size of the order of 0.1mm, introducing strong thermal gradients. Measuring thermal properties in high-temperature experiments can be as hard as impossible.

Experiments performed in [38] point out the possibility, in the 100GPa region, that two superionic phases may be present, according to a detected discontinuity in the sound velocity measurements. One of those two phases may behave like a liquid, possibly generating convective phenomena inside the planetary layer and generating a completely different outcome in the planetary model.

The experimentally well-established phases in this pressure range are two solid phases named IV and V with an HCP-like structure and a plastic phase named III[45, 53]. In those experiments, the α superionic phase can have both HCP-like and FCC-like structures [54, 55].

4.2 Simulations overview

Many series of simulations were done with 108- and 144-molecule systems in HCP- and FCC-like lattices for the nitrogen atoms. The pressure, the temperature, and the density of all of them are shown in figure 4.1. In each series, we used the CP workflow described in section 3.3.4. With this procedure, we explored the system till 3000K and 100GPa, where the decoupling between electronic and ionic degrees of freedom becomes difficult. We used an integration timestep in the range of 3.0-1.5 atomic units depending on the temperature. We took advantage of all the GPU acceleration work that was done on the `cp.x` code. This gives a performance of roughly between 1 and 3 picoseconds of trajectory per day on 2 nodes in the Marconi 100 cluster (8 GPUs in total).

We started with solid phase I in a 108-molecule supercell. After that, we explored the plastic FCC phase III in a cubic supercell with 108 ammonia molecules, which becomes a superionic FCC phase when temperature and pressure are increased. After this, we did a completely new series of simulations starting directly from a superionic FCC or HCP phase at 100GPa with a 144-molecule supercell that was cut along the 111 direction of the conventional FCC cubic cell. With this system, we explored the behavior of strained cells. In the following sections, I will show the behavior of the plastic phase III and the superionic phase, more interesting from a planetary point of view.

4.3 Plastic phase III - FCC

We reached the plastic phase at around 500-600K, 30GPa, starting from a high-pressure phase I configuration. Nitrogen atoms have fixed positions on a conventional FCC 3x3x3 supercell. As it is possible to see from figure 4.2, the hydrogen atoms during the dynamics are rotating around their nitrogen site in a hopping-like way, visiting on average some preferred points on the sphere with radius 1Å. The shape of the molecule remains similar to the original ammonia molecule. This can be seen by visual inspection and is confirmed by the fact that the peak of the H-H plot in figure 4.3 is at 1.62Å, a distance consistent with a tetrahedral geometry where the nitrogen is in the center and the angle between the N-H bonds is 109°. In the ammonia molecule, the angle is 107°. The mean square displacement of the rotating molecule can be roughly estimated with a very simple model where each hydrogen has an equal probability to fill all 4 sites placed at the vertex of a tetrahedron inscribed in a sphere of radius 1Å centered on the nitrogen atom. Inserting

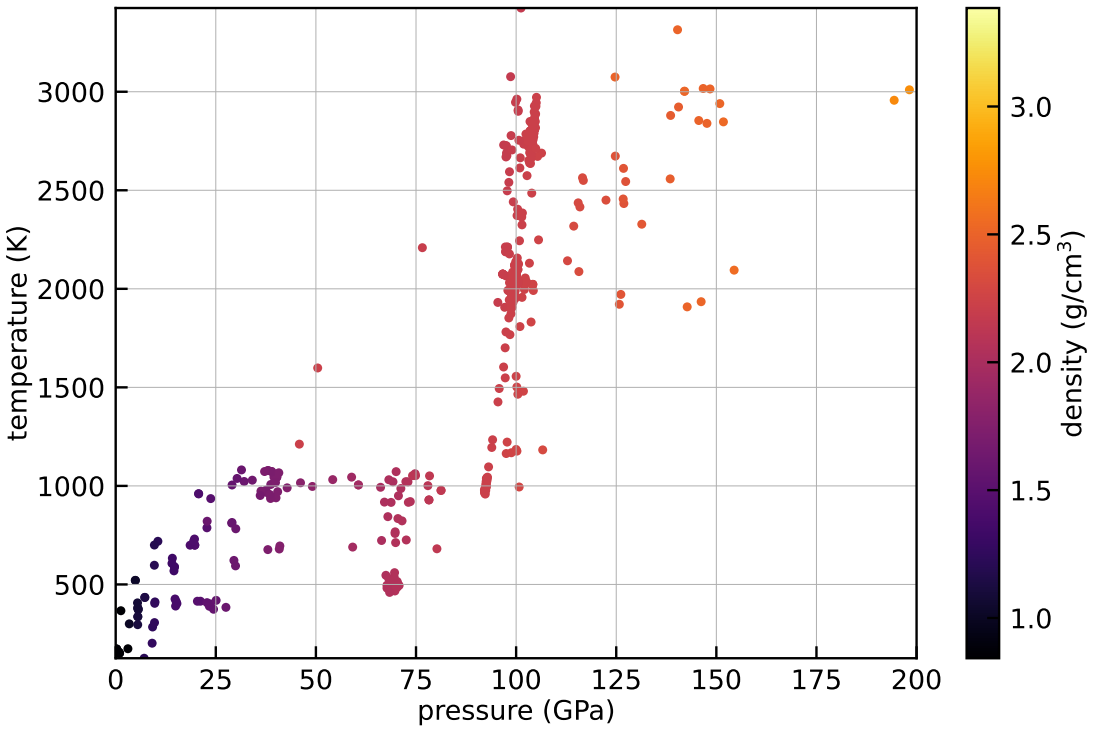


Figure 4.1: Explored PT space for superionic ammonia. The data is extracted from the `AiiDA` database and shows the equation of state.

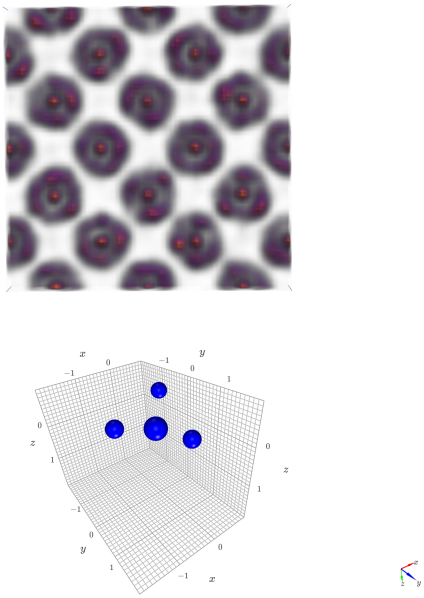


Figure 4.2: Atomic density in the plastic phase at 800K, 30GPa. The molecule has some preferred orientations around the nitrogen sites, but always maintains the structure of an ammonia molecule, and no hopping between sites is present.

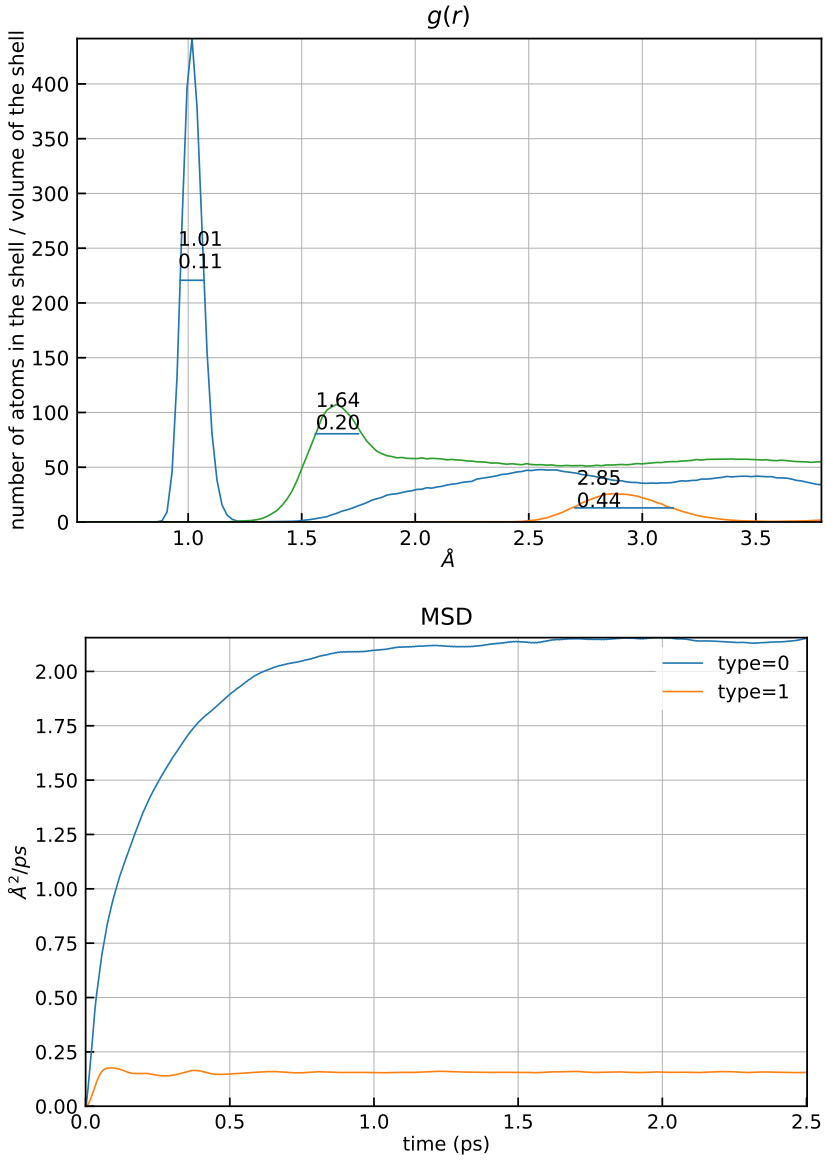


Figure 4.3: $g(r)$ and MSD of the plastic phase. The numbers inside the plot are the position and the width of the peak at the nearby horizontal segment.

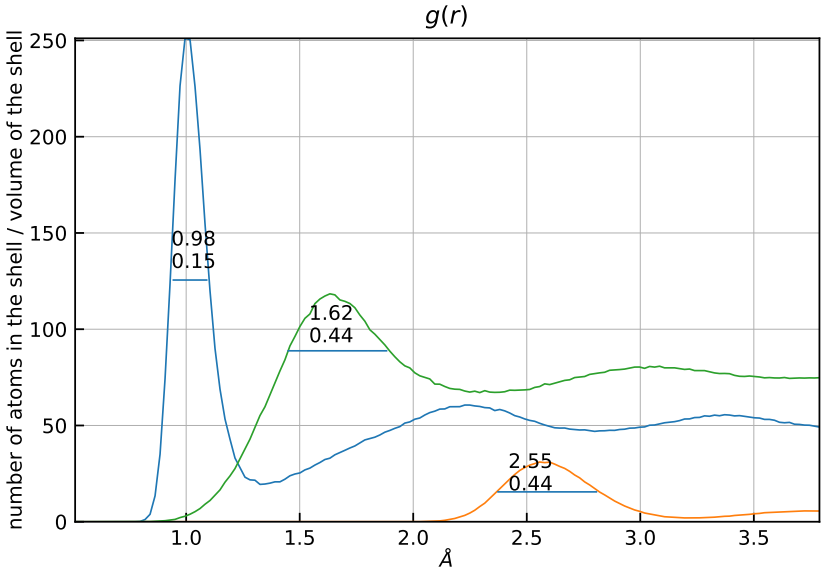


Figure 4.4: Pair correlation function at 2200K, 100GPa.

the H-H distance of the peak and assuming a fixed geometry one gets roughly, for the average square distance, $1.62\text{Å}^2 \cdot 3/4 = 1.97\text{Å}^2$, not too far from the actual value extracted from the trajectory. If one uses a cuboctahedron, with 12 equivalent vertexes, one gets given a radius of 1Å, a sharp 2Å^2 as the limit value for the MSD. The preferred orientation of the molecule is with a hydrogen atom pointing to the nitrogen of a nearby molecule, oriented oppositely.

4.4 Superionic phase

4.4.1 108-molecule system

In the first series of simulations, we arrived at the superionic phase from an FCC-like plastic phase. After the transition, that is at about 750K 70GPa, nitrogen atoms are bound to an FCC lattice, while hydrogen atoms stay most of the time around some preferred sites, roughly with the same tetrahedral structure of the plastic phase, as shown in figure 4.6 but with increased mobility, as we can see also from the $g(r)$ plot of figure 4.4. There are in total 12 favorite sites for the hydrogen, one in the direction of each neighbor nitrogen, showing that free sites of nitrogen neighbors are attractive for H. The sites are located on the vertex of a cuboctahedron, as in figure 4.5, with a similar structure for the HCP case, where a triangular face of the cuboctahedron is rotated of 60°.

There is no possible subset of the vertex of the polyhedron that is able to produce the quasi-tetrahedral structure that the ammonia molecule would like to have, causing the system to be always in an unstable state, being hydrogens attracted both by the molecular tetrahedral sites and the free cuboctahedron sites of the nearest nitrogens. Probably this is also what causes the rotation in the plastic phase too. But here, with bigger thermal fluctuations, the oscillations of the N-H distance cause some H atoms to hop to a neighbor ammonia molecule, which becomes for a short time an NH_4^+ molecule, and then the extra hydrogen is expelled again. All this process happens while the hydrogen atoms rotate randomly around their nitrogen site. A time series of the radial distance of the nearest

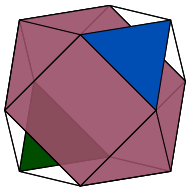


Figure 4.5: Cuboctahedron. It is a polyhedron with 12 vertexes, and each edge has the same length. The distance between the center of the solid and each vertex is equal to the edge length. If we put nitrogen in the center, each vertex is occupied by the nearest neighbor nitrogen. The highlighted planes are the ones taken to build the simulation. This particular example will build an FCC structure. To get the HCP structure is necessary to rotate a triangular face of the cuboctahedron to get an ABA stacking.

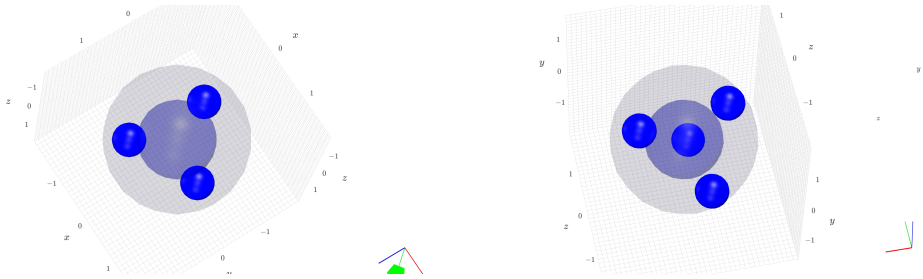


Figure 4.6: Possible configurations of hydrogen atoms in SI phase. The darker sphere has a radius of 0.8Å, while the lighter one has a radius of 1.5ÅIn the first plot the structure is similar to the one of the ammonia molecule, and it is the most stable. In the second particular snapshot, the hydrogen atoms are temporally on the vertex of a tetrahedron. This latter structure is stable on a timescale of 50 femtoseconds at 2800K. The preferred sites are positioned in the direction of the nearest nitrogen atoms on the lattice, at the distance of 1Å

neighbor of an N atom during hopping is shown in figure 4.7.

Cell fluctuations

The FCC simulation cell during the NPT equilibration runs at 100GPa was allowed to tilt and relax in any direction. To better study the behavior of the cell we rotated the cell vectors using a QR matrix decomposition constraining the first cell vector to point in the x direction, and the second vector to lay in the XY plane. In this way, we can read the size of the cell in the diagonal elements of the cell matrix and we can read the tilt factors in the 3 off-diagonal elements. At lower temperatures, till 2000K, the cell vectors oscillate around the high-close-packed equilibrium value, but at 2200K we can see a different behavior. In figure 4.8 some small oscillations around some different equilibrium values appear. Raising the temperature the cell vectors completely change their equilibrium (if any) value, and the cell tilts away from its initial shape, as shown in the time series of the cell parameters of figure 4.9. The situation repeats at higher temperatures, changing on a 5-10 picoseconds timescale the equilibrium value of the cell vectors, till the point at 2800K where on the accessible timescale it is not clear if an equilibrium cell vector exists, as shown in figures 4.10 and 4.11. On the other side, all the simulations where the cell was constrained to be orthorhombic, did not show any unexpected behavior.

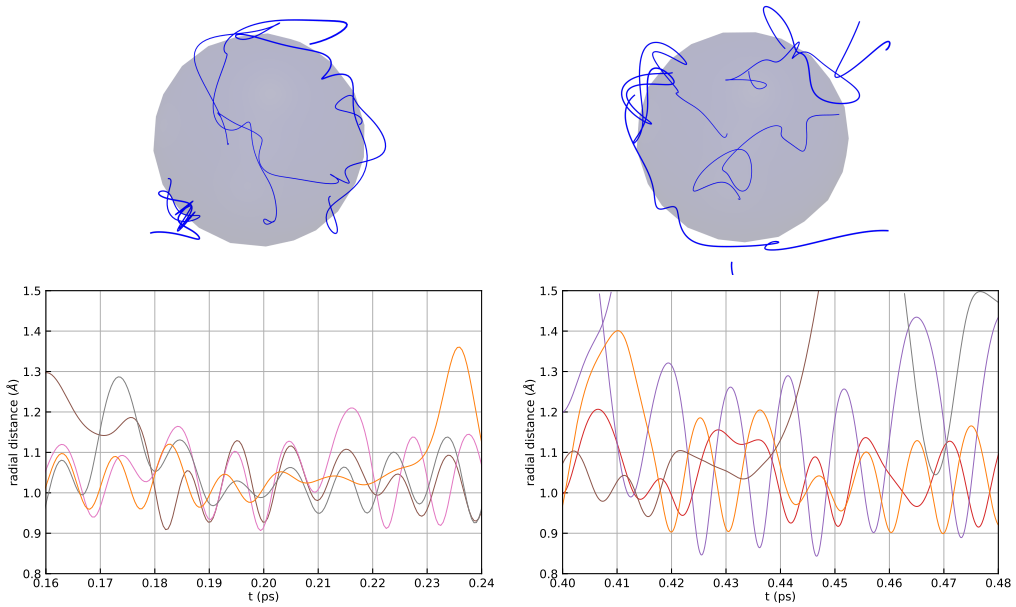


Figure 4.7: Mechanism of diffusion of H atoms in the superionic phase at a temperature of 2000K on the left and 2800K on the right. Example of two small parts of the trajectory 80fs long, where an NH_4^+ ion is formed, for around 50fs. On the top panel, the trajectory of the H on the sphere of radius 1Å is shown. Below is the corresponding radial distance from the central nitrogen. A short period of time after the ion is formed, a hydrogen atom hops away. Most of the time, in other parts of the trajectory, the number of hydrogen atoms around nitrogen is 3.

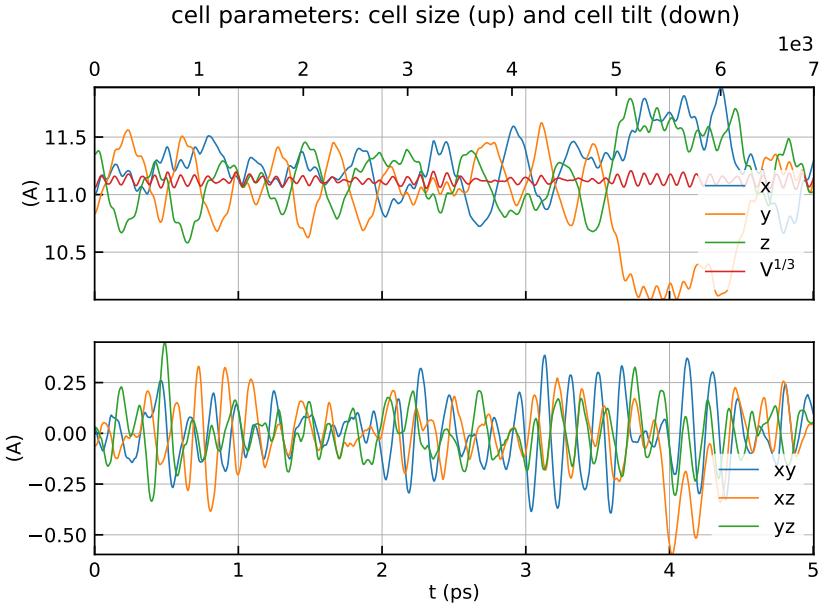


Figure 4.8: 108-molecule system at 2200K, 100GPa. The cell is oscillating around the cubic shape

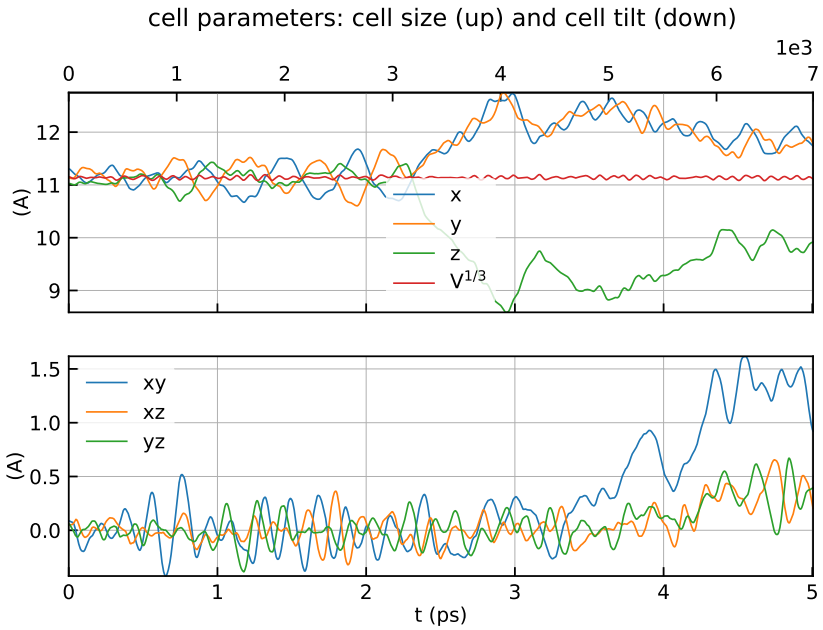


Figure 4.9: 108-molecule system at 2400K, 100GPa. The cell changes its shape at constant volume

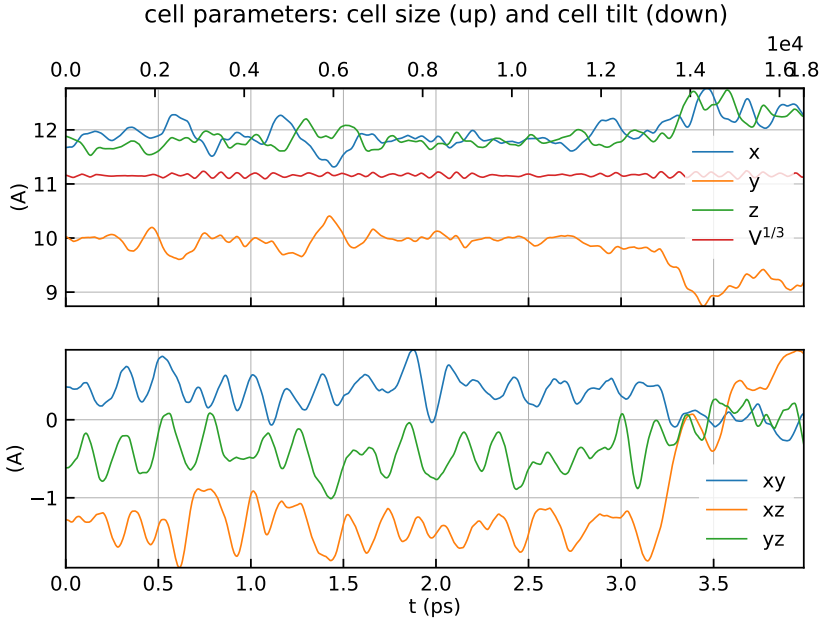


Figure 4.10: 108-molecule system at 2800K, 100GPa. Typical behavior in this regime.

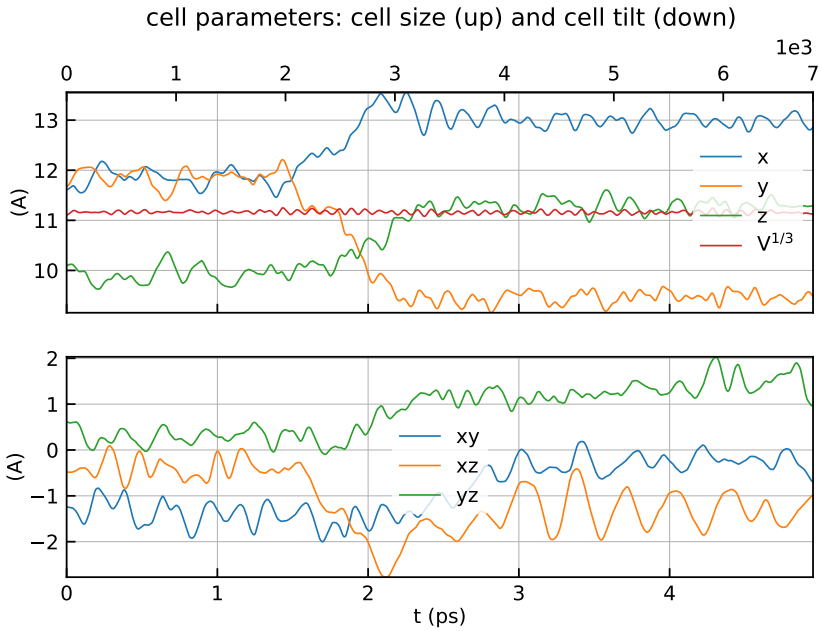


Figure 4.11: 108-molecule system at 2800K 100GPa, a different set of simulations that shows a new change of shape.

Looking at the cell vectors or at the atomic position and trying to understand if there was some repeated pattern in the changing order of the nitrogen sites was not practical with this kind of behavior and this FCC cubic cell cut. This is why now we introduced the use of a local order parameter, that spreads some light on what is happening there.

4.4.2 Steinhardt descriptors

This local parameter is both simple and useful in the description of our system. It is implemented in the open-source C++ library described in section 3.4. It is an atomic quantity defined for the atom i as

$$q_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |q_{lm}(i)|^2} \tag{4.1}$$

$$q_{lm}(i) = \frac{1}{N_b(i)} \sum_{j=1}^{N_b(i)} Y_{lm}(\mathbf{r_{ij}}) \tag{4.2}$$

where $N_b(i)$ is the number of neighbour of atom i and the sum in q_{lm} is over those neighbours. Y_{lm} are the spherical harmonics and $\mathbf{r_{ij}}$ is the vector pointing at atom j starting from atom i. An issue with the Steinhardt descriptors, particularly relevant for our system, is that it is not able to distinguish the local BCC and HCP structures when computing the $(q_6(i), q_4(i))$ histogram. This issue is solved by the averaged order

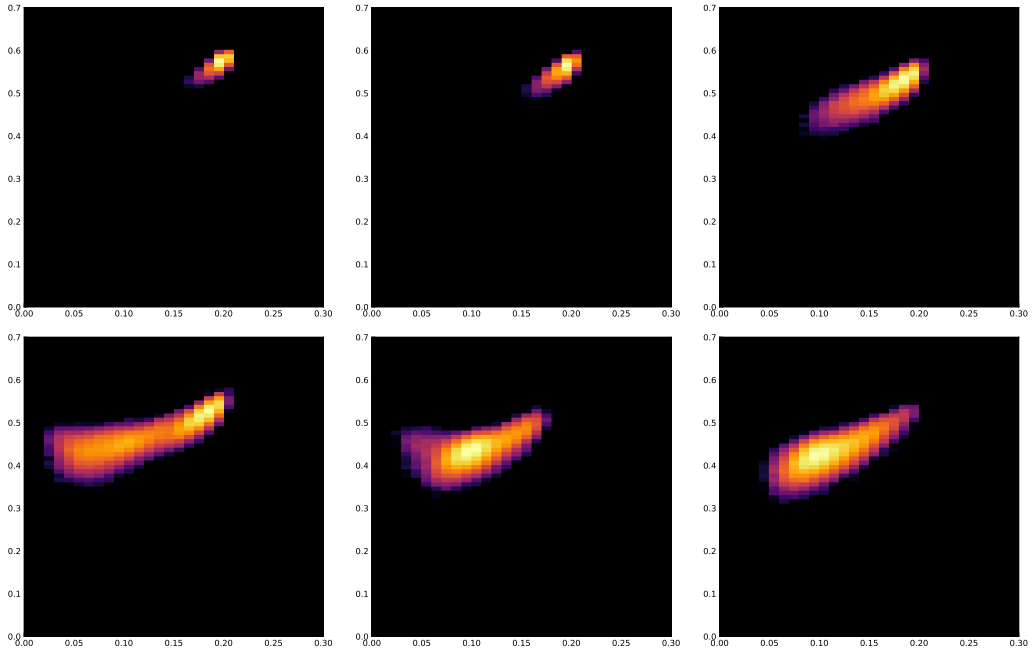


Figure 4.12: $(\bar{q}_4, \bar{q}_6)$ histogram for the superionic system with 108 molecules at the temperature, ordered by row: 1200K, 1500K, 2200K, 2400K, 2800K, 3000K. Each plot shows the quantity only for N-N pairs.

parameter[44], a slightly different version that is defined as:

$$\bar{q}_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |\bar{q}_{lm}(i)|^2} \tag{4.3}$$

$$\bar{q}_{lm}(i) = \frac{1}{N_b(i)} \sum_{k=0}^{N_b(i)} q_{lm}(\bar{k}(k)) \tag{4.4}$$

$$\tag{4.5}$$

where the last sum is performed on all the first neighbors plus the atom i itself, and $\bar{k}(k)$ is a mapping from the neighbor index k to the atomic index $\bar{k}$. To find the nearest neighbors we used the SANN[73] algorithm. In figure 4.12 you can find the histogram produced by computing for all atoms the 2-dimensional quantity $(\bar{q}_6(i), \bar{q}_4(i))$ for the 108-molecule simulations. We can see in this plot that by raising the temperature two small peaks in the N-N histogram changes intensity. The higher one can be produced by atoms arranged in an HCP lattice, while the lower one is produced by an FCC lattice, as shown in figure 2 (b) of ref. [44]. In some frames, the BCC lattice peak appears. This behavior suggests that the nitrogen lattice subsystem tries to move away from a pure FCC lattice, mixing FCC, HCP, and maybe some BCC configurations. This was not reported in any known work up to now.

4.4.3 144-molecule system

Both FCC and HCP are close packing structures. If we cut the FCC conventional cell along 111 planes, we get planes of atoms arranged on an equilateral triangle lattice and stacked with an ABCABC scheme. On the other side, the HCP structure is made of

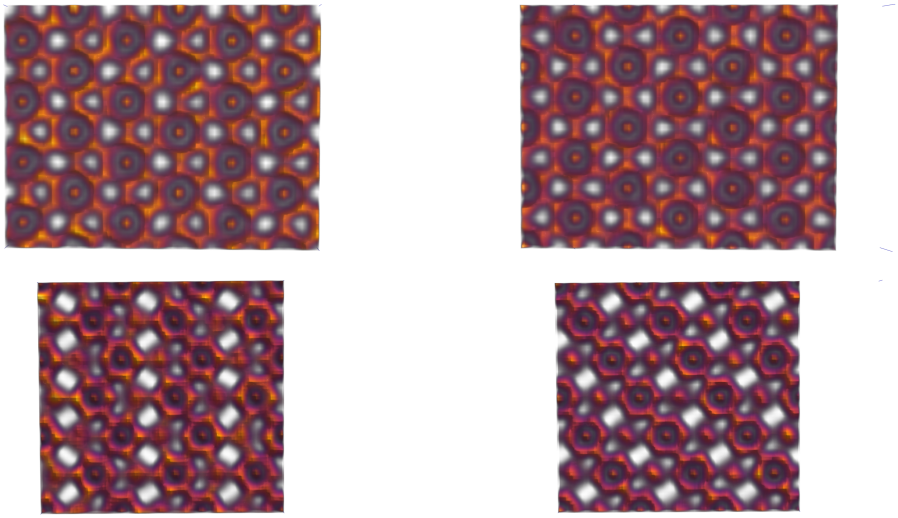


Figure 4.13: Average atomic density at 2800K, 104GPa. On the left, we show the HPC structure, and on the right the FCC one. Only a section of the 144 molecules cell is shown. Top: plane section. Bottom: side section where it is possible to see the ABABAB and ABCABC packing. The N atoms produce the density points on the HCP/FCC lattice only.

planes of equilateral triangles stacked with an ABABAB scheme. It is possible to build a supercell with 6 planes capable of easily transforming in both HCP and FCC structures with a cell deformation. The properties of the FCC and HCP structures are almost equivalent, see for example the vibrational spectrum of figure 4.14, but some measurable differences exist.

We built a new simulation cell with 144 ammonia molecules arranged in 6 planes. We arbitrarily choose the orientation of the ammonia molecules in the initial configuration, as shown in figure 4.15, and we initialized the velocities randomly from a Maxwell-Boltzmann distribution. A first NPT run of an FCC system at 100GPa, 2800K showed at the beginning a movement of the cell vector of exactly one atomic site on the upper plane, making a step toward the HCP phase. On the other hand, at the same thermodynamical conditions, an initially HCP stacked system showed displacement of the planes, making some local FCC structure, after a few picoseconds. The fixed cell NVT simulations did not show any movement of planes in this configuration. In figure 4.18 it is possible to see that the evolution of tilt factors can be fully explained with a plane shift mechanism, and the shrinking of the cell is due to a temporary transition to a BCC lattice.

The constant volume simulations are stable. The density plot of the preferred sites of the hydrogen for two 20ps long NVE simulations at 2800K, calculated by performing the histogram of all the atomic positions is shown in figure 4.13. By looking carefully at the figure, the in-plane structure of the FCC system, which one expects exactly identical to the FCC one, shows some differences in the H density. The shape of the H voids is different for FCC and HCP planes, where the latter case has a lower symmetry.

The cell tilt mechanism that we observe in the NPT simulations is the most efficient that this system can choose to make their planes slide, but it cannot be a physical one. The energy needed to translate a plane is infinite in the thermodynamic limit, thus forbidden in any real system. In our small system, the energy needed to form the plane defects is

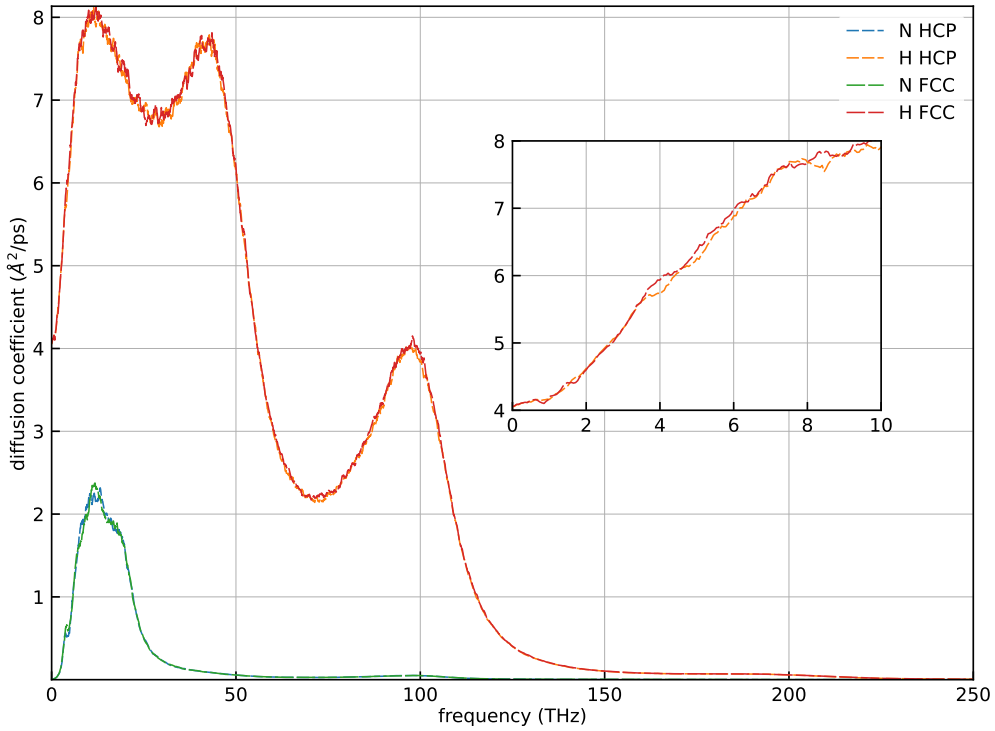


Figure 4.14: Vibrational spectrum of superionic ammonia at 2800K, 104GPa, in units of the diffusion coefficient. FCC and HCP structures are plotted, and they are exactly the same within statistical error. In the inset, it is possible to inspect the plot near the zero frequency, showing that the self-diffusion coefficient for the H atoms is exactly the same for the two structures. Nitrogen atoms do not show diffusion.

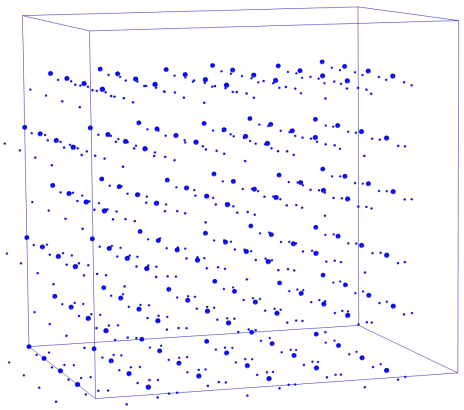


Figure 4.15: Initial structure for the SI simulation at 2000K. This is an arbitrary arrangement of ammonia molecules on an HCP lattice. The rotation of the molecules is chosen so hydrogen atoms are not too near each other.

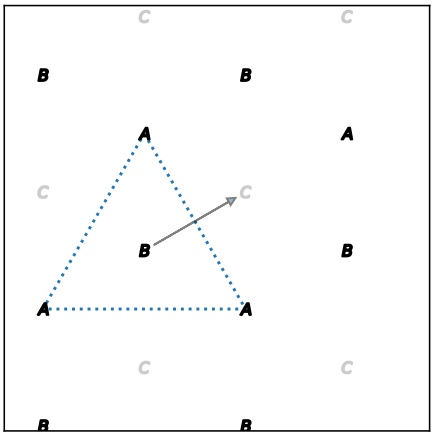


Figure 4.16: The dotted line indicates the location of the energy barrier that the system need to cross to transform a plane B into a plane C. One of the 3 possible directions of the plane translation is indicated by the arrow. When a tilt is applied to the cell, it happens, for example, that the B plane is forced to stay nearer to the energy barrier because of geometrical constraints.

pretty high, requiring the crossing of very high free-energy hills. To visualize better the energy barrier see figure 4.16, where we can see its geometrical origin. If we tilt the system in any of the 3 directions perpendicular to the 3 directions of the in-plane N lattice, we push some plane to move to the next free site on the triangular lattice. To say if the system is solid-like or liquid-like is important to understand how the system reacts under such applied strain.

4.4.4 Shear modulus

We evaluated the shear modulus

$$\mu = \frac{\sigma_{yz}}{\epsilon_{yz}} \tag{4.6}$$

where σ is the stress tensor and ϵ is the strain tensor. We used a deformation of 0.22Å in the y direction of the top face on the highest z coordinate. The cell is 12.84Å tall in the z direction, giving a $\epsilon_{yz} = 0.017$. At a fixed cell, that has a volume of 1831Å^3, we performed many NVT simulations from 1000K to 2700K, obtaining the results of figure 4.19. The result of roughly 50GPa shows that the system that we simulated behaves like a solid.

We used those lightly deformed simulation cells to estimate also the specific heat, that is 6310J/(kg K) at a density of 2219 kg/m^3 at temperatures between 1000 and 2700 K. The internal energy of the system as a function of the temperature in the same series of simulations is shown in figure 4.20.

4.5 A plane sliding path between HCP and FCC SI phases

To investigate all possible plane configurations, we defined a continuous one-parameter function able to construct both the FCC and HCP lattices. The transformation is based

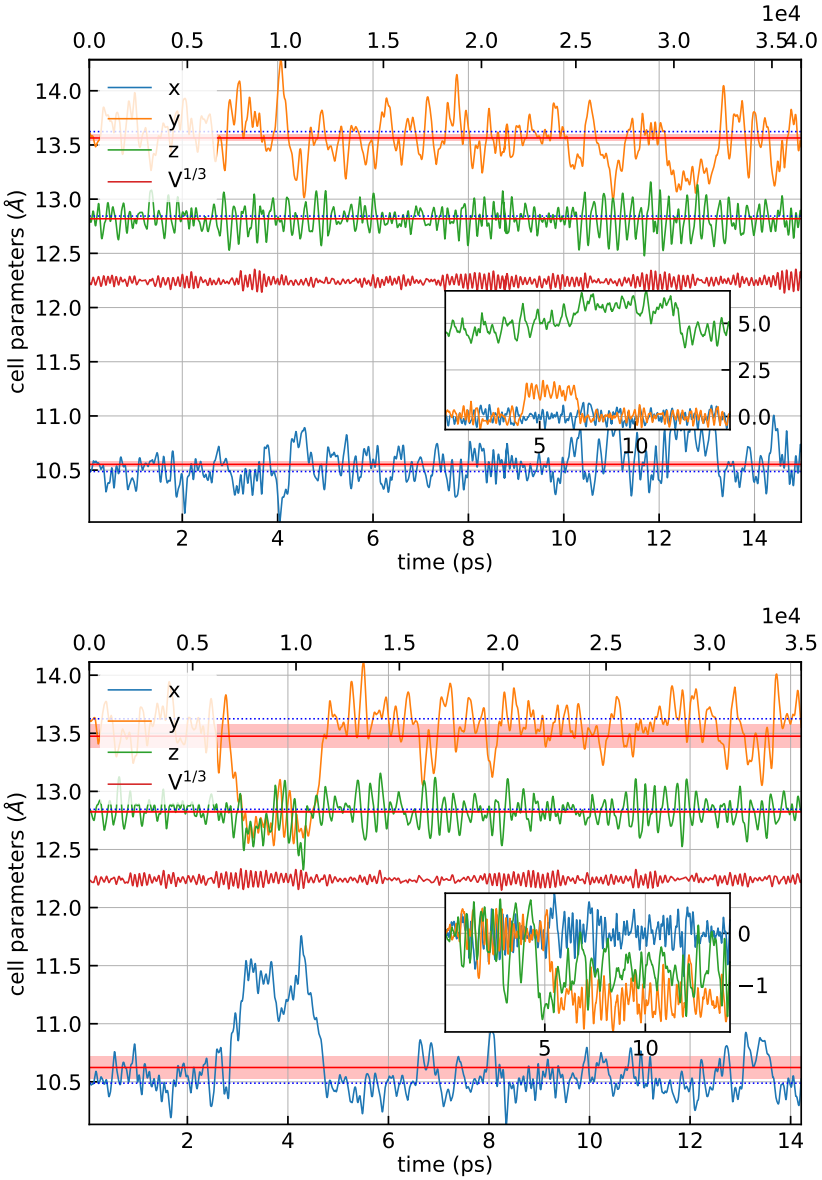


Figure 4.17: Cell vectors time series in the long NPT simulation. Tilt factors are plotted in the inset. The solid horizontal lines are the average values with their statistical error, calculated with a block analysis. The dotted horizontal lines are the value of a cell with exactly a high-close-packed structure with the same volume. In the first plot the difference is more than the statistical error, but of the order of 0.05Å

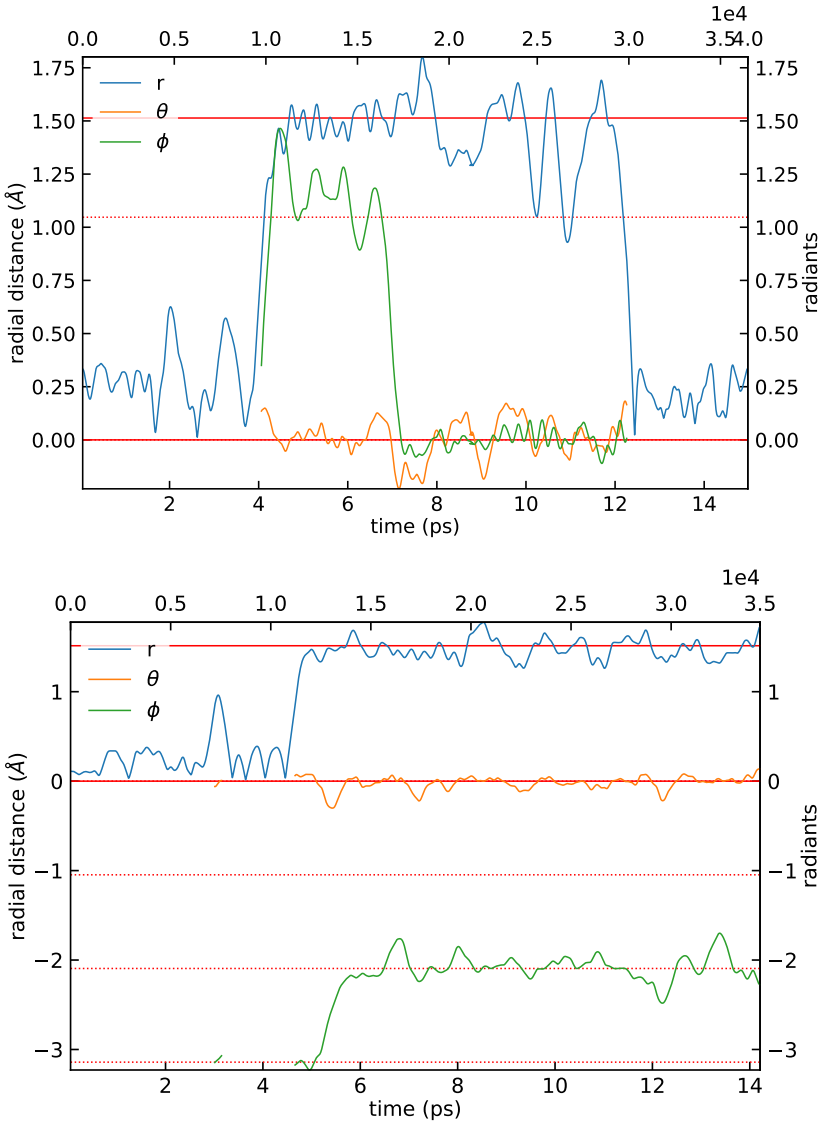


Figure 4.18: Tilt factors time series in spherical coordinates for the long NPT simulation. This plot shows that the movements of the tilt factors are consistent with a plane sliding mechanism as described in the text. A sliding average with a window of size 1000 timesteps is applied to smooth the time series. For low values of the radial distance, the angles are not plotted, since the coordinates system becomes nearly singular. The dotted horizontal lines are the angles allowed by the plane tilt mechanism on the xy plane, which corresponds to angles of the (xz, yz) tilt factors. The solid horizontal lines are the displacement allowed by the plane tilt mechanism. The xy tilt factor always oscillates around zero, and we choose the coordinate system of those plots such that when the θ angle is zero the tilt factors stay on the (xz, yz) plane.

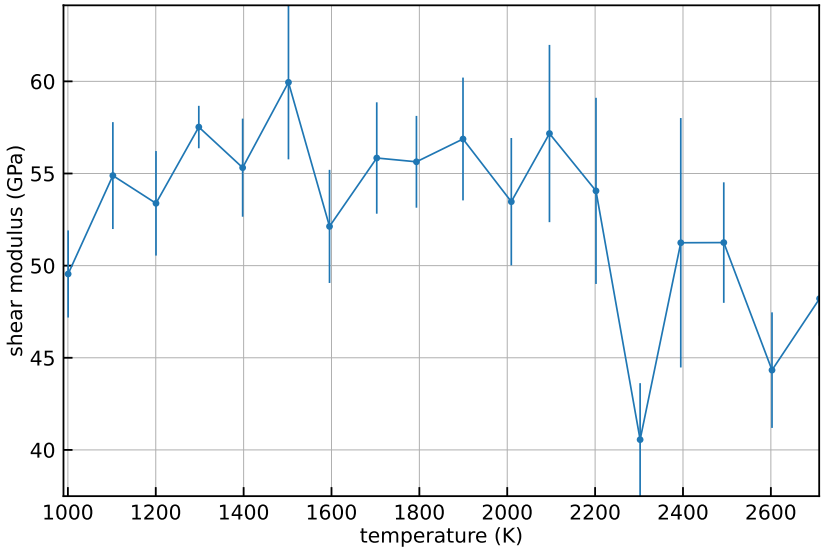


Figure 4.19: Shear modulus μ at fixed volume. It is almost constant but significantly different from zero meaning that the system, without taking into account plastic deformations that occurs on a planetary scale, behaves like a solid.

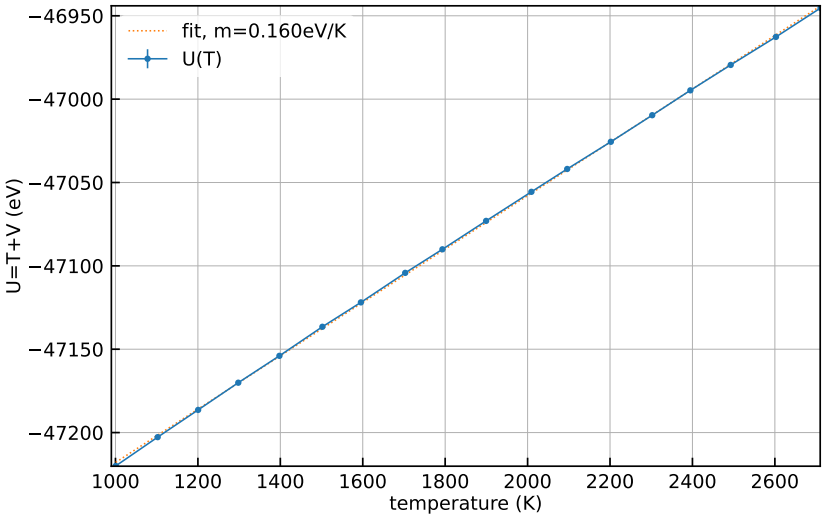


Figure 4.20: Internal energy as a function of the temperature for the simulations series used for the computation of the shear modulus. There was a deformation of 1.7%. The corresponding specific heat at constant volume is $c_v = 6310$ J/(kg K) at the simulation density of $\rho = 2219$ kg/m^3.

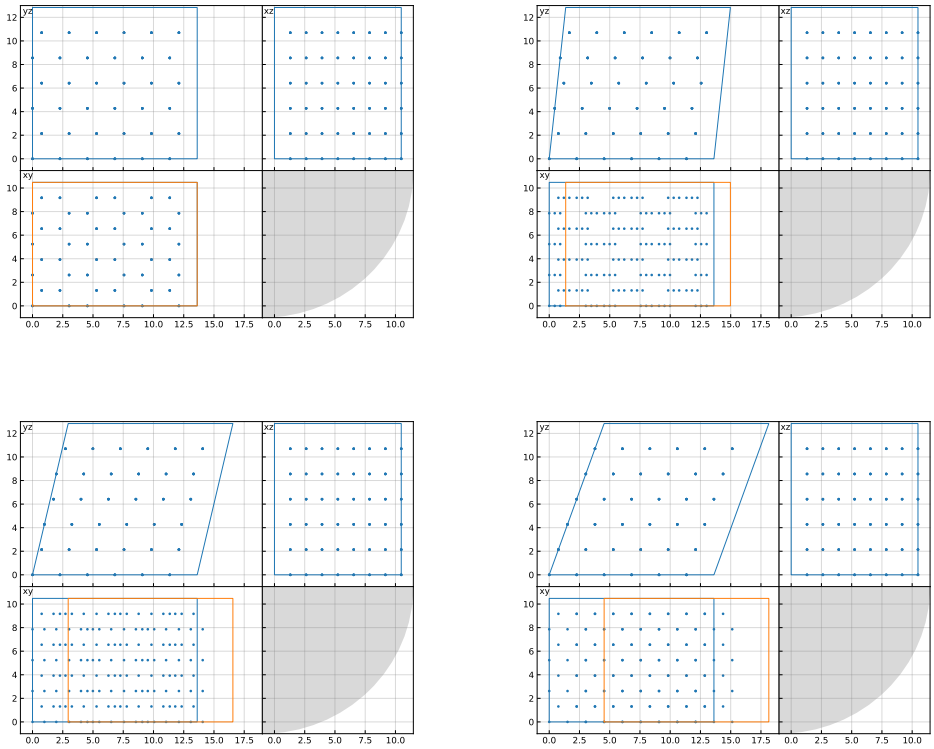


Figure 4.21: Thermodynamic integration path. In the four panels you can find figures of the orthogonal projections of the initial state of each simulation on the yz, xy, xy planes for the simulations with index idx = 0, 6, 13 and 20, with $\lambda = \text{idx}/20$. Simulation with index 0 ($\lambda = 0$) is an HCP stack, while index 20 ($\lambda = 1$) is FCC. The path is obtained by translating the top face of the simulation box along the y-axis, and at the same time translating pair of planes as shown. During the dynamics, atomic planes move and arrange themselves in their preferred configuration. The energy barrier for plane shifts is very high for simulations near $\lambda = 0$ and $\lambda = 1$, while it is low for simulations near $\lambda = 0.5$

on a mixture of cell strain and plane shift. It is shown in figure 4.21. By running a simulation for each initial configuration, it is also possible in principle to compute a free energy difference by performing a thermodynamic integration and possibly point out the most stable configuration:

$$\Delta A = \int_0^1 \left\langle \frac{dU(\lambda)}{d\lambda} \right\rangle_\lambda d\lambda \tag{4.7}$$

where λ is the parameter that we use to move from one state to the other and U is the potential energy of the system. The state for $\lambda = 0$ is an orthorhombic system of 6 HCP stacked (ABABAB) planes, each one containing 6 rows of 4 ammonia molecules with a total of 144 molecules. The state with $\lambda = 1$ is a system with 6 FCC stacked (ABCABC) planes, with the cell tilted in the zy component by $2a\sqrt{3}/2$ where a is the first neighbor distance of the nitrogen atoms. Those numbers are better understood by drawing triangles on figure 4.21, the last panel. The in-plane structure does not change. Doing a simple rescaling of all the positions according to the cell tilt was not the best option to get a

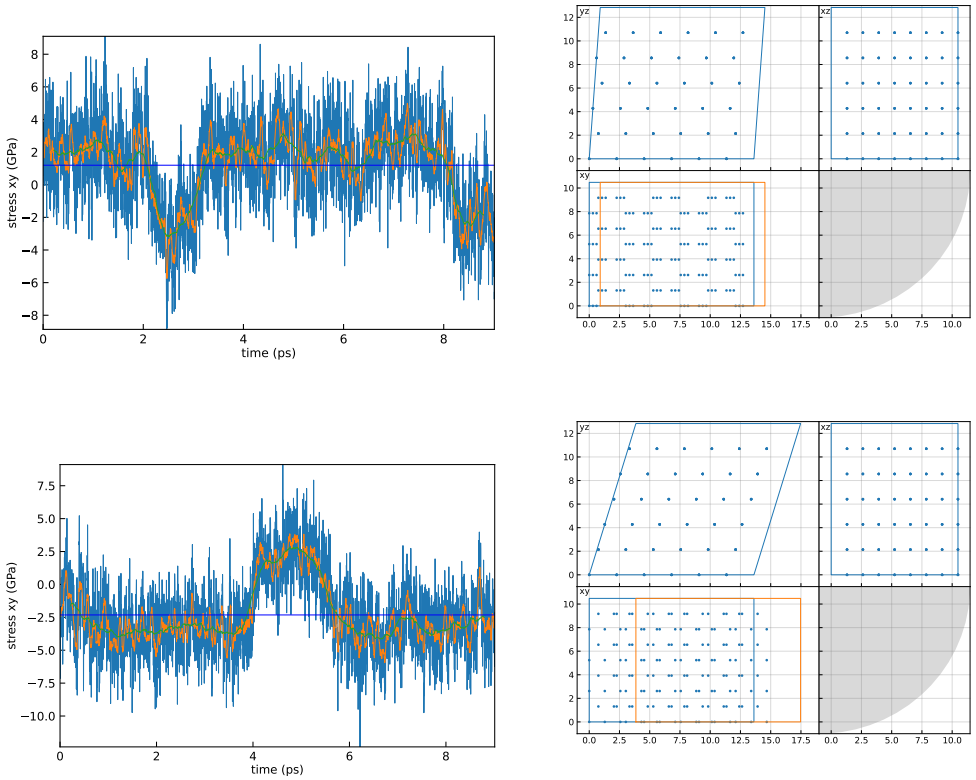


Figure 4.22: Initial condition and yz stress time series in the simulations where it is difficult to calculate averages: randomly there are plane shifts that change the sign of the stress on timescales of few ps. The simulation time accessible with this method is of the order of tens of picoseconds.

stable enough initial configuration. So the following scheme of translating the atomic planes was chosen. First, we divide the planes into three pairs A, B, and C. Then the first group A is kept fixed for all λ. The planes in B are rigidly translated in the y direction with $y' = y + \lambda \cdot 2/3 \cdot a\sqrt{3}/2$. The planes in C are translated with $y' = y + \lambda \cdot 4/3 \cdot a\sqrt{3}/2$. Pairs of translating planes were observed also in the NPT simulation series.

We can have ergodicity problems in the simulations, as shown in figure 4.22. In each simulation with a given strain can happen that for the given temperature, two different planes configuration are accessible by the simulation. However, the transition shows up rarely within the accessible computational time because the energy barrier can still be pretty high. This complicates the calculation of averages (without using metadynamics) and suggests that the energy differences may not be trusted because the ensemble averages on which they are based have insufficient statistics. The energy values of this thermodynamic integration should be taken *cum grano salis*. First, because the plane shift mechanism is not physical (so we can guess only which configuration is most stable, but not by how much energy), and second because the calculated averages may be wrong. But it provides additional useful insight into the dynamics of the system.

In figure 4.23 you can see the results for three temperatures at constant volume. The evolution of the Steinhardt histogram with the strain is shown in figure 4.24, showing the expected peaks when different plane configurations are present.

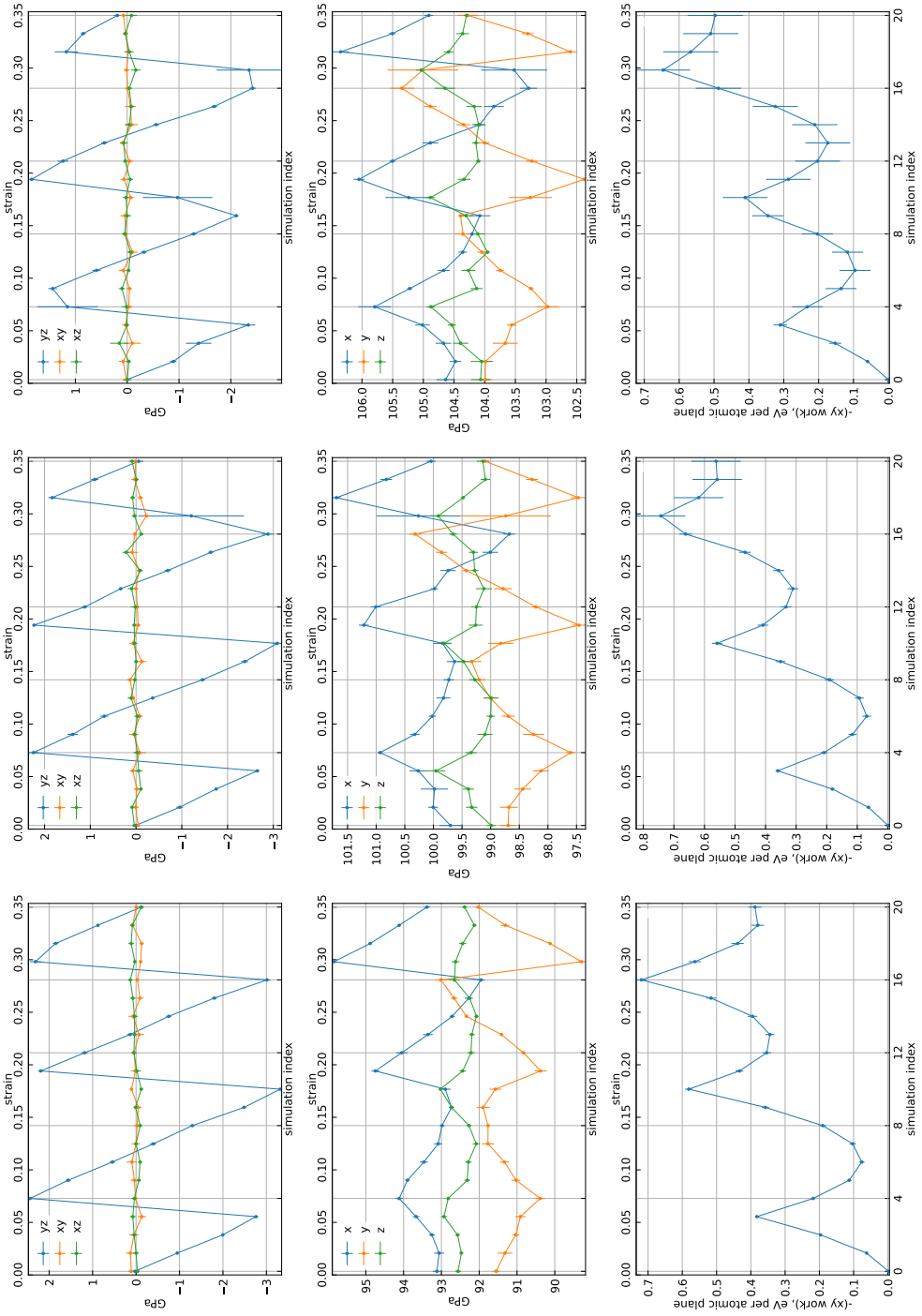


Figure 4.23: First and second rows: components of the stress tensor. In the last row: the result of the thermodynamic integration. Note that the absolute value of the energy, as explained in the text, does not have any physical meaning. Each column is a different T: 1000K, 2000K, 2800K. The volume of all 63 simulations is the same. k_bT is respectively: 0.09 eV, 0.17 eV, 0.24 eV. Simulation index 0 is ABABAB, and simulation index 20 is ABCABC. Cell size is fixed to the high-close-packed value.

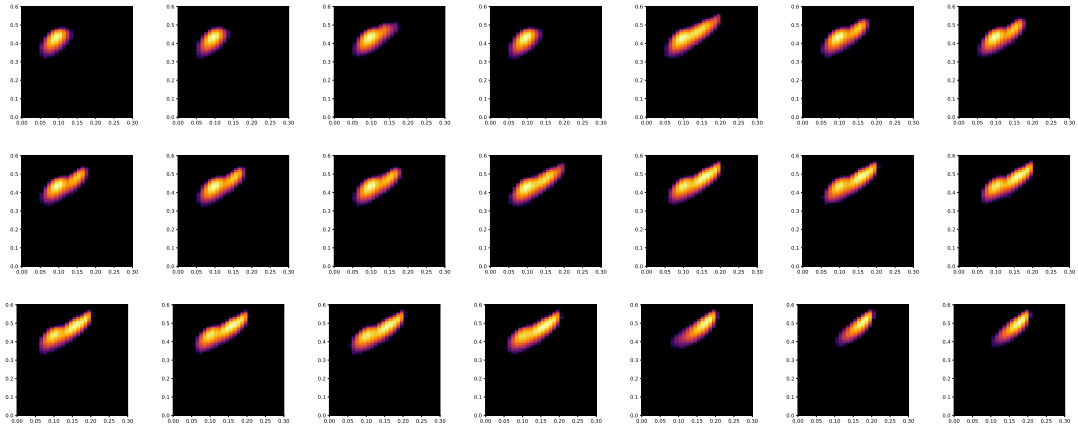


Figure 4.24: Histograms of the quantity $(\bar{q}_4, \bar{q}_6)$ for each simulation from the thermodynamic integration path at 2800K, 104GPa. The first simulation is HCP, the last one is FCC. Note the movements of the peaks with the strain.

At this point of our work, with the experience maturated with those simulations, it is possible to speculate that may exist a transition temperature, much lower than 2800K, and probably around 2000K, where the close-packed structures are not stable, and the real system becomes disordered in the alternation of ABC planes, with local domains with many different configurations.

There is a work in progress to scale up the simulation size using neural network potentials, to be able to detect in a much bigger system the birth of defects and possibly run long metadynamics simulations.

An interesting feature of those simulations, seen in the path between HCP and FCC phases, is the fact that pure HCP and FCC phases show a small anisotropy in the stress, meaning that those structures are not stable in that fixed cell configuration. The anisotropy is not noticeable by simply looking at the stress tensor. It is much smaller than the thermal fluctuations. This small anisotropy, expected experimentally, can also be seen in our NPT simulations of figure 4.17 and 4.18. Here the deviation from the cell vectors of a pure high-close-packed structure is of the order of 0.05Å for the x and y direction, and 0.01Å for the z one.

4.6 Thermal conductivity

To perform the thermal conductivity calculation we choose to use the 144 molecules system with the pure HCP structure. The production run in the NVE ensemble was 20ps long. We expect to get a very similar result for the FCC structure. For example, the two vibrational spectra of figure 4.14 are indistinguishable. Preliminary investigation with the neural network potential also showed little dependence on temperature. The adiabaticity of the CP simulation is acceptable. The time series of the instantaneous temperature is plotted in figure 4.25. A small decrease in temperature is due to a flow of energy from the ionic system to the electronic one, which heated up to the equivalent of 50K of the ionic system. With respect to the temperature of the system, we think this is an acceptable amount since our biggest error is the statistical uncertainty of the final result.

The energy current flux was calculated with the `QEHeat` code. Since the numerical precision depends on the convergence parameter `conv_thr` and the numerical differentia-

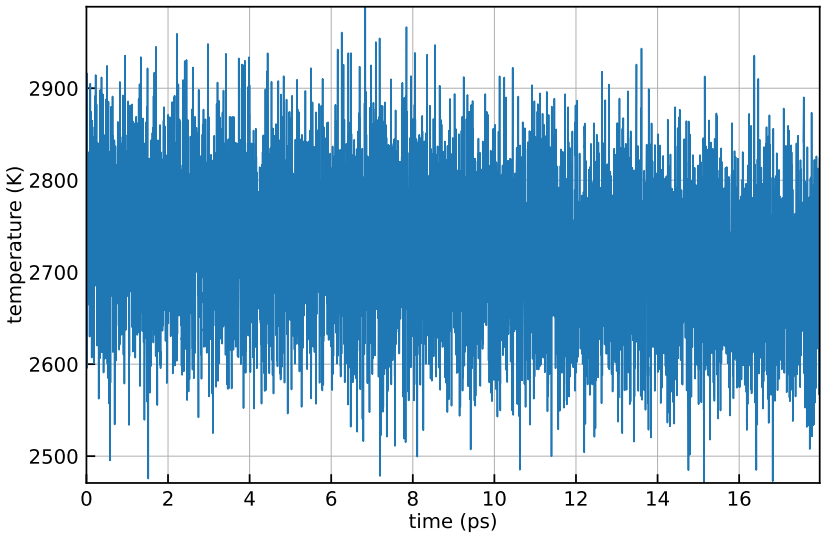


Figure 4.25: Instantaneous temperature time series for the 144 molecule HCP simulation at 104GPa. Some energy is transferred to the electronic system, causing a very low negative slope that is possible to see in the plot.

tion timestep `delta_t`, we used the feature of `QEHeat` that allows us to easily repeat the calculation for a single timestep with a different random initialization of the self-consistent cycle, as explained in 3.1.4. We end up using a differentiation timestep of 0.5 and a convergence threshold of 1e-11, both in atomic units. We evaluated the energy current once every 0.658 fs, obtaining a time series like the one of figure 4.26. This sampling timestep allowed the power spectrum of the energy current to safely go to zero before the Nyquist frequency is reached, as shown in figure 4.27.

To perform the cepstral analysis we used the multicomponent technique with the energy current, the mass current of the hydrogen (or equivalently the nitrogen one), and the electronic current. The last one is particularly important only from a data analysis point of view, since it removes from the energy flux very big contributions at finite frequency, without changing the value at $\omega = 0$ but enhancing the performance of the

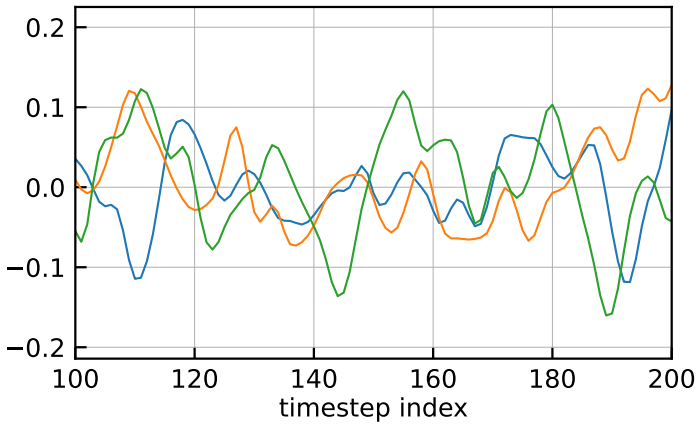


Figure 4.26: Energy current time series of the HCP SI ammonia system

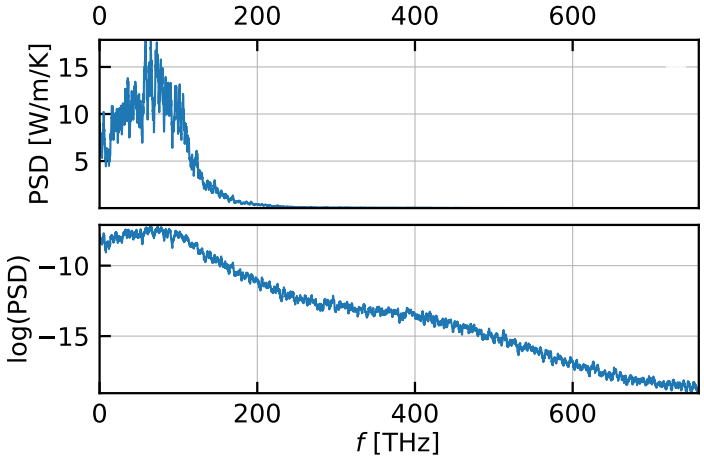


Figure 4.27: Reduced spectrum of the HCP SI ammonia system

cepstral filter as it is possible to see in figure 4.29. The calculation of the electronic current comes with a little overhead from the QEHeat code, in the same run that calculated the energy current. We used a resampling frequency of 100THz, and only 3 cepstral coefficients since the reduced spectrum was very smooth thanks to our analysis techniques. The use of the 3 component analysis allows us to obtain a small statistical error. The result is 6.5 ± 0.5 W/(m K). The convergence of κ is shown in figure 4.28.

4.7 Convergence tests and computational details

We checked the convergence of the energy, forces, and stress at the beginning of the simulations. We confirmed that the choice was correct by performing the same convergence test again with the superionic system at 2800K. In figure 4.30 it is possible to see that the diagonal elements of the stress are indeed well converged at a wavefunction cutoff of 90Ry. The off-diagonal elements converge faster, as expected.

We also checked that the band gap of the same simulation was big enough. As shown in figure 4.31, the band gap never closes and has an average value of 1.89eV, always much higher than k_bT that is about 0.24eV.

We used for the production CP simulations a fictitious electronic mass of 20 a.u and a `emass_cutoff` of 2.5 Ry. The integration timestep for the production run was of 1.7 atomic units (0.041fs), which ensures a well enough conservation of the CP constant of motion. The 144-molecule system had 1152 electrons. We used pseudopotential from PseudoDojo[74] generated with the ONCVPSP code[30]. All the computations but the energy flux one were performed on the MARCONI100 Cluster, equipped with 980 compute nodes with two 16-core IBM POWER9 AC922 at 3.1GHz, 4 NVIDIA Volta V100 GPUs 16GB, and 256GB of system RAM. For the CP verlet integration, we used four nodes and the average time per molecular dynamics step was around 2.0s. A histogram of the performance of each run is shown in figure 4.32.

The computation of the energy flux was done on the GALILEO100 Cluster, equipped with 554 compute nodes with two 24-core Intel CascadeLake 8260 at 2.4GHz and 384GB of system RAM, because the code was not yet fully ported to the GPU architecture. The wall clock time needed to evaluate the energy current for one timestep was, on average,

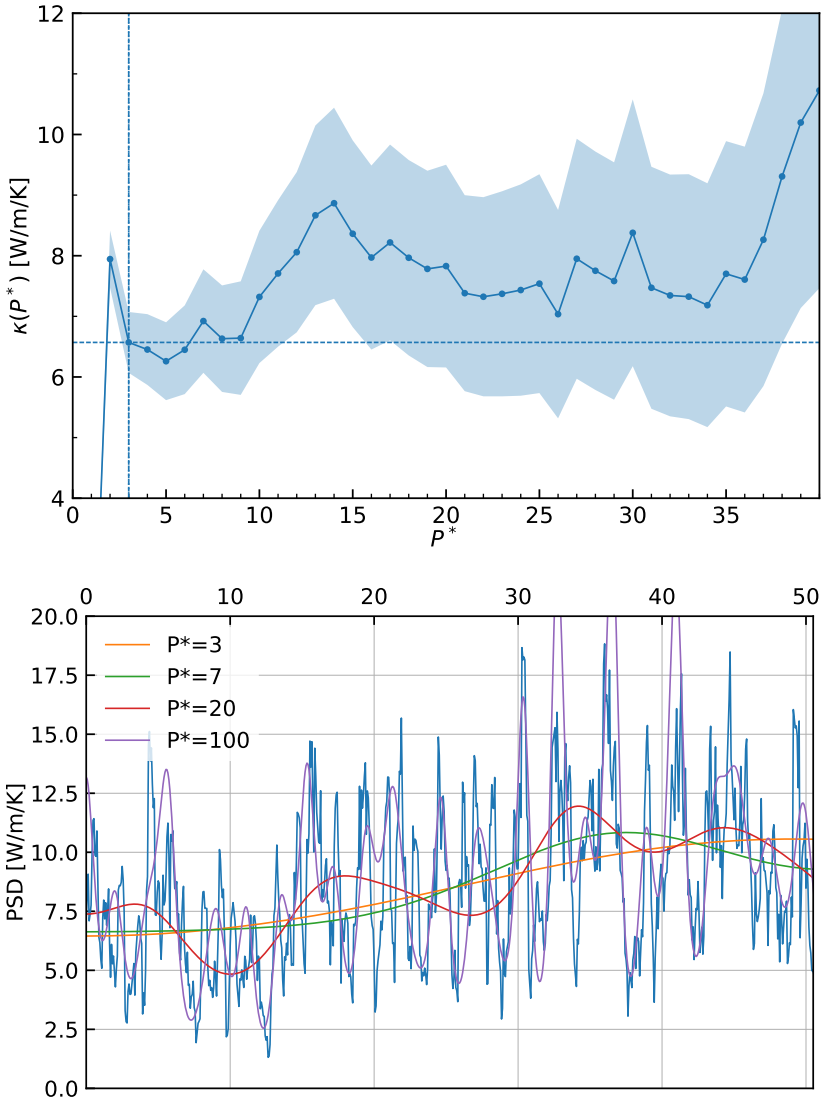


Figure 4.28: Convergence of κ for the HCP SI ammonia system.

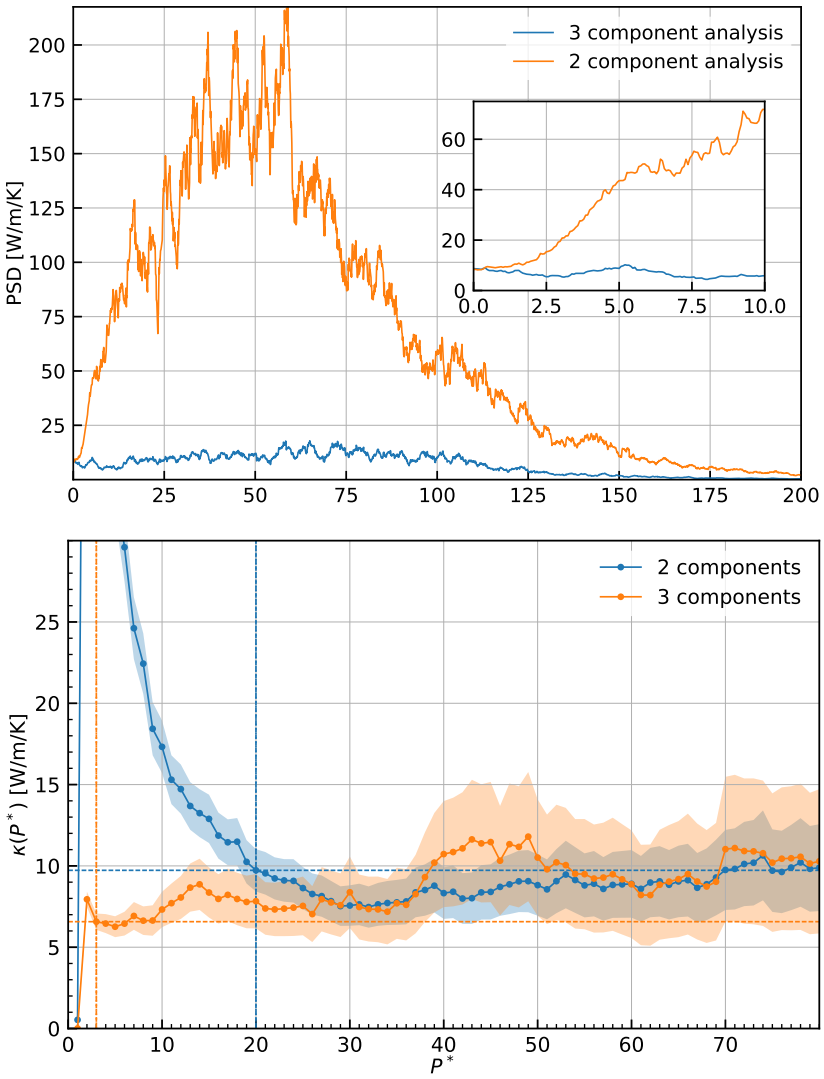


Figure 4.29: Comparison of 2 and 3 component analysis. It is seen that using the additional electronic current a lot of non-diffusing contributions are removed from the reduced periodogram. This allows one to take 3 cepstral coefficients instead of 20, resulting in a lower error despite the higher error at the same P^* that we have in the 3 component case due to the different statistics of the estimator.

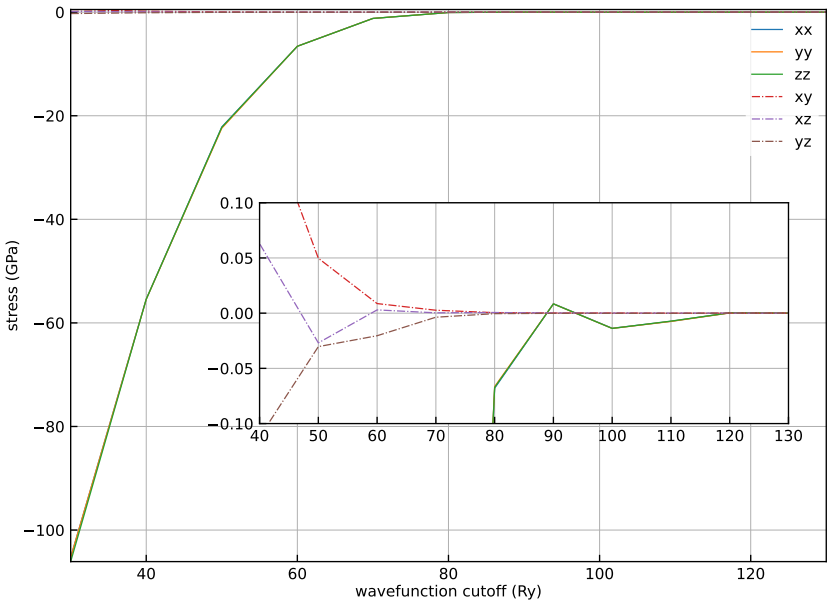


Figure 4.30: Convergence of the DFT stress on a snapshot from a superionic HCP configuration at 2800K, varying the wavefunction cutoff. We plotted the value of stress minus the value of stress at the cutoff of 130Ry so that the last point is zero by definition.

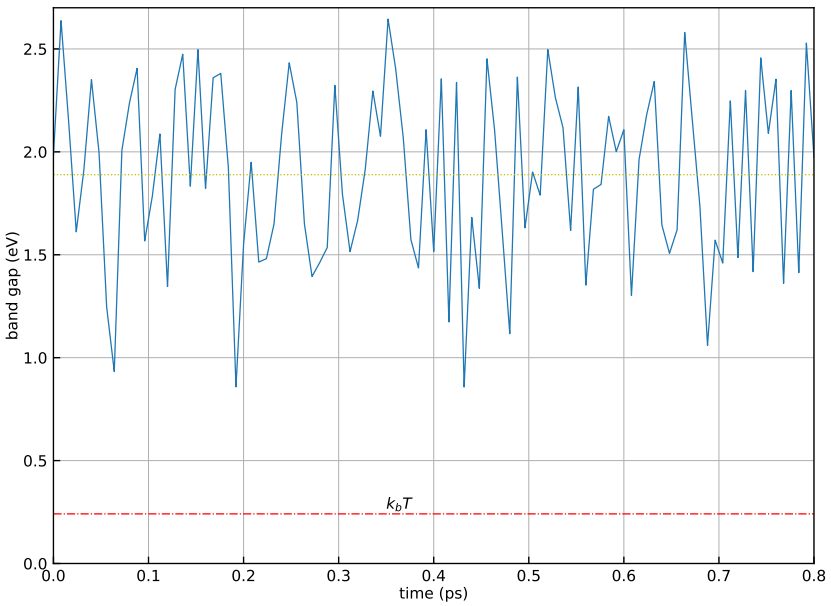


Figure 4.31: Time series of the band gap of a segment of a simulation of the superionic HCP system at 2800K. The dotted horizontal line is the value of k_bT in electronvolt at that temperature

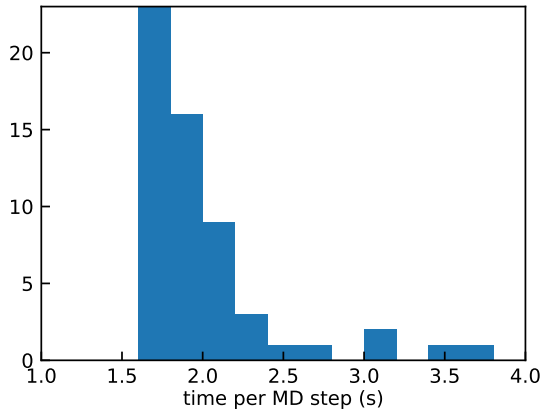


Figure 4.32: Average time per molecular dynamic step for the last 144-molecule system simulation on 4 GPU nodes of Marconi100.

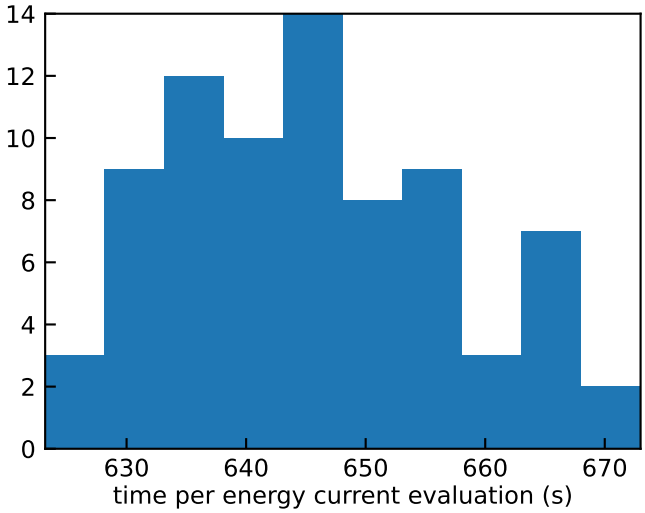


Figure 4.33: Average time per step in the QEHeat energy flux calculation, for many chunks of trajectory, for a single CPU node of Galileo100.

646s on a single node of GALILEO100. A histogram for the average time per chunk is shown in figure 4.33. We evaluated in total 22500 timesteps, consuming in total roughly 194000 Galileo100 CPU hours. This is a very big number that pushed us to develop a neural network potential, following [70, 77, 75, 78], for this system. The magnitude of hours needed to train a well-behaving neural network, with all the necessary tests and trial and error, is similar in magnitude to the one of this work. Still, in the end, one can get trajectories that cost many orders of magnitude less than the ones that we computed. Our computation of the energy flux presented two groups of 3-4 neighbor points that were not able to converge without the introduction of smearing in the SCF calculation. Those points were substituted with a set of points obtained by drawing a line between the good points at the boundary of the small interval.

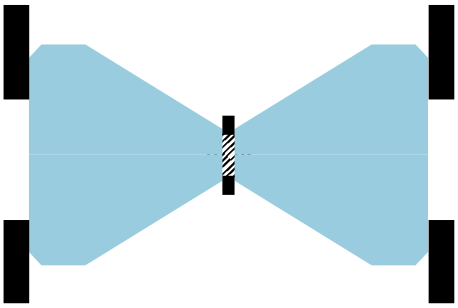


Figure 4.34: Scheme of a section of a diamond anvil cell (DAC). The scale of the sample, at the center between the two diamonds, is of the order of 100-140μm. The sample is kept in place by a gasket, and the diamonds are pushed against each other by two pieces of metal at each side. By applying a moderate force (for example with some screws), it is possible to obtain huge static pressures of the order of 100GPa on the very small surface area of the diamond tips.

4.8 Comparison with other works

Cavazzoni, Chiarotti, Scandolo, Tosatti, Bernasconi, Parrinello

[14] This work first predicted that superionic phases of water and ammonia exist at high pressure and temperatures. They performed NPT simulations on supercells with 64 ammonia molecules in a range of 30-300GPa and 300-7000K. The simulation length was limited from 1ps to 3ps. In this work, an HCP SI cell was found since they come from phase IV that has an HCP-like structure.

Diamond anvil cell (DAC) experiments

DAC is made of two diamonds with two flat tips pushed against each other, as schematized in figure 4.34. It is possible to statically reach pressures of many hundreds of GPa. The experiments can be performed also in a high-temperature regime, where usually the cell is cooled with some liquids and the heating is performed by an external laser. The diamonds need to be very pure, so Rahman and Brillouin scattering experiments can be performed. To measure the pressure inside the very small cell usually a small sample of a material whose equation of state is known is inserted inside the cell next to the sample, and it is measured independently to obtain the pressure. The temperature can be estimated by measuring the black body radiation. Possible issues of a high-temperature experiment are the extreme gradient of temperatures that are found and unexpected chemical reactions between various parts of the cell, which can lead to issues as can be the case in [56].

Ninet, Datchi, and Saitta

[55] In this work, the phase space of ammonia was explored up to 100GPa and 1000K with a DAC. They found experimentally that the superionic transition exists, and that depending on the path done to arrive at the superionic phase, it is possible to have both HCP and FCC structures for the nitrogen lattice. They also investigated with ab initio molecular dynamics the transition between phase III, phase IV, and superionic phase with great detail, stating that in the transition at the beginning the hydrogen atoms hop only on the planes, then, when the transition is completed, between the planes. The molecular

dynamics simulations were done with the QUANTUM ESPRESSO package and ultrasoft pseudopotential, with supercells of 32 ammonia molecules (2x2x2) or 64 molecules (4x2x2) and a cutoff of 32Ry. On the experimental side, the transition between solid phase IV and SI has a tiny change in a lattice parameter along the between-planes direction, which changes from 4.355Å to 5.365Å at 70GPa, with no discontinuities of the in-plane lattice parameters.

Bethkenhagen, French, and Redmer

[8] They performed ab-initio molecular dynamics with Vienna Ab Initio Simulation Package (VASP 4.6)[39, 40] with a 32-molecule system, a cutoff of 73Ry, and NVT dynamics. They state explicitly the fact that the nitrogen lattice structure is not known: *Moreover, we find an FCC nitrogen lattice in the superionic phase instead of the HPC structure predicted by Cavazzoni et al. This is in concordance with the fcc lattice we find in the neighboring rotationally disordered solid, which is also in very good agreement with the experimental and simulation data by Ninet et al. as shown in Fig. 2. However, this does not necessarily prove that the fcc nitrogen lattice is the thermodynamically favored lattice type in the superionic phase.*

Kimura and Murakami

[38] This DAC experiment explored ammonia up to 3000K and 65GPa. They propose that the SI phase behaves like a liquid over a certain temperature. This affirmation is based on a detected discontinuous decrease of the sound velocity while in the SI phase before the melting temperature of the sample. At the moment, our results do not agree with this affirmation. But a better investigation of the plane shift (that may be the manifestation of a phase transition) is necessary. This could provide a drop in the sound velocity.

5

Conclusions and future perspectives

The multicomponent theory of Green-Kubo thermal transport and the code that we developed were shown to have a very broad range of applications. They were used in many different works. The availability of all our codes provides a very solid and accessible starting point for any future work on thermal transport from equilibrium molecular dynamics. We think that the simplicity of the user interfaces developed in the `SporTran` tool allows the future researcher not to waste time implementing again the data analysis workflow that is full of complicated little details but to move the research to a higher level of complexity.

The application of our theory to superionic ammonia is an example of that. Without the work on multicomponent theory, we would have two issues. The issue of convergence of the GK integrals and the problem of decorrelating the energy current from the high-power components that do not contribute to thermal conductivity. Those are entirely removed. The multicomponent formula, implemented in the code, can remove any non-diffusing current. The data analysis is as simple as running the code with the flux time series as inputs. The same approach was used to develop automated tools for running the simulations. This allows us to forget many low-level details like the repetitive work of copying files around, preparing inputs, checking for convergence, etc., which we automated using the `AiiDA` python library.

The system that we studied, on the other hand, showed many unexpected challenges. We think that to understand how this material behaves under high PT conditions, and in particular if the plane shifts that we saw happen in the natural system, it is necessary to study cells at least one order of magnitude bigger. The neural network potential approach is the solution to the performance issue of the ab-initio simulations we used. We are now developing a neural network that successfully predicts the transport properties, compatible with our ab-initio simulations. The data in the `AiiDA` database produced in this thesis will be used to validate the neural network with very high confidence.

Despite all the problems, we provided our best estimate of the ab-initio thermal conductivity coefficient for superionic ammonia with the structure that we think is the most significant.

Bibliography

[1] H. Akaike. *Information theory and an extension of the maximum likelihood principle, in 2nd International Symposium on Information Theory pp. 267-281.* edited by B. N. Petrov and F. Csáki, 1972.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Trans. Autom. Control*, 19(6):716–723, December 1974. doi:10.1109/tac.1974.1100705.

[3] Dong An and Lin Lin. Quantum dynamics with the parallel transport gauge. *Multiscale Modeling & Simulation*, 18(2):612–645, 2020. doi:10.1137/18M1179304.

[4] T. W. Anderson. *The Statistical Analysis of Time Series*. Wiley, June 1994. doi:10.1002/9781118186428.

[5] Stefano Baroni, Riccardo Bertossa, Loris Ercole, Federico Grasselli, and Aris Marcolongo. Heat transport in insulators from ab initio green-kubo theory. In Wanda Andreoni and Sidney Yip, editors, *Handbook of Materials Modeling*, pages 809–844. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-319-44680-6_12.

[6] Riccardo Bertossa. Analisi: Your Swiss army knife of molecular dynamics analysis. *github*, 2017-2022. URL: https://github.com/rikigigi/analisi.

[7] Riccardo Bertossa, Federico Grasselli, Loris Ercole, and Stefano Baroni. Theory and numerical simulation of heat transport in multicomponent systems. *Phys. Rev. Lett.*, 122(25):255901, June 2019. arXiv:1808.03341, doi:10.1103/physrevlett.122.255901.

[8] Mandy Bethkenhagen, Martin French, and Ronald Redmer. Equation of state and phase diagram of ammonia at high pressures from ab initio simulations. *The Journal of Chemical Physics*, 138(23):234504, 2013. doi:10.1063/1.4810883.

[9] Martin Bilodeau and David Brenner. *Theory of Multivariate Statistics (Springer Texts in Statistics)*. Springer-Verlag, New York, 1999.

[10] B. P. Bogert, J. R. Healy, and J. W. Tukey. The quefrency analysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In *Proceedings of the Symposium on Time Series Analysis*, pages 209–243, 1963.

[11] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.*, 55(22):2471–2474, November 1985. doi:10.1103/physrevlett.55.2471.

[12] Christian Carbogno, Rampi Ramprasad, and Matthias Scheffler. Ab initio green-kubo approach for the thermal conductivity of solids. *Phys. Rev. Lett.*, 118(17):175901, April 2017. doi:10.1103/physrevlett.118.175901.

[13] H. B. G. Casimir. On onsager's principle of microscopic reversibility. *Rev. Mod. Phys.*, 17(2-3):343–350, April 1945. doi:10.1103/revmodphys.17.343.

[14] C. Cavazzoni, G. L. Chiarotti, S. Scandolo, E. Tosatti, M. Bernasconi, and M. Parrinello. Superionic and metallic states of water and ammonia at giant planet conditions. *Science*, 283(5398):44–46, January 1999. doi:10.1126/science.283.5398.44.

[15] Gerda Claeskens and Nils Lid Hjort. *Model Selection and Model Averaging*. Cambridge University Press, January 2001. doi:10.1017/cbo9780511790485.

[16] Pablo G Debenedetti. Fluctuation-based computer calculation of partial molar properties . I . Molecular dynamics simulation of constant volume fluctuations Molecular dynamics simulation of constant volume fluctuations. *J. Chem. Phys.*, 86(12):7126, 1987. doi:10.1063/1.452362.

[17] Loris Ercole, Riccardo Bertossa, and Sebastiano Bisacchi. Sportran public repository. *github*, 2022. URL: https://github.com/sissaschool/sportran.

[18] Loris Ercole, Riccardo Bertossa, Sebastiano Bisacchi, and Stefano Baroni. Sportran: A code to estimate transport coefficients from the cepstral analysis of (multivariate) current time series. *Computer Physics Communications*, page 108470, 2022. doi:10.1016/j.cpc.2022.108470.

[19] Loris Ercole, Aris Marcolongo, and Stefano Baroni. Accurate thermal conductivities from optimally short molecular dynamics simulations. *Sci. Rep.*, 7(1):15835, November 2017. doi:10.1038/s41598-017-15843-2.

[20] Loris Ercole, Aris Marcolongo, Paolo Umari, and Stefano Baroni. Gauge invariance of thermal transport coefficients. *J. Low Temp. Phys.*, 185(1-2):79–86, April 2016. doi:10.1007/s10909-016-1617-6.

[21] P Giannozzi, O Andreussi, T Brumme, O Bunau, M Buongiorno Nardelli, M Calandra, R Car, C Cavazzoni, D Ceresoli, M Cococcioni, N Colonna, I Carnimeo, A Dal Corso, S de Gironcoli, P Delugas, R A DiStasio, A Ferretti, A Floris, G Fratesi, G Fugallo, R Gebauer, U Gerstmann, F Giustino, T Gorni, J Jia, M Kawamura, H-Y Ko, A Kokalj, E Küçükbenli, M Lazzeri, M Marsili, N Marzari, F Mauri, N L Nguyen, H-V Nguyen, A Otero-de-la Roza, L Paulatto, S Poncé, D Rocca, R Sabatini, B Santra, M Schlipf, A P Seitsonen, A Smogunov, I Timrov, T Thonhauser, P Umari, N Vast, X Wu, and S Baroni. Advanced capabilities for materials modelling with quantum ESPRESSO. *J. Phys. Condens. Matter*, 29(46):465901, October 2017. doi:10.1088/1361-648x/aa8f79.

[22] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri,

Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials. *J. Phys. Condens. Matter*, 21(39):395502, September 2009. doi:10.1088/0953-8984/21/39/395502.

[23] Paolo Giannozzi, Oscar Baseggio, Pietro Bonfà, Davide Brunato, Roberto Car, Ivan Carnimeo, Carlo Cavazzoni, Stefano de Gironcoli, Pietro Delugas, Fabrizio Ferrari Ruffino, Andrea Ferretti, Nicola Marzari, Iurii Timrov, Andrea Urru, and Stefano Baroni. Quantum ESPRESSO toward the exascale. *J. Chem. Phys.*, 152(15):154105, April 2020. doi:10.1063/5.0005082.

[24] N. R. Goodman. The distribution of the determinant of a complex wishart distributed matrix. *The Annals of Mathematical Statistics*, 34(1):178–180, March 1963. doi:10.1214/aoms/1177704251.

[25] N. R. Goodman. Statistical analysis based on a certain multivariate complex Gaussian distribution (An introduction). *The Annals of Mathematical Statistics*, 34(1):152–177, March 1963. doi:10.1214/aoms/1177704250.

[26] Federico Grasselli. Investigating finite-size effects in molecular dynamics simulations of ion diffusion, heat transport, and thermal motion in superionic materials. *The Journal of Chemical Physics*, 156(13):134705, 2022. doi:10.1063/5.0087382.

[27] Federico Grasselli, Lars Stixrude, and Stefano Baroni. Heat and charge transport in H2O at ice-giant conditions from ab initio molecular dynamics simulations. *Nat. Commun.*, 11(1):3605, July 2020. doi:10.1038/s41467-020-17275-5.

[28] Melville S. Green. Markoff random processes and the statistical mechanics of Time-Dependent phenomena. II. irreversible processes in fluids. *J. Chem. Phys.*, 22(3):398–413, March 1954. doi:10.1063/1.1740082.

[29] S. R. De Groot and P. Mazur. *Non-Equilibrium Thermodynamics*. Dover Publications, 1984.

[30] D. R. Hamann. Optimized norm-conserving vanderbilt pseudopotentials. *Phys. Rev. B*, 88:085117, Aug 2013. doi:10.1103/PhysRevB.88.085117.

[31] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2.

[32] Eugene Helfand. Transport coefficients from dissipation in a canonical ensemble. *Phys. Rev.*, 119(1):1–9, July 1960. doi:10.1103/physrev.119.1.

[33] Ravit Helled and Jonathan J. Fortney. The interiors of uranus and neptune: current understanding and open questions. *Philosophical Transactions of the Royal Society A:*

Mathematical, Physical and Engineering Sciences, 378(2187):20190474, 2020. doi:10.1098/rsta.2019.0474.

[34] Ravit Helled, Nadine Nettelmann, and Tristan Guillot. Uranus and Neptune: Origin, Evolution and Internal Structure. *Space Science Reviews*, 216(3):38, March 2020. doi:10.1007/s11214-020-00660-3.

[35] Sebastiaan P. Huber, Spyros Zoupanos, Martin Uhrin, Leopold Talirz, Leonid Kahle, Rico Häuselmann, Dominik Gresch, Tiziano Müller, Aliaksandr V. Yakutovich, Casper W. Andersen, Francisco F. Ramirez, Carl S. Adorf, Fernando Gargiulo, Snehal Kumbhar, Elsa Passaro, Conrad Johnston, Andrius Merkys, Andrea Cepellotti, Nicolas Mounet, Nicola Marzari, Boris Kozinsky, and Giovanni Pizzi. AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Sci. Data*, 7(1):300, September 2020. doi:10.1038/s41597-020-00638-4.

[36] Jun Kang and Lin-Wang Wang. First-principles green-kubo method for thermal conductivity calculations. *Phys. Rev. B*, 96(2):20302, July 2017. doi:10.1103/physrevb.96.020302.

[37] A. Khintchine. Korrelationstheorie der station?ren stochastischen prozesse. *Math. Ann.*, 109(1):604–615, December 1934. doi:10.1007/bf01449156.

[38] Tomoaki Kimura and Motohiko Murakami. Fluid-like elastic response of superionic NH3 in Uranus and Neptune. *Proc. Natl. Acad. Sci.*, 118(14):e2021810118, March 2021. doi:10.1073/pnas.2021810118.

[39] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54:11169–11186, Oct 1996. URL: https://link.aps.org/doi/10.1103/PhysRevB.54.11169, doi:10.1103/PhysRevB.54.11169.

[40] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54:11169–11186, Oct 1996. URL: https://link.aps.org/doi/10.1103/PhysRevB.54.11169, doi:10.1103/PhysRevB.54.11169.

[41] A. M. Kshirsagar. Bartlett decomposition and wishart distribution. *The Annals of Mathematical Statistics*, 30(1):239–241, March 1959. doi:10.1214/aoms/1177706379.

[42] Ryogo Kubo. Statistical-mechanical theory of irreversible processes. i. general theory and simple applications to magnetic and conduction problems. *J. Phys. Soc. Jpn.*, 12(6):570–586, June 1957. doi:10.1143/jpsj.12.570.

[43] Ryogo Kubo, Mario Yokota, and Sadao Nakajima. Statistical-mechanical theory of irreversible processes. II. response to thermal disturbance. *J. Phys. Soc. Jpn.*, 12(11):1203–1211, November 1957. doi:10.1143/jpsj.12.1203.

[44] Wolfgang Lechner and Christoph Dellago. Accurate determination of crystal structures based on averaged local bond order parameters. *J. Chem. Phys.*, 129(11):114707, September 2008. doi:10.1063/1.2977970.

[45] J. S. Loveday, R. J. Nelmes, W. G. Marshall, J. M. Besson, S. Klotz, and G. Hamel. Structure of deuterated ammonia iv. *Phys. Rev. Lett.*, 76:74–77, Jan 1996. URL: https://link.aps.org/doi/10.1103/PhysRevLett.76.74, doi:10.1103/PhysRevLett.76.74.

[46] Cesare Malosso, Linfeng Zhang, Roberto Car, Stefano Baroni, and Davide Tisi. Viscosity in water from first-principles and deep-neural-network simulations. *npj Computational Materials*, 8(1):139, July 2022. doi:10.1038/s41524-022-00830-7.

[47] Aris Marcolongo. *Theory and ab initio simulation of atomic heat transport*. PhD thesis, Scuola Internazionale Superiore di Studi Avanzati, Trieste, 2014. URL: http://hdl.handle.net/20.500.11767/3897.

[48] Aris Marcolongo, Riccardo Bertossa, Davide Tisi, and Stefano Baroni. QEHeat: An open-source energy flux calculator for the computation of heat-transport coefficients from first principles. *Comput. Phys. Commun.*, 269:108090, December 2021. doi:10.1016/j.cpc.2021.108090.

[49] Aris Marcolongo, Loris Ercole, and Stefano Baroni. Gauge fixing for heat-transport simulations. *J. Chem. Theory Comput.*, 16(5):3352–3362, April 2020. doi:10.1021/acs.jctc.9b01174.

[50] Aris Marcolongo, Paolo Umari, and Stefano Baroni. Microscopic theory and quantum simulation of atomic heat transport. *Nat. Phys.*, 12(1):80–84, October 2015. doi:10.1038/nphys3509.

[51] Dominik Marx and Jürg Hutter. *Ab Initio Molecular Dynamics*. Cambridge University Press, April 2009. doi:10.1017/cbo9780511609633.

[52] Daya K. Nagar and Arjun K. Gupta. Expectations of functions of complex wishart matrix. *Acta Appl. Math.*, 113(3):265–288, January 2011. doi:10.1007/s10440-010-9599-x.

[53] S. Ninet and F. Datchi. High pressure–high temperature phase diagram of ammonia. *The Journal of Chemical Physics*, 128(15):154508, 2008. doi:10.1063/1.2903491.

[54] S. Ninet, F. Datchi, S. Klotz, G. Hamel, J. S. Loveday, and R. J. Nelmes. Hydrogen bonding in nd_3 probed by neutron diffraction to 24 gpa. *Phys. Rev. B*, 79:100101, Mar 2009. doi:10.1103/PhysRevB.79.100101.

[55] S. Ninet, F. Datchi, and A. M. Saitta. Proton disorder and superionicity in hot dense ammonia ice. *Phys. Rev. Lett.*, 108:165702, Apr 2012. doi:10.1103/PhysRevLett.108.165702.

[56] J. G. O. Ojwang, R. Stewart McWilliams, Xuezhi Ke, and Alexander F. Goncharov. Melting and dissociation of ammonia at high pressure and high temperature. *The Journal of Chemical Physics*, 137(6):064507, 2012. arXiv:https://doi.org/10.1063/1.4742340, doi:10.1063/1.4742340.

[57] Lars Onsager. Reciprocal relations in irreversible processes. i. *Phys. Rev.*, 37(4):405–426, February 1931. doi:10.1103/physrev.37.405.

[58] Lars Onsager. Reciprocal relations in irreversible processes. II. *Phys. Rev.*, 38(12):2265–2279, December 1931. doi:10.1103/physrev.38.2265.

[59] Paolo Pegolo. Theory and numerical simulation of heat conductivity in water. Master's thesis, Università degli Studi di Trieste, 2018.

[60] Paolo Pegolo, Stefano Baroni, and Federico Grasselli. Temperature- and vacancy-concentration-dependence of heat transport in Li3ClO from multi-method numerical simulations. *npj Computational Materials*, 8(1):24, January 2022. doi:10.1038/s41524-021-00693-4.

[61] Paolo Pegolo, Federico Grasselli, and Stefano Baroni. Oxidation states, thouless' pumps, and nontrivial ionic transport in nonstoichiometric electrolytes. *Phys. Rev. X*, 10:041031, Nov 2020. doi:10.1103/PhysRevX.10.041031.

[62] Magda Peligrad and Wei Biao Wu. Central limit theorem for Fourier transforms of stationary processes. *Ann. Probab.*, 38(5):2009–2022, September 2010. doi:10.1214/10-aop530.

[63] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77(18):3865–3868, October 1996. doi:10.1103/physrevlett.77.3865.

[64] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.*, 117(1):1 – 19, 1995. doi:10.1006/jcph.1995.1039.

[65] P Sindzingre, C Massobrio, and G Ciccotti. Calculation of partial enthalpies of an Argon-Kripton mixture by NPT molecular dynamics. *Chem Phys.*, 129:213–224, 1989.

[66] Lars Stixrude, Stefano Baroni, and Federico Grasselli. Thermal and tidal evolution of uranus with a growing frozen core. *The Planetary Science Journal*, 2(6):222, nov 2021. doi:10.3847/psj/ac2a47.

[67] Paul Tangney. On the theory underlying the car-parrinello method and the role of the fictitious mass parameter. *The Journal of chemical physics*, 124(4):044111, 2006. doi:10.1063/1.2162893.

[68] Paul Tangney and Sandro Scandolo. How well do car–parrinello simulations reproduce the born–oppenheimer surface? theory and examples. *The Journal of chemical physics*, 116(1):14–24, 2002. doi:10.1063/1.1423331.

[69] Davide Tisi. *Green-Kubo simulation of transport properties: from ab initio to neural-network molecular dynamics*. PhD thesis, Scuola Internazionale Superiore di Studi Avanzati, Trieste, 2022. URL: http://hdl.handle.net/20.500.11767/127869.

[70] Davide Tisi, Linfeng Zhang, Riccardo Bertossa, Han Wang, Roberto Car, and Stefano Baroni. Heat transport in liquid water from first-principles and deep neural network simulations. *Phys. Rev. B*, 104(22):224202, December 2021. doi:10.1103/physrevb.104.224202.

[71] Martin Uhrin, Sebastiaan P. Huber, Jusong Yu, Nicola Marzari, and Giovanni Pizzi. Workflows in AiiDA: Engineering a high-throughput, event-based engine for robust and modular computational workflows. *Nato. Sc. S. Ss. Iii. C. S.*, 187:110086, February 2021. doi:10.1016/j.commatsci.2020.110086.

[72] Léon Van Hove. Correlations in space and time and born approximation scattering in systems of interacting particles. *Phys. Rev.*, 95:249–262, Jul 1954. URL: https://link.aps.org/doi/10.1103/PhysRev.95.249, doi:10.1103/PhysRev.95.249.

[73] Jacobus A. van Meel, Laura Filion, Chantal Valeriani, and Daan Frenkel. A parameter-free, solid-angle based, nearest-neighbor algorithm. *J. Chem. Phys.*, 136(23):234107, June 2012. doi:10.1063/1.4729313.

[74] M.J. van Setten, M. Giantomassi, E. Bousquet, M.J. Verstraete, D.R. Hamann, X. Gonze, and G.-M. Rignanese. The pseudodojo: Training and grading a 85 element optimized norm-conserving pseudopotential table. *Computer Physics Communications*, 226:39–54, 2018. doi:https://doi.org/10.1016/j.cpc.2018.01.012.

[75] Han Wang, Linfeng Zhang, Jiequn Han, and Weinan E. Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228:178–184, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0010465518300882, doi:https://doi.org/10.1016/j.cpc.2018.03.016.

[76] Norbert Wiener. Generalized harmonic analysis. *Acta Math-djursholm.*, 55(0):117–258, 1930. doi:10.1007/bf02546511.

[77] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, Apr 2018. doi:10.1103/PhysRevLett.120.143001.

[78] Yuzhi Zhang, Haidi Wang, Weijie Chen, Jinzhe Zeng, Linfeng Zhang, Han Wang, and Weinan E. Dp-gen: A concurrent learning platform for the generation of reliable deep learning based potential energy models. *Computer Physics Communications*, 253:107206, 2020. URL: https://www.sciencedirect.com/science/article/pii/S001046552030045X, doi:https://doi.org/10.1016/j.cpc.2020.107206.