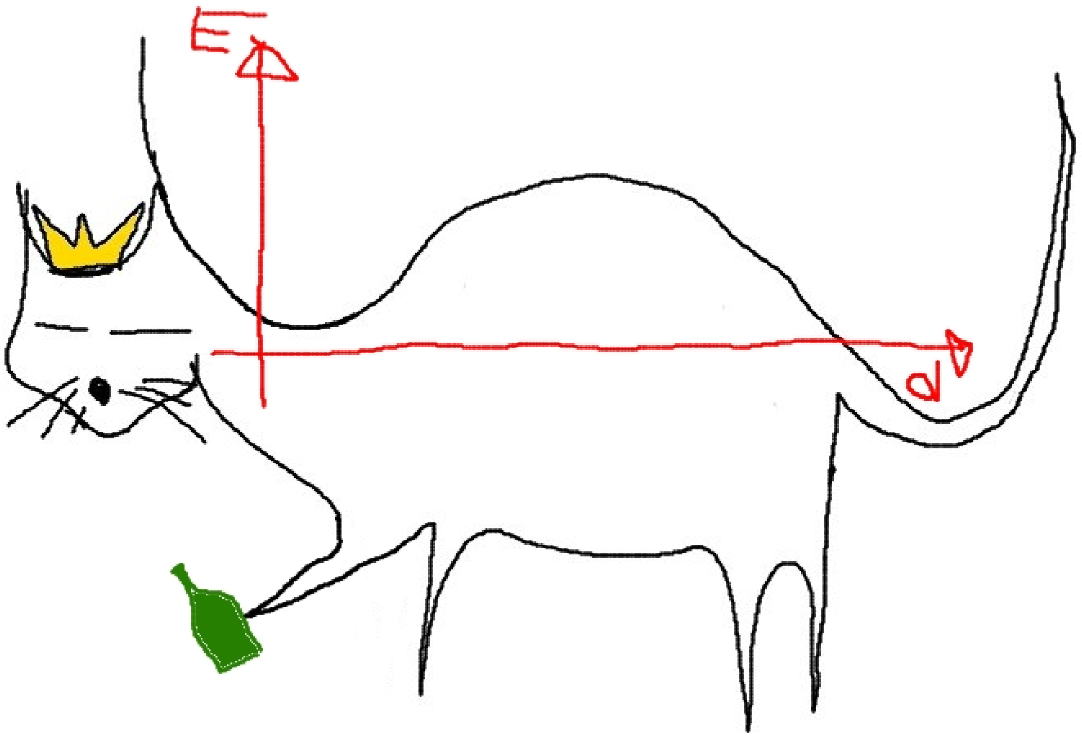
CHALMERS
UNIVERSITY OF TECHNOLOGY



An efficient approach for extracting anharmonic force constants from atomistic simulations

Master's thesis in Applied Physics

FREDRIK ERIKSSON

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2017

Master's thesis 2017

An efficient approach for extracting anharmonic force constants from atomistic simulations

FREDRIK ERIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Division of Materials and Surface Theory
Chalmers University of Technology
Gothenburg, Sweden 2017

An efficient approach for extracting anharmonic force constants
from atomistic simulations
FREDRIK ERIKSSON

Supervisor and examiner: Paul Erhart, Department of Physics
Supervisor: Erik Fransson, Department of Physics

Master's Thesis 2017
Department of Physics
Division of Materials and Surface Theory
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: hofcat - the mascot of the HiΦve project.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2017

An efficient approach for extracting anharmonic force constants
from atomistic simulations
Fredrik Eriksson
Department of Physics
Chalmers University of Technology

Abstract

Phonon theory is an important tool when analyzing solid state systems which is of high importance in modern technology. Central to the theory is the so called force constants (FCs) which determines the thermal behaviour. The super cell method is one way to extract the FCs from the force field of a displaced specimen but they are computationally expensive by requiring a lot of carefully prepared input data. By constructing an interatomic potential which uses the harmonic and higher order FCss as parameters the FCs can be extracted by a simple fitting procedure. Some small number of quasi random input configurations and the resulting force fields can contain enough information to extract all FCs with comparably low amount of computation. The underlying symmetries of the lattice must be used to reduce the number of free parameters in the model. This thesis demonstrates the implementation and application of the Force Constant Model. The model performs well and anharmonic FCs can be extracted. Given the FCs the model can be used as a potential and molecular dynamics can be performed yielding a direct method for computing thermal properties. Possible applications include analysis of thermal stabilization and phonon life time determination.

Keywords: Force Constants, Anharmonic, Molecular Dynamics, Symmetries, Phonons

Acknowledgements

Fredrik Eriksson, Gothenburg, October 2017

hofcat

Contents

1 Introduction

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.

John von Neumann

Being able to design new materials with specific properties is very important for out technological advancement. A good theoretical understanding of the microscopic processes is necessary to understand the macroscopic behaviour of the materials. Computer aided materials modeling bridges the gap between the fundamental theories and experiments. Modern computers make is possible to model the interaction between atoms in the material and how the resulting microscopic mechanisms emerge as macroscopic properties.

Atomic scale simulations

Atomic scale simulations is a branch of materials modeling. Among the tools available, Molecular dynamics (MD) is one of the most important tools for our understanding of the connection between microscopic models of the interacting atoms to the macroscopic world of thermodynamics. Its application ranges from the study of metallic alloys used in thermoelectrics[1] to the development of new pharmaceuticals. The method rely on an accurate underlying description of the interacting atoms which necessarily must take into account quantum mechanics. However such models (e.g. Density Functional Theory (DFT) or Hartree-Fock methods), called ab initio models or methods, are often too computationally expensive to be used directly in MD. Instead the forces computed from such models may be used to fit simpler functional forms, called interatomic potentials, describing the atomic interactions [2, 3]. Traditionally, simple forms are used with a few free parameters which can be fitted to experiments (e.g. neutron scattering [4] or Raman scattering[5]) or ab initio calculations. The parameters are often directly related to some physical property of the interaction such as bond strength, bond range or angular dependencies. Examples of such potentials are simple pair potentials such as the Lennard-Jones potential or the Embedded Atom Model[6] among others. Potentials with many body interactions include MEAM[7] and Stillinger-Weber[8] potentials. There is however a different approach where general functional forms with a lot of parameters are fitted to data (e.g. Gaussian Approximation Potentials[9], Artificial Neural Network potentials[10] or SNAP [11]). These potentials are in general not simple pair potentials and require a lot of time consuming ab initio simulations to determine all parameters. One such potential is the Force Constant Model. Apart from being of a general form it has a direct connection to the theory of lattice phonons which makes it interesting from a theoretical point of view.

Anharmonic effects

The functional form of the force constant model is equivalent to the Taylor expansion of the crystal energy in atomic displacements. This is the starting point for the theory of lattice phonons and the expansion paramteres (the force constants) completely determines the dynamics of the crystal atoms. Higher order terms in the expansion corresponds to non linear many body interactions among atoms and in reciprocal space to phonon scattering. In general third order force constants and higher introduces these interactions, called anharmonicity, in the model of lattice vibrations and many thermodynamic phenomena is not present without anharmonicity. For example the third order force constants can be used to calculate the first order correction to the phonon energies and their life times [12][13]. This can be used in Boltzmann transport theory to determine e.g. thermal conductivity. Thermal expansion is another example of a property not present in the harmonic theory. Of course, higher order force constants can be used in nearly all setting to compute more corrections to the phonon energies and state corrections but fourth order force constants have two especially interesting features. The first one is that any reasonable model of lattice vibrations must be stable. Only incorporating third order force constants the model will be mathematical unstable. This is not per se a practical problem in every setting but care must be taken. The second feature is dynamic stabilization. This is seen in some crystals where the zero Kelvin lattice is unstable but is stabilized by thermal motion [14][15][16][17]. This is in general thought to emerge when a phonon mode have local minima which is not probed by the large amplitude phonons. The system will have a broken symmetry which is only restored at high temperatures. Solid Helium and Titanium is known to be stabilized in this way.

Force Constant Models

The idea to use the force constants as the potential parameters makes it a general method. No special care must be taken for different materials and the fitting of the potential automatically yields the theoretically interesting force constants. Furthermore the model can be used as a potential and perform MD simulations which, if the model incorporates anharmonic terms, are expected to show previously mentioned anharmonic phenomena. The strength of the method in containing a lot of tunable parameters to fit different materials is also a weakness. The method must scale in such a way that the computational gain of using the force constants out weights the computational cost of fitting, i.e. the number of ab initio calculations needed to determine the free parameters must be low compared to more direct approaches like finite difference methods (the supercell method) [18]. In a naïve approach the number of free parameters in this model scales very badly. However the force constants obey the same symmetries as the underlying lattice and, from recent results in signal processing, methods have emerged that allows sparse solutions to optimization problems to be computed efficiently which have successfully been used to fit potentials [19][20][21]. The idea is that these kind of methods can extract the important parameters in our model with a comparably low amount of data. The process of fitting models to data is a field in itself and general data must be used in order to capture the correct force constants. In addition to ordinary over and under fitting issues different thermal regimes must be probed in order to capture a good set of parameters useful over a wide temperature range. To small displacements in the initial data and anharmonic effects is missed while to large displacements only probes an effective model potentially missing small features of the energy landscape. There is also an issue when designing the training data. The polynomial basis is not uncorrelated and thus the corresponding parameters might be unstable to the training data. Different methods exists to assess these kind of problems but ultimately, non-general training data, in general, gives a non-general fit. Of course being able to obtain even effective force constants opens up for techniques such as self consistent phonon approaches [22][13].

Purpose

This work work is dedicated to the problem of reducing the number of free parameters in the Force Constant Model. Symmetry conditions of the underlying lattice can be used for crystals with periodic boundary conditions and some force constants are expected to be small from physical arguments. A simple MD calculator will naturally follow from the formulation of the problem which may be used to do direct MD-simulations. The algorithm can also be used to categorize clusters in symmetrical systems which may be of its own use in e.g. classical cluster expansions.

2

Theory

2.1 Phonons and Force Constants

The quantum theory of phonons begins with the Born-Oppenheimer approximation. The energy of the crystal can then be expanded in a Taylor series in ionic displacements u_i away from some equilibrium positions R_i which defines the structure of the lattice.

$$H = H_0 + \Phi_i^\alpha u_i^\alpha + \frac{1}{2}\Phi_{ij}^{\alpha\beta} u_i^\alpha u_j^\beta + \frac{1}{3!}\Phi_{ijk}^{\alpha\beta\gamma} u_i^\alpha u_j^\beta u_k^\gamma + \dots \tag{2.1}$$

The Einstein summation convention is used for all repeated indices. Roman letters denote the atom labels and Greek letters denote the Cartesian coordinates $\alpha = x, y, z$. The expansion coefficients Φ are the force constants since they relate the displacements to the energy of the system. The atomic forces can conveniently be written as

$$-F_i^\alpha = \Phi_i^\alpha + \Phi_{ij}^{\alpha\beta} u_j^\beta + \frac{1}{2}\Phi_{ijk}^{\alpha\beta\gamma} u_j^\beta u_k^\gamma + \dots \tag{2.2}$$

The first term H_0 in (2.1) is a constant energy shift and can thus be chosen as the reference for a particular equilibrium lattice. However, this term is important when dealing with e.g. thermal expansion and other large scale dynamics. The second term also vanishes since we assume that the expansion is around static equilibrium, i.e. the expansion is around a stationary point in the energy landscape since it's the static force on each atom in absence of displacements as seen in equation (2.2).
Some nice literature on the subject of phonons can be found in the books by Ziman[23], Wallace[24], Born and Huang[25].

2.1.1 The harmonic crystal

The dynamics of the lattice is determined by the equations of motion derived from the Hamiltonian. If the displacements are small the harmonic approximation can be used and terms of order higher than two can be ignored in the Hamiltonian. The harmonic Hamiltonian looks like

$$H = \frac{1}{2m_i} p_i p_i + \frac{1}{2}\Phi_{ij} u_i u_j$$

in which the kinetic term has been reintroduced into and the Cartesian and labeling index have been combined. The solution to the harmonic problem is well known and by changing to a Fourier basis of plane waves

$$u_i = q_k \mathrm{e}^{-ikr_i}$$

and when assuming Born-von Karman boundary conditions, the Hamiltonian will decouple into a set of independent harmonic oscillators

$$H = \frac{1}{2m}|p_k|^2 + \frac{m}{2}\omega_k^2|q_k|^2$$

in the phonon coordinates q_k and conjugate momenta p_k. The procedure can take many different shapes but in one form or another the core object will be the dynamical matrix

$$D_{\alpha\beta}(\nu\mu, k) = \frac{1}{\sqrt{m_\nu m_\mu}N_0}\sum_{NP}\Phi^{\alpha\beta}(N\nu, P\pi)\mathrm{e}^{ik\cdot(R_{P\pi}-R_{N\nu})} \tag{2.3}$$

where α and β enumerate the Cartesian coordinates, ν and μ enumerate the ion types in a unit cell, N_0 is the number of unit cells in the supercell, N and P enumerate the unit cells and R is the position of an ion (which is uniquely identified by N and ν). The eigen values and eigenvectors of the dynamical matrix will determine the the possible energies and polarizations of a phonon mode with wave vector k.
The harmonic approximation is a good model to describe the heat capacity of crystals. Since the phonons are bosons they obey Bose-Einstein statistics

$$n(\omega_k) = \frac{1}{\exp\left(\frac{\hbar\omega_k}{k_B T}\right) - 1}$$

and the different modes can be populated thereafter. This connects the temperature with the energy of the lattice and it predicts the Debye T^3 law at low temperatures as well as the Dulong-Petit law at high temperatures. The harmonic model of phonons however does not predict two other important thermal properties of crystals, the thermal resistivity and thermal expansion. The thermal expansion can be accommodated for by the quasi harmonic approximation but actually is an anharmonic effect. The harmonic theory however have no coupling between modes (and thus infinite thermal conductivity) which is easiest dealt with using first order perturbation theory.

2.1.2 First order perturbation

The harmonic part of our Hamiltonian is well defined and have an analytical solution which describes the real world to a great extent. The theory also has a natural expansion parameter in the order of the force constants included in the expansion. Thus it is natural to try and use perturbation theory to find corrections to the harmonic approximation. To first order we can include the third order force constants into the Hamiltonian. The harmonic model can be used as the ground state and the anharmonic higher order terms is treated as a small perturbations. This is easiest done by second quantization where the phonon creation and annihilation operators are introduced

$$A_k = \frac{1}{\sqrt{2\hbar\omega_k}}(\omega_k q_k + ip_{-k}) \quad A_k^\dagger = \frac{1}{\sqrt{2\hbar\omega_k}}(\omega_k q_{-k} - ip_k)$$

The creation and anihilation operators obey the commutation relations

$$[A_k, A^\dagger_{k'}] = \delta_{kk'}$$

and thus represent a set of independent bosons. The harmonic Hamiltonian becomes

$$H_2 = \sum_k \hbar\omega_k \left(A^\dagger_k A_k + \frac{1}{2} \right)$$

which simply counts the number of phonons in each mode k with energy ω_k. The third order Hamiltonian can then be shown to be

$$H_3 = \sum_{kk'k''} \Phi_{kk'k''}(A_k + A^\dagger_k)(A_{k'} + A^\dagger_{k'})(A_{k''} + A^\dagger_{k''})$$

where Φ is given by a quite complicated function of the phonon energies and polarization vectors as well as the third order force constants. More specifically the phonon linewidth is given by

$$\begin{aligned}\tau_\lambda^{-1} = \frac{36\pi}{\hbar^2} \sum_{\lambda'\lambda''} |\Phi_{-\lambda\lambda'\lambda''}|^2 \,[\,&(n_{\lambda'} + n_{\lambda''} + 1)\delta(\omega - \omega_{\lambda'} - \omega_{\lambda''}) + \\ &(n_{\lambda'} - n_{\lambda''})(\delta(\omega + \omega_{\lambda'} - \omega_{\lambda''}) - \delta(\omega - \omega_{\lambda'} - \omega_{\lambda''}))\,]\end{aligned}$$

where n is the Bose occupation from (2.1.1). The three-phonon coupling strength is given by (hmb)

$$\begin{aligned}\Phi_{\lambda\lambda'\lambda''} = &\frac{1}{\sqrt{N}}\frac{1}{3!} \sum_{\kappa\kappa'\kappa''} \sum_{\alpha\beta\gamma} W_\alpha(\kappa,\lambda) W_\beta(\kappa',\lambda') W_\gamma(\kappa'',\lambda'') \sqrt{\frac{\hbar}{2m_\kappa\omega_\lambda}} \sqrt{\frac{\hbar}{2m_{\kappa'}\omega_{\lambda'}}} \sqrt{\frac{\hbar}{2m_{\kappa''}\omega_{\lambda''}}} \\ &\times \sum_{l'l''} \Phi_{\alpha\beta\gamma}(0\kappa, l'\kappa', l''\kappa'') \mathrm{e}^{iq'[r(l'\kappa') - r(0\kappa)]} \mathrm{e}^{iq''[r(l''\kappa'') - r(0\kappa)]} \mathrm{e}^{i(q+q'+q'')r(0\kappa)} \Delta(q + q' + q'')\end{aligned}$$

which is only dependent on the polarizations from the harmonic solution (2.3) and the third order force constants. See also [12]. In many body perturbation theory this represents phonon-phonon interactions and would correspond to first order Feynman diagrams. From this new Hamiltonian the energy corrections can be calculated and by Fermis golden rule

$$\Lambda_{i\to f} = \frac{2\pi}{\hbar} |\langle f| H_3 |i\rangle|^2 \rho_f$$

the transition amplitude and thus life time τ of a phonon mode can be calculated. The life time can be used to determine a scattering integral in the Boltzmann transport theory and from that calculate the thermal conductivity.

$$(v \cdot \nabla T)\frac{\partial n_0}{\partial T} = \frac{\delta n}{\tau}$$

2.1.3 Higher order force constants

Apart from the possibility to use higher order force constants to calculate higher order corrections to energies and modes they have some inherent interesting features.

In phonon-phonon interactions the 4-phonon scattering may be of equal strength as the 3-phonon interaction and be equally important e.g. in determining life times to use in transport theory. The fourth order term is also important since a Hamiltonian in which the highest order term is odd does not define a stable system. Thus a lattice with only a third order interaction may be stable for small displacements but there is no lower bound for arbitrary displacements. Not only does a fourth order term stabilize the lattice it may also introduce local stationary points in configuration space. Consider a phonon mode with a fourth order interaction shaped like a Mexican hat. If the phonon displacement (corresponding to a low temperature) is low enough it may get stuck in one of the minima whereas for large displacements it will oscillate back and forth with a mean displacement equal to zero. This mode may take the whole lattice from one structure to another. This is known as dynamical (or thermal) stabilization as the structure is unstable for low temperatures, i.e. the vacuum state is unstable. This corresponds to a broken symmetry that would be restored for high enough energies i.e. high temperatures.

2.2 Clusters and Symmetries

The Hamiltonian of the crystal is

$$H = \frac{1}{2}\Phi_{ij}u_i u_j + \frac{1}{3!}\Phi_{ijk}u_i u_j u_k + \dots$$

Depending on the symmetry of the system the force constants may be related in some way, i.e. there is some rule to transform e.g. Phi_{ijk} to $Phi_{i'j'k'}$. The number of force constants grow rapidly ($\mathcal{O}(N^n)$) as a function of the number of atoms N in the system and the order n of the force constant. To reduce the number of force constants and the number of parameters in each force constant, the symmetries of the lattice can be used. In addition, some force constants are expected to give negligible contribution to the total energy under normal conditions and may be completely ignored. In general this is expected if the constituent atoms are far apart and physically it describes an interaction cutoff.
The interactions between individual atoms are often considered to be of a finite range. This is in general a good approximation and is found in e.g. both the Ising model as a direct cutoff and in the Lennard-Jones potential as an indirect cutoff. The interaction length is often seen as a single distance between two atoms but can be generalized to higher orders by introducing the concept of a cluster as a collection of atoms $(i, j, k, \dots)$. A cluster is considered to be a valid cluster if all the inter atomic distances in the cluster are less than a certain cutoff r. Two atoms are neighbors if their equilibrium lattice positions are within the cutoff. Each force constant has an associated cluster and this gives us a simple rule whether to include a specific term in the expansion or not. So instead of summing over all possible sequences of indices $(i, j, \dots)$ lets just sum over sequences where the corresponding atoms are all neighbors of each other, i.e. valid clusters. Then the nth order contributions in the Hamiltonian can be written

$$H_n = \frac{1}{n!}\sum_{\text{clusters}} \Phi_{ij\dots}u_i u_j \dots$$

So instead of being a sum over N^n terms it has been brought down to $\mathcal{O}(N(r^3)^n)$ which is linear in N. Two properties of the force constants can be used to dramatically reduce the number of free parameters in the model. In an ordered system some clusters may be related by some symmetry. That is the lattice positions of one cluster may be one-to-one mapped to the lattice positions of another cluster via a symmetry operation of the underlying lattice. The same symmetry operation can be used to relate their force constants. It might also happen that a cluster is invariant under some symmetry. Thus the force constant must also be invariant under the same symmetry and the elements within the force constant may be related.
Some terminology. Clusters that can be mapped to each other by a symmetry of the lattice are said to be equivalent or that they are in the same orbit. If two clusters (necessarily in the same orbit) can be mapped onto each other by a trivial lattice translation they are said to be part of the same orientation family within the orbit.

2.2.1 Atom label symmetries

First of all $\Phi_{ij\dots}$ is symmetric in the label-index meaning that

$$\Phi^{\alpha\beta\dots}_{ij\dots} = \Phi^{P(\alpha\beta\dots)}_{P(ij\dots)}$$

where P is a permutation operator. E.g. $\Phi^{\alpha\beta}_{ij} = \Phi^{\beta\alpha}_{ji}$. This means that a cluster is a multiset and not a sequence, i.e. order doesn't matter. So instead of summing over all possible clusters we only need to consider ordered clusters and compensate with a pre factor to compensate for the left out equivalent terms.

$$H_n = \frac{1}{n!} \sum_{\substack{\text{ordered}\\ \text{clusters}}} m_\omega \Phi_\omega u_{\omega_1} u_{\omega_2} \dots$$

where m_ω is the multiplicity of the cluster in the expansion and is related to the multinomial coefficients as

$$m(i, j, \dots) = \frac{n!}{k_1! \dots k_m!}$$

where k is the multiplicity of each element of the set of indices in the sequence. E.g. $(1, 1, 3, 5, 5, 5)$ would have $k_1 = 2$, $k_2 = 1$ and $k_3 = 3$ meaning 2 ones, 1 three and 3 fives. A cluster (i, j, k) where i, j and k are all different have multiplicity 6 whereas a cluster (i, i, j) with i and j not equal will have multiplicity 3. This symmetry in labeling indices are always present independent of any symmetry of the underlying lattice and stems from the fact that we are doing physics.
Furthermore suppose the enumerated crystal is transformed by a lattice symmetry. This will in general lead to a new enumeration of the atoms as well as every displacement vector gets rotated. The crystal energy however is invariant under such transformations. So let a symmetry transformation with rotation R map a cluster $(i, j, \dots)$ onto $(i', j', \dots)$. The invariance of the energy can be written as

$$\Phi^{\alpha\beta\dots}_{ij\dots} u^\alpha_i u^\beta_j \dots = \Phi^{\alpha'\beta'\dots}_{i'j'\dots} R^{\alpha'\alpha} u^\alpha_i R^{\beta'\beta} u^\beta_j \dots$$

but since this must hold for all configurations of u

$$\Phi^{\alpha\beta\dots}_{ij\dots} = \Phi^{\alpha'\beta'\dots}_{i'j'\dots} R^{\alpha'\alpha} R^{\beta'\beta} \dots$$

Two clusters in the same orbit have exactly this property. The orbit is thus defined by some prototype cluster and its force constant and the different clusters it may be transformed to together with the appropriate rotation matrix. This means that the clusters can be ordered into groups, which is called orbits, sharing the same force constant apart from a well defined rotation for every cluster. Naturally, in each orbit the clusters in the same orientation family will share the same rotation matrix and thus have equal force constants apart from an permutation since we consider ordered clusters. The Hamiltonian contribution can now schematically be written

$$H_n = \frac{1}{n!} \sum_{\text{Orbits}} m_{\text{Orbit}} \Phi_{\text{Orbit}} \sum_{\text{Orientations}} R_{\text{Orientation}} \cdots \sum_{\substack{\text{ordered}\\ \text{clusters}}} u_{P(i)} \dots$$

2.2.2 Cartesian symmetries (eigensymmetries)

If a cluster is mapped to some permutation of itself during a lattice symmetry the corresponding force constant must obey the same symmetry. E.g. the cluster $(i, j, \dots)$ is mapped to the cluster $P(i, j, \dots)$ by some symmetry with rotation R.

$$\Phi^{\alpha\beta\dots}_{ij\dots} = \Phi^{\alpha'\beta'\dots}_{P(ij\dots)} R^{\alpha'\alpha} R^{\beta'\beta} \dots \tag{2.4}$$

but the labels may be freely permuted

$$\Phi^{\alpha\beta\dots}_{ij\dots} = \Phi^{P^{-1}(\alpha'\beta'\dots)}_{ij\dots} R^{\alpha'\alpha} R^{\beta'\beta} \dots \tag{2.5}$$

and renamed

$$\Phi^{\alpha\beta\dots}_{ij\dots} = \Phi^{\alpha'\beta'\dots}_{ij\dots} R^{P(\alpha')\alpha} R^{P(\beta')\beta} \dots \tag{2.6}$$

This can be thought of as a tensor eigen-equation. There are some symmetries we want to take into account. The first one is the symmetry under the spatial transformations of the crystal. This can be straightforwardly incorporated by noting those symmetry operations and index permutations that map a cluster onto itself. We also want to incorporate the general symmetry that two equal indices can be interchanged. This can be solved in many ways but one way is to flatten the tensor to a vector and noting that the above relations become ordinary matrix eigenvalue equations. Define a map M taking a collection of indices to one single index $M(\alpha, \beta, \dots) \to a$. Now the right hand sides rotation matrices acts like a map from two collections of indices $((\alpha, \beta, \gamma), (\alpha', \beta', \gamma')) \to (a, b)$. With this the above relation for some cluster can be written like

$$\Phi^{a} = \Phi^{b} R^{P(b)a} \tag{2.7}$$

Any vector fulfilling the above equation can be reshaped by the inverse map M^{-1} to an equivalent tensor again. Since the number of indices is exactly the number of parameters in the tensor it is easy to see that there is a possibility for all indices to be different. As with vectors any two tensors which fulfills the equation may be added with some parameters to make up a new tensor which also fulfills the equation. So assume $(\phi_i)^b$ is a valid eigenvector (with eigenvalue 1) then Φ can be written as

$$\Phi^{\alpha\beta\ldots} = \sum_i a_i(\phi_i)^{\alpha\beta\ldots}$$

Now there may be several symmetries that map a cluster onto itself. Each of these give rise to a particular form of the matrix R^{ij}. Suppose we have a nth order tensor and m symmetries. We go through the preceding procedures for each one and ends up with m matrices of size 3^n. We place those on top of each other and ends up with a $m3^n \times 3^n$ matrix. This will contain redundant information and may be row-reduced to a $3^n \times 3^n$ matrix again.
A 3^n by 3^n can also be constructed that encodes the index symmetry. Consider a cluster $(0, 0, 1, 1, 1)$. Given a specific array of Cartesian indices (y, x, z, y, y), which maps to some index i by the map M, an order can be imposed since the element at index (x, y, y, y, z) should be the same. This new order maps to a new index j and any arrangement of indices that can be reordered to the above form is also equivalent.
There are also constraints imposed by the flying ice cube. All the forces must sum to zero and the constraint can be written as

$$\sum_i \Phi^{\alpha\beta\ldots}_{ij\ldots} = 0$$

All the details how the different constraints and symmetries are imposed are described in the Algorithm section.

2.2.3 Forces

Now, given the reduced force constants we want to calculate forces. The Hamiltonian at this point is

$$H_n = \frac{1}{n!}\sum_{\text{Orbits}} m_{\text{Orbit}}\Phi_{\text{Orbit}} \sum_{\text{Orientations}} R_{\text{Orientation}} \cdots \sum_{\text{clusters}} u_i \ldots$$

The force acting on an atom i due to some force constant of degree n is give by

$$F_i = -\frac{1}{n!}\sum_{\text{Orbits}} m_{\text{Orbit}}\Phi_{\text{Orbit}} \sum_{\text{Orientations}} R_{\text{Orientation}} \cdots \sum_{\text{clusters}} \frac{\partial}{\partial u_i}(u_j \ldots \tag{2.8}$$

From the above it is apparent that only clusters containing i will contribute to the force. Lets see how atom 2 gets its force from cluster $(1, 2, 2)$.

$$F_2^\delta = -\frac{1}{3!}\frac{3!}{1!2!}\Phi^{\alpha\beta\gamma}_{122}\frac{\partial}{\partial u_2^\delta}(u_1^\alpha u_2^\beta u_2^\gamma) = -\frac{1}{1!2!}\Phi^{\alpha\beta\gamma}_{122}(u_1^\alpha \delta^{\beta\delta} u_2^\gamma + u_1^\alpha u_2^\beta \delta^{\gamma\delta}) \tag{2.9}$$

which simplifies to

$$F_2^\beta = -\frac{1}{1!2!}2\Phi^{\alpha\beta\gamma}_{122}u_1^\alpha u_2^\gamma \tag{2.10}$$

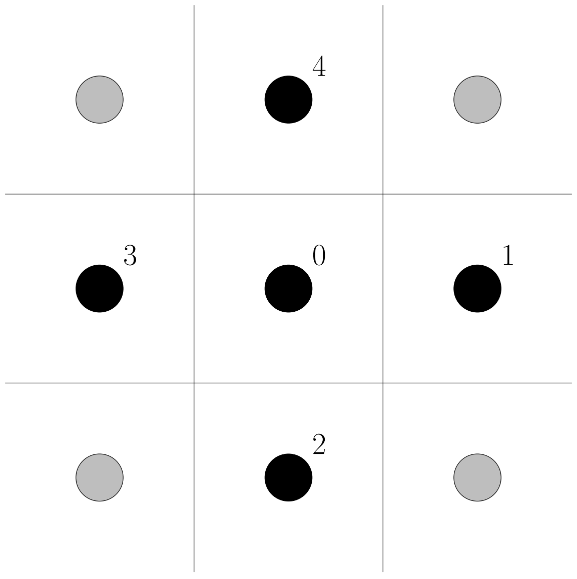


Figure 2.1: A simple square lattice with only nearest neighbor interactions. Atom 0 is the center atom and will form exactly five clusters. $[0, 0]$, $[0, 1]$ etc. Only two orbits exist, the on site and the nearest neighbor orbit. The former with one cluster and the latter with four clusters distributed in two orientation families.

So the prefactor comes from two places. First the multiplicity of the total cluster and then the regained multiplicity when taking the derivative. So to get the forces is in principle the same summation as evaluating the total energy. From the above it should be apparent how to generalize and it will be showed in detail later. Note that the summation should only be over ordered clusters!

2.2.4 A simple example on a square lattice

Consider the simple cubic crystal in 2D. Now we want to formulate a simple harmonic force constant model with only on-site and nearest neighbor interactions. The lattice and enumerated atoms within the cutoff can be seen in figure 2.1. Now we can write a partial Hamiltonian with all terms including the center atom. Let $u_{ij}^{\alpha\beta} = u_i^\alpha u_j^\beta$ and the partial Hamiltonian becomes

$$H = \frac{1}{2!}(\Phi_{00}^{\alpha\beta}u_{00}^{\alpha\beta} + \Phi_{01}^{\alpha\beta}u_{01}^{\alpha\beta} + \Phi_{02}^{\alpha\beta}u_{02}^{\alpha\beta} + \Phi_{03}^{\alpha\beta}u_{03}^{\alpha\beta} + \Phi_{04}^{\alpha\beta}u_{04}^{\alpha\beta} + \tag{2.11}$$

$$\Phi_{10}^{\alpha\beta}u_{10}^{\alpha\beta} + \Phi_{20}^{\alpha\beta}u_{20}^{\alpha\beta} + \Phi_{30}^{\alpha\beta}u_{30}^{\alpha\beta} + \Phi_{40}^{\alpha\beta}u_{40}^{\alpha\beta}) \tag{2.12}$$

Now the lower line will give the exact same contributions as the corresponding element of the upper line. E.g.

$$\sum_{\alpha\beta} \Phi_{01}^{\alpha\beta}u_{01}^{\alpha\beta} = \sum_{\alpha\beta} \Phi_{10}^{\alpha\beta}u_{10}^{\alpha\beta}$$

So we can skip the lower line and just multiply the corresponding terms in the upper one with 2. This is the multiplicity of a cluster of order two with two different atoms. The cluster 00 has multiplicity 1 and gets no extra prefactor.

$$H = \frac{1}{2!}(\Phi_{00}^{\alpha\beta}u_{00}^{\alpha\beta} + 2\Phi_{01}^{\alpha\beta}u_{01}^{\alpha\beta} + 2\Phi_{02}^{\alpha\beta}u_{02}^{\alpha\beta} + 2\Phi_{03}^{\alpha\beta}u_{03}^{\alpha\beta} + 2\Phi_{04}^{\alpha\beta}u_{04}^{\alpha\beta}) \tag{2.13}$$

Now lets examine Φ_{00}. It will obvious be invariant under a 90 degree rotation clockwise. Thus the corresponding system of equations would be

$$\Phi_{00}^{\alpha\beta} = \Phi_{00}^{\alpha'\beta'} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}^{\alpha'\alpha} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}^{\beta'\beta} \tag{2.14}$$

Now lets apply the map M from before and write out the equations

$$\Phi_{00}^{xx} = \Phi_{00}^{xx} R^{xx} R^{xx} + \Phi_{00}^{xy} R^{xx} R^{yx} + \Phi_{00}^{yx} R^{yx} R^{xx} + \Phi_{00}^{yy} R^{yx} R^{yx} \tag{2.15}$$

$$\Phi_{00}^{xy} = \Phi_{00}^{xx} R^{xx} R^{xy} + \Phi_{00}^{xy} R^{xx} R^{yy} + \Phi_{00}^{yx} R^{yx} R^{xy} + \Phi_{00}^{yy} R^{yx} R^{yy} \tag{2.16}$$

$$\Phi_{00}^{yx} = \Phi_{00}^{xx} R^{xy} R^{xx} + \Phi_{00}^{xy} R^{xy} R^{yx} + \Phi_{00}^{yx} R^{yy} R^{xx} + \Phi_{00}^{yy} R^{yy} R^{yx} \tag{2.17}$$

$$\Phi_{00}^{yy} = \Phi_{00}^{xx} R^{xy} R^{xy} + \Phi_{00}^{xy} R^{xy} R^{yy} + \Phi_{00}^{yx} R^{yy} R^{xy} + \Phi_{00}^{yy} R^{yy} R^{yy} \tag{2.18}$$

which can be written as (remember that $R^{xx} = 0$, $R^{xy} = 1$, $R^{yx} = -1$ and $R^{yy} = 0$)

$$\begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} \tag{2.19}$$

In the same spirit the symmetry from $\Phi_{00}^{\alpha\beta} = \Phi_{00}^{\beta\alpha}$ can be written as

$$\begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} \tag{2.20}$$

Combining those two equation we get

$$\begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} = 0 \tag{2.21}$$

Notice the -1 added in the diagonals when subtracting the right hand side of the two equation before joining them. The above equation get row reduced to

$$\begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} = 0 \tag{2.22}$$

and the only eigenvector with eigenvalue 0 is

$$\begin{bmatrix} \Phi_{00}^{xx} \\ \Phi_{00}^{xy} \\ \Phi_{00}^{yx} \\ \Phi_{00}^{yy} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.23}$$

Meaning that the on site orbit only has one eigentensor which is

$$\phi_{00}^{\alpha\beta} = a \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.24}$$

for some parameter a to be determined by the fit to forces for some specific material. Next up is the $[0,1]$ cluster with the force constant Φ_{01}. This cluster can obviously be transformed to all the other clusters $[0,2]$, $[0,3]$ and $[0,4]$. For example: a 90 degree rotation maps $[0,1]$ onto $[0,2]$ which can be translated with a lattice translation onto $[4,0]$ (notice the permutation). In the same manner $[0,3]$ has the same force constant as $[0,1]$ but for a permutation. So now we can write

$$H = \frac{1}{2!}(a\phi_{00}^{\alpha\beta}u_{00}^{\alpha\beta} + 2\Phi_{01}^{\alpha\beta}u_{01}^{\alpha\beta} + 2\Phi_{01}^{\alpha'\beta'}R_{-90}^{\alpha'\alpha}R_{-90}^{\beta'\beta}u_{02}^{\alpha\beta} + 2\Phi_{01}^{\beta\alpha}u_{03}^{\alpha\beta} + 2\Phi_{01}^{\alpha'\beta'}R_{-90}^{\alpha'\beta}R_{-90}^{\beta'\alpha}u_{04}^{\alpha\beta}) \tag{2.25}$$

Notice how the $[0,1]$ is unchanged, the $[0,2]$ gets a rotation -90 degrees since the tensors transform contravariant, the $[0,4]$ get the same treatment as $[0,2]$ but also a permutation and $[0,3]$ gets only a permutation. Before we write it all out we notice that an improper rotation mapping up to down the $[0,1]$ cluster stays the same. The transformation is

$$R_{-y} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.26}$$

so, taking a shortcut compared to the previous computation, we see that

$$\begin{bmatrix} \Phi_{01}^{xx} & \Phi_{01}^{xy} \\ \Phi_{01}^{yx} & \Phi_{01}^{yy} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \Phi_{01}^{xx} & \Phi_{01}^{xy} \\ \Phi_{01}^{yx} & \Phi_{01}^{yy} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.27}$$

reducing to

$$\begin{bmatrix} \Phi_{01}^{xx} & \Phi_{01}^{xy} \\ \Phi_{01}^{yx} & \Phi_{01}^{yy} \end{bmatrix} = \begin{bmatrix} \Phi_{01}^{xx} & -\Phi_{01}^{xy} \\ -\Phi_{01}^{yx} & \Phi_{01}^{yy} \end{bmatrix} \tag{2.28}$$

Meaning that $[0,1]$ has two eigentensors

$$(\phi_1)_{01}^{\alpha\beta} = b \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad (\phi_2)_{01}^{\alpha\beta} = c \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.29}$$

Now lets evaluate the terms in the Hamiltonian, the first $[0,1]$ is as before

$$\Phi_{01}^{\alpha\beta} = \begin{bmatrix} b & 0 \\ 0 & c \end{bmatrix} \tag{2.30}$$

the second $[0,2]$ is kind of expected changing the role of x and y

$$\Phi_{02}^{\alpha\beta} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} b & 0 \\ 0 & c \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} c & 0 \\ 0 & b \end{bmatrix} \tag{2.31}$$

the third $[0,3]$ is just a transpose of the $[0,1]$ doing nothing

$$\Phi_{03}^{\alpha\beta} = \begin{bmatrix} b & 0 \\ 0 & c \end{bmatrix} \tag{2.32}$$

while $[0, 4]$ is just the transpose of $[0, 2]$ also changing nothing.

$$\Phi_{04}^{\alpha\beta} = \begin{bmatrix} c & 0 \\ 0 & b \end{bmatrix} \tag{2.33}$$

Up to now we have three parameters a for the on-site and b and c for the only orbit. But remember that we also must consider the constraints from the sum rules. Lets evaluate

$$\sum_i \Phi_{0i} = 0 \tag{2.34}$$

and written out

$$\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} + \begin{bmatrix} b & 0 \\ 0 & c \end{bmatrix} + \begin{bmatrix} c & 0 \\ 0 & b \end{bmatrix} + \begin{bmatrix} b & 0 \\ 0 & c \end{bmatrix} + \begin{bmatrix} c & 0 \\ 0 & b \end{bmatrix} = 0 \tag{2.35}$$

Meaning that a, b and c can't be set independent but some hyper parameters must be used. One choice could be to use b and c unchanged as hyper parameters B and C

$$\begin{bmatrix} B & C \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix} \tag{2.36}$$

This is the mechanical background of the algorithm and even though this example was trivial all the key elements were present. We started with the partial full Hamiltonian containing all terms including atom 0. Then we identified the atom label symmetry and introduced the prefactor which was dependent on the multiplicity of the cluster. Next we identified some eigensymmetries for the on-site orbit which together with the atom label symmetry reduced it to only one eigentensor. We solved this tensor equation by flattening the indices and after solving the corresponding eigenvalue problem we mapped back to our tensor. Next we saw that rotations which were symmetries of the lattice could be used to relate all the clusters so that they belong to the same orbit. This orbit also had some eigensymmetries resulting in two independent parameters and eigentensors. Finally we used the sum rule constraint to find possible linear combinations of the parameters fulfilling the sum rule.
Having found the Hamiltonian it is easy to find the resulting forces of the model as a function of the hyper parameters and the displacements. If the displacements and forces are given the hyper parameters can be fitted by some method e.g. linear least squares or some regularized derivative of it.

3 Algorithm

3.1 Setup

Before any real work can be done the scene must be set.

1. The primitive cell, i.e. the structure, must be defined and the corresponding lattice symmetries of the crystal must be determined.
2. The neighboring atoms to the center cell must be found and enumerated.
3. Neighbor lists must be created in preparation for finding unique clusters

3.1.1 Symmetries of the lattice

The structure of the crystal is given by the primitive cell which consists of three objects:

1. The primitive cell metric given by three vectors a, b and c in Cartesian coordinates

$$\texttt{cell} = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix} \tag{3.1}$$

   Notice the convention of having the basis vectors as row vectors.

2. The basis, given by a set of points in the primitive cell, where each point is represented by its scaled coordinates s with respect to the primitive cell metric.

$$\texttt{basis} = \begin{bmatrix} \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}, & \dots \end{bmatrix} \tag{3.2}$$

   Thus the Cartesian position of the first atom in the basis is given by

$$\begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix} \tag{3.3}$$

   When representing a basis atom, each component i of a scaled coordinate s can only take values in the interval $0 \leq s_i < 1$.

3. The atomic elements of each atom in the basis. They can be represented by an array of integers referring to the table of elements.

$$\texttt{numbers} = [14, \dots] \tag{3.4}$$

As an example the primitive cell of Si diamond could look like

$$\texttt{cell} = \begin{bmatrix} 0 & 2.715 & 2.715 \\ 2.715 & 0 & 2.715 \\ 2.715 & 2.715 & 0 \end{bmatrix} \tag{3.5}$$

$$\texttt{basis} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0.25 & 0.25 & 0.25 \end{bmatrix} \end{bmatrix} \tag{3.6}$$

$$\texttt{numbers} = \begin{bmatrix} 14 & 14 \end{bmatrix} \tag{3.7}$$

indicating a lattice constant of 5.43. The cell metric (or simply just "cell") and the basis is very central and used throughout the algorithm.
This primitive cell will be thought of as sitting in the center of a infinite crystal stretching out in all dimensions by repeating the primitive cell. Naturally any atom in the crystal can be indexed by its position in a primitive cell and the corresponding offset compared to the original primitive cell. Thus the original primitive cell at offset $[0, 0, 0]$ will be called the center cell and the corresponding atoms will be called the center atoms.
Next the symmetries of the lattice must be found. The symmetries for common structures as bcc and fcc etc. are well known but in order to deal with any structure some software is usually used. One such software is spglib[26]. Given a primitive cell all the transformations which leaves the structure invariant is returned. A transformation consists of two objects: one rotation matrix R given in scaled coordinates with respect to the primitive cell metric and one translation vector T also with respect to the primitive cell metric. The symmetry operation acts as follows on a scaled position v

$$v' = Rv + T$$

All the transformations of the lattice is returned as two lists, one with the rotation matrices and one with the translation vectors.
In python these first steps would for the example be

```python
cell = [[0, 2.715, 2.715],
        [2.715, 0, 2.715],
        [2.715, 2.715, 0]]
basis = [[0, 0, 0], [0.25, 0.25, 0.25]]
numbers = [14, 14]

import spglib
symmetries = spglib.find_primitive((cell, basis, numbers))
rotation_matrices = symmetries['rotations']
translation_vectors = symmetries['translations']
```

3.1.2 Identify the neighborhood

By indexing the atoms in the vicinity of the center cell a cluster can be denoted by a list of integers indicating which atoms are part of it. It is also clear that any cluster in the crystal can be found by finding the clusters containing the center atoms and

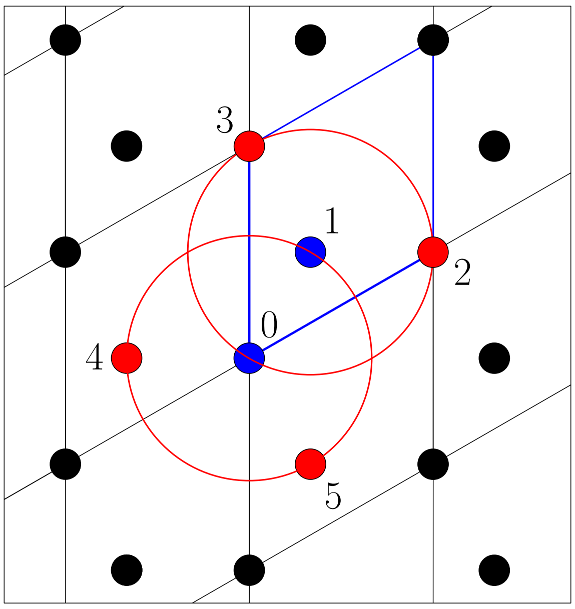


Figure 3.1: The center cell is indicated by blue lines. The two basis atoms 0 and 1 are also colored blue. The maximum cutoff is indicated by a red circle originating from each basis atom in the center cell. The atoms 2,3,4 and 5 are within the cutoff and are also colored red and given an index. This particular cutoff is quite small and only allows nearest neighbor interactions.

then translating the cluster. Thus only clusters containing the center atoms must be found. Since only clusters where each inter atomic distance are smaller than a maximum cutoff are valid only atoms within said cutoff from any of the center atoms may form clusters with the center atoms.
Consider figure 3.1. Since the cutoff are very small only nearest neighbor can be part of a valid cluster. Note also that since no cluster containing more than two atoms can be formed the only valid clusters of order higher than two must contain the same atom more than once, e.g. $[0, 5, 5]$ is a valid third order cluster but not $[0, 4, 5]$ since 4 and 5 are too far apart.
The atoms of interest are stored in a list where their position in the list indicates their given index. Every atom are represented by their basis index and offset. The atom list of figure 3.1 would be

$$\texttt{atom\_list} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{3.8}$$

the first column is the basis index and the next two are the offsets. In a real application in 3D there would be four rows, one for the basis index and three for the offsets. For reasons explained later it is important that the first atoms in the list are the center atoms.s

```
# Init. list
```

```python
atom_list = []
# Populate first elements with center atoms
for i in range(len(basis)):
  atom_list.append([i, 0, 0, 0])
# Populate the following with atoms from the neighborhood
for atom in prim:
  basis_indices, offsets = atom.get_neighbors()
  for basis_index, offset in zip(basis_indices, offsets):
    basis_index_offset = [basis_index] + list(offset)
    # A neighbor may be a neighbor to more than one atom in the center
   cell
    if basis_index_offset not in atom_list:
      atom_list.append(basis_index_offset)
```

Now we have a list of all the atoms that may form a cluster with any of the atoms in the center cell. All of the atoms will form at least one cluster with one or more of the atoms in the center cell. No atom outside the neighborhood will ever be part of a valid cluster with any of the atoms in the center cell.

3.1.3 Generate neighbor matrices

We want to be able to generate clusters with different inter atomic distances depending on the order of the cluster, e.g. pair interactions have longer range than many body interactions. A convenient tool to have when generating clusters are neighbor matrices which tells us whether two atoms are neighbors given some cutoff. An element $M_{\text{order}}(i, j)$ in a neighbor matrix is true if atom i and atom j are neighbors given some cutoff defined by the order.

We use the software ASE's[27] functionality to find neighbors to atoms. For each cutoff belonging to some order, e.g. 4 Å for pairs and 3 Å for triplets, we loop over all atoms and check whether the distance between them is less than the cutoff. If it is the corresponding element in the neighbor matrix is set to true.

```python
import numpy as np
neighbor_matrices = {}
for order in range(2, max_order + 1):
  scaled_positions = []
  for atom in atom_list:
    basis_index = atom[0]
    offset = atom[1:]
    spos = get_spos(basis_index, offset, basis)
    scaled_positions.append(spos)
  atoms = Atoms(cell=cell, scaled_positions=scaled_positions, numbers=
   numbers, pbc=False)
  nl = NeighborList([cutoffs[order]/2] * len(atoms), skin=0, bothways=
   True)
  nl.build(atoms)
  neighbor_matrix = np.zeros((len(atoms), len(atoms)), dtype=bool)
  for atom in atoms:
    indices, offsets = nl.get_neighbors(atom.index)
    neighbor_matrix[atom.index, indices] = True
  neighbor_matrices[order] = neighbor_matrix
```

3.2 Cluster Generation

Now we want to create a list of all the clusters that the atoms in the center cell can be part of. The indices of the atoms of our candidate cluster is generated in lexicographical order from the atoms in the neighborhood. E.g. for a triplet we start by generating $[0, 0, 0]$, then $[0, 0, 1]$ up to $[0, 0, n-1]$ if there are n atoms in the neighborhood list. Then it goes around again and do $[0, 1, 1]$ (instead of $[0, 1, 0]$ since it is the same cluster as $[0, 0, 1]$ which we have already generated). Then they are checked against:

1. Are all the atoms, represented by the indices, neighbors with respect to the properties of the cluster? I.e. do they fulfill the cutoff criteria? This is checked with the neighbor matrices.
2. Is any of the atoms in the center ($[0, 0, 0]$) cell part of the cluster?

An option here is whether the cluster $[0, 0, 1]$ for example should be checked against the cutoff for pairs or triplets. It arises from the third order expansion coefficients but involve only two atoms. In general force constants containing fewer unique atoms are in general more important.

If both checks pass we say that the cluster is a valid cluster and should be added to the list of clusters. Since we made sure earlier that the first atoms in the atoms list belong to the center cell we can stop generating clusters of a specific order once the first index is larger or equal the number of atoms in the basis.

```
from itertools import combinations_with_replacement
cluster_list = []
for order in range(2, max_order + 1):
  for cluster in combinations_with_replacement(range(len(atoms)),
   repeat=order):
    if cluster[0] >= len(prim):
      break
    elif np.all(neighbor_matrices[order][cluster, :][:, cluster]):
      cluster_list.append(cluster)
```

Up to third order, the valid clusters from figure 3.1 would be

$$\texttt{cluster\_list} = \begin{bmatrix} \begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 4 & 5 & 1 & 2 & 3 \end{matrix} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 5 & 1 & 1 & 1 & 2 & 3 \\ 0 & 1 & 4 & 5 & 1 & 4 & 5 & 1 & 2 & 3 & 2 & 3 \end{matrix} \end{bmatrix}^T \tag{3.9}$$

3.3 Cluster Categorization

3.3.1 Create the permutation map

To determine if two clusters a and b belong to the same orbit we might do as follows.

1. Pick a lattice symmetry.
2. For every atom in cluster a calculate the scaled position of the atom.
3. Transform the scaled positions with the symmetry transformation.

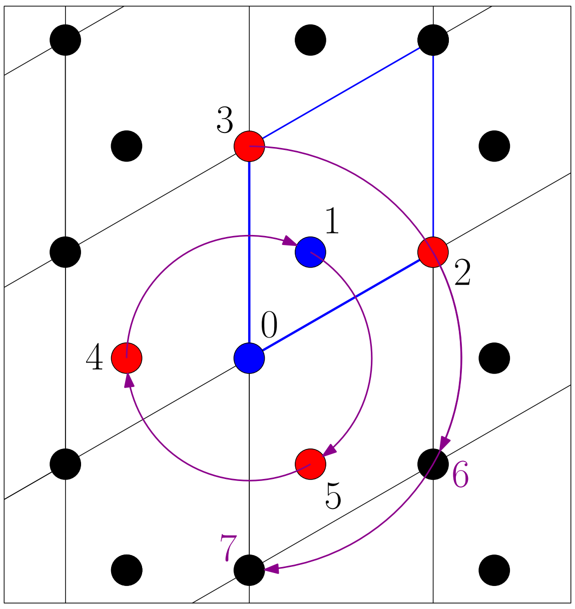


Figure 3.2: A simple rotation will map the atoms in the neighborhood of the center cell onto new positions, some which may be outside the original neighborhood. By picking a cluster e.g. $[1, 2, 5]$ it is easy to see that it belongs to the same orbit as $[5, 7, 4]$. The cluster is however not interesting since 5 is too far away from the other atoms violating the cutoff.

4. Check if the new scaled positions are the same as the one in cluster b
5. If the scaled positions was not the same. Check if a single lattice translation can map the transformed cluster a' to b.

Consider figure 3.2. During a 120 degree clockwise rotation the atoms change as

$$\begin{array}{ccc} 0 & \rightarrow & 0 \\ 1 & \rightarrow & 5 \\ 2 & \rightarrow & 7 \\ 3 & \rightarrow & 6 \\ 4 & \rightarrow & 1 \\ 5 & \rightarrow & 4 \end{array} \tag{3.10}$$

This means that e.g. cluster $[0, 5]$ is in the same orbit as $[0, 4]$. Note also that atom 2 and 3 were mapped to atoms not in the atom list. Atoms that appear during the categorization step but are not in the atom list will be denoted as ghost atoms. They are placed in a separate list continuing the indexing of the original atom list. E.g. the first ghost atom will be placed at the first position in the ghost list but will be thought of as having an index n if there were n atoms in the original atom list (remember zero indexing).

This game can be played again now using a 60 degree rotation clockwise and a

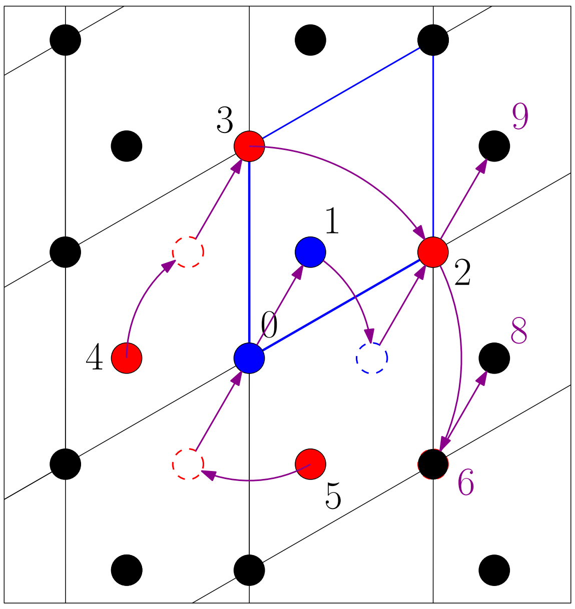


Figure 3.3: Some lattices like the graphene lattice has two atoms in the basis. To completely map out all orbits it is necessary to perform both a rotation and translation.

translation, see figure 3.3 so that the mapping becomes

$$\begin{aligned} 0 &\rightarrow 1 \\ 1 &\rightarrow 2 \\ 2 &\rightarrow 8 \\ 3 &\rightarrow 9 \\ 4 &\rightarrow 3 \\ 5 &\rightarrow 0 \end{aligned} \tag{3.11}$$

From this we can see that $[0, 1]$ is the same as $[1, 2]$ for example. But there may be many other symmetry transformations and a lot of clusters so some systematic way of determining the clusters must be used. One way is to define the permutation map. The permutation map is an matrix with as many rows as there are atoms in the neighborhood. The number of columns are as many as there are symmetries of the lattice. It is essentially a look-up table to see which atom is mapped to another if a lattice symmetry is applied to the lattice. So, for example, in the permutation map matrix at element $[i, s]$ is a number j. This means that when symmetry number s where applied to the lattice, atom number i was mapped to atom number j in the atom list or ghost list.
If the above two symmetries were the first two symmetries after the identity symmetry the permutation map would look like

$$\texttt{permutation\_map} = \begin{bmatrix} 0 & 0 & 1 & \dots \\ 1 & 5 & 2 & \dots \\ 2 & 7 & 8 & \dots \\ 3 & 6 & 9 & \dots \\ 4 & 1 & 3 & \dots \\ 5 & 4 & 0 & \dots \end{bmatrix} \tag{3.12}$$

and the ghost list

$$\texttt{ghost\_list} = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & -1 \\ 1 & 1 & -1 \\ 1 & 1 & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{3.13}$$

So the algorithm goes on as first pick a symmetry. With the symmetry transform all atoms in the atom list. If new atoms appear, add the to the ghost list. Add the indices of the new atoms in the corresponding column of the permutation map. Move on an pick a new symmetry.

```
ghost_list = []
pm = np.zeros((len(atom_list), len(rotations)), dtype=int)
for sym_index, (R, T) in enumerate(zip(rotations, translations)):
    for atom_index, atom in enumerate(atom_list):
        spos = atom_to_spos(atom, basis)
        new_spos = np.dot(R, spos) + T
        new_atom = spos_to_atom(new_spos, basis)
        if new_atom not in atom_list:
            ghost_list.append(new_atom)
            new_atom_index = len(atom_list) + ghost_list.index(new_atom
    )
        else:
            new_atom_index = atom_list.index(new_atom)
        pm[atom_index, sym_index] = new_atom_index
```

3.3.2 Categorize clusters and identify eigensymmetries

The basic idea is that we can pick a cluster, say $[0, 0, 1]$ and a symmetry, say symmetry number 2 corresponding to some rotation and translation of the lattice. By looking in the permutation map at column 2 and row 0,0 and 1 we can figure out what cluster $[0, 0, 1]$ was transformed to. This mean that these two clusters are equivalent and so are their force constants apart from a rotation given by the rotation matrix of symmetry number 2.
Furthermore all clusters that can be translated to each other by a lattice translation belong to the same orbit. This means that not only are we interested in the symmetries in the permutation map we also must translate the clusters to completely map out all relations. Consider again figure 3.2. The symmetry transformation maps $[0, 1]$ onto $[0, 5]$. But since $[0, 5]$ can be translated onto $[3, 1]$ by a lattice translation we can deduce that $[0, 1]$, $[0, 5]$ and $[0, 5]$ all belong to the same orbit and furthermore that $[0, 5]$ and $[3, 1]$ are oriented in the same way thus belonging to the same orientation family. In general a cluster can be made to map out all equivalent clusters in the same orbit by translating it so that every atom in the cluster has offset $[0, 0, 0]$ at least once.
Finally, during the categorization step we group clusters into orbits which means that we can relate the force constants of the different orientation families with each other by the lattice symmetries. We are also interested in symmetries which maps a cluster onto itself. Those symmetries, called eigensymmetries, can be used to

reduce the number of free parameters in the clusters force constant (in addition to the fundamental atom label symmetry). Thus the eigensymmetries should be noted during the categorization.

Now we have everything we need in order to categorize the clusters we previously generated.

1. Pick the first cluster from the list of clusters.
2. If the cluster has already been categorized, pick the next cluster in the list and repeat this step.
3. If the cluster wasn't categorized before it can be used as a prototype cluster for an orbit.
4. Pick the first symmetry.
5. Use the permutation map to see how the prototype cluster would transform under the symmetry.
6. Pick the first atom in the new cluster.
7. Translate all the atoms with the picked atoms offset.
8. If this translated cluster is a permutation of the prototype cluster add the symmetry and permutation to the families' list of eigensymmetries.
9. If the cluster is already categorized, pick the next atom and go to step 7
10. If the cluster is not categorized, mark the cluster as categorized and add its index and permutation to the family.

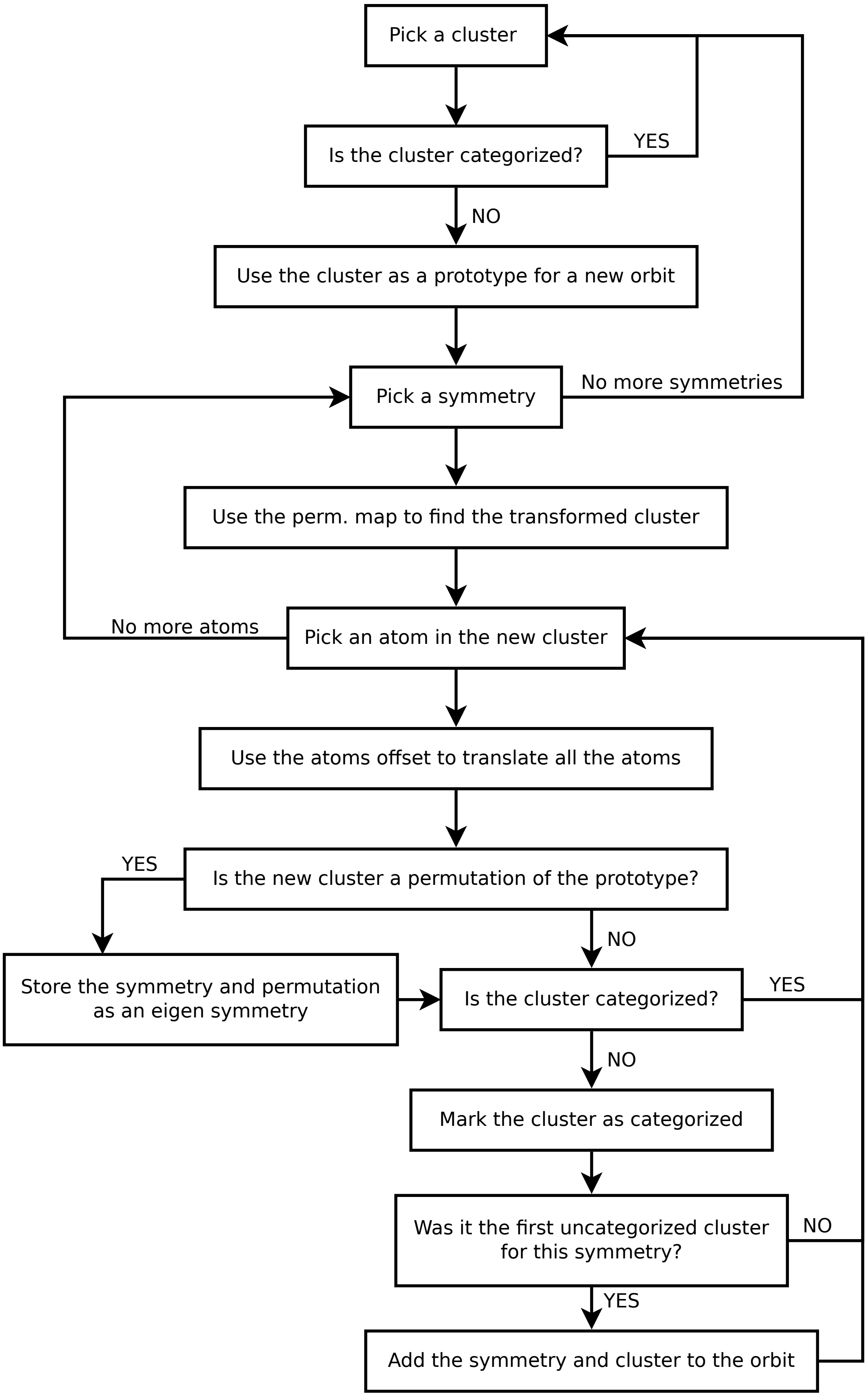
In order to save some space all the permutations is precomputed in a list and referred to by the index in the list.

```python
FIX!!
is_categorized = [False] * len(cluster_list)
for cluster_index, cluster in enumerate(cluster_list):
  if is_categorized(cluster_index): continue
  orbit = Orbit()
  for symmetry in symmetries:
    transformed_cluster = permutation_map.get_transformed_cluster(
  cluster, symmetry)
    for i, atom in enumerate(transformed_cluster):
        translated_cluster = transformed_cluster.translate(-atom.offset
  )
        if sorted(translated_cluster.indices) == orbit.prototype.
  indices:
            orbit.add_eigen_symmetry(symmetry, translated_cluster.
  permutation)
        if is_categorized(translated_cluster.index): continue
        if i == 0: orbit.append(translated_cluster, symmetry)
        is_categorized(translated_cluster) = True
```

3.4 Force constant reduction

3.4.1 Reduction by eigensymmetries

The different types of clusters have now been categorized into orbits sharing the same force constant apart from a rotation. Some symmetry operations of the lattice left certain clusters invariant and can thus be used to reduce the number of components


Pick a cluster
Is the cluster categorized?
YES
NO
Use the cluster as a prototype for a new orbit
Pick a symmetry
No more symmetries
Use the perm. map to find the transformed cluster
No more atoms
Pick an atom in the new cluster
Use the atoms offset to translate all the atoms
YES
Is the new cluster a permutation of the prototype?
NO
Store the symmetry and permutation as an eigen symmetry
Is the cluster categorized?
YES
NO
Mark the cluster as categorized
Was it the first uncategorized cluster for this symmetry?
NO
YES
Add the symmetry and cluster to the orbit

in that orbits force constant. During the categorization step, if a cluster was mapped to itself, the corresponding rotation and permutation was noted. Following the description in section 2.2.2 we write out the tensor equation to solve

$$\Phi_{ijk}^{\alpha\beta\gamma} = \Phi_{ijk}^{\alpha'\beta'\gamma'} R^{\beta'\alpha} R^{\alpha'\beta} R^{\gamma'\gamma}$$

Where it is evident that (i, j, k) was mapped to (j, i, k) since the corresponding indices is permuted. We must also keep in mind that should the rotation matrices been given in scaled coordinates they must be transformed to a Carteesian system by the cell metric V

$$R_{\text{Cartesian}} = V^T R_{\text{scaled}} V^{-T}$$

The map M used is a simple flatten map which uses the number of dimensions (the order n) and size of each dimension (which is 3). For a third order map it would be

$$M(i, j, k) \to a = i \cdot 3^2 + j \cdot 3^1 + k \cdot 3^0$$

```python
def M(*args):
    a = 0
    for n, arg in enumerate(reversed(args)):
        a += arg * 3**n
    return a
```

and we are looking for the matrix corresponding to the expanded equation equivalent to the above

$$\Phi^a = M^{ab}\Phi^b$$

We begin by creating the expanded matrix given a rotation matrix and permutation vector which orders the cluster again.

```python
import numpy as np
import itertools as it
def get_expanded_matrix(rotation_matrix, arg_sort_permutation):
    order = len(arg_sort_permutation)
    M = np.ones((3**order, 3**order))
    for args in it.product((0,1,2), repeat=order):
        row = M(args)
        for args_primed in it.product((0,1,2), repeat=order):
            col = M(args_primed)
            for arg, arg_prim_permuted in zip(args, args_primed[
   arg_sort_permutation]):
                M[row, col] *= rotation_matrix[arg_prim_permuted, arg]
    return M
```

If a cluster is invariant under two or more symmetries, its M-matrices can be combined by slightly altering the above equation

$$(M^{ab} - I^{ab})\Phi^b = 0$$

and combining it with another matrix M' gives

$$\begin{bmatrix} M - I \\ M' - I \end{bmatrix} \Phi = 0$$

which may be row reduced to a new matrix $\tilde{M}$

$$\tilde{M}\Phi = 0$$

In addition to the spatial symmetries we also want symmetry in equivalent indices. For this we can create a special matrix

```python
def get_index_symmetry(cluster):
    M = np.zeros((3**order, 3**order))
    (atom_indices, cluster_indices, multiplicity) = np.unique(cluster,
   return_counts=True, return_index=True)
    for args in it.product((0,1,2), repeat=order):
        row = M(args)
        new_args = []
        for i, m in zip(cluster_indices, multiplicity):
            new_args.append(sorted(args[i, i+m]))
        col = M(new_args)
       M[rwo, col] = 1
    return M
```

Combining the above we get

```python
from scipy.linalg import lu
M_tilde = get_index_symmetry(cluster)
M_tmp = np.zeros((2 * 3**order, 3**order))
for R, p in zip(list_of_roations, list_of_arg_sorts):
    M = get_expanded_matrix(R, p)
    M_tmp[:3**order, :] = M_tilde
    M_tmp[3**order:, :] = M - np.eye((3**order, 3**order))
    M_tilde = lu(M_tmp)[2]
```

Now the equivalent question of asking what tensors solve ??? is to ask what vectors solve ???.

```python
from scipy.linalg import eig
eigen_tensors = []
eig_vals, eig_matrix = eig(M_tilde)
for eig_val, eig_vec in zip(eig_vals, eig_matrix.T):
    if abs(eig_val) > tol:
        continue
    eig_tensor = np.zeros((3,) * order)
    for args in it.product((0,1,2), repeat=order):
        eig_tensor[args] = eig_vec[M(args)]
    eigen_tensors.append(eig_tensor)
```

which gives us the possible solutions as

$$\Phi = \sum_i a_i \phi_i$$

where ϕ is the eigen tensors from the last step.

3.4.2 Sum rule constraints

The sum rule constraint can be implemented as a constraint on the possible parameters in the force constants. Consider the constraint from atom 0 in the center cell

$$\sum_k \Phi_{0ijk}^{\alpha\beta\gamma\delta} = 0 \quad \forall \alpha\beta\gamma\delta ij \tag{3.14}$$

and partition the force constant into its eigentensors ϕ and parameters a

$$\sum_{kl} a_l (\phi_l)_{0ijk}^{\alpha\beta\gamma\delta} = 0 \quad \forall \alpha\beta\gamma\delta ij \tag{3.15}$$

Now since if the above hold for some i and j and all Cartesian indices the same must hold for any permutation of i and j. Thus only ordered clusters need to be considered. Now lets fix i and j to some numbers (atoms) and write the sum as

$$\sum_l a_l \sum_k (\phi_l)_{0ijk}^{\alpha\beta\gamma\delta} = 0 \quad \forall \alpha\beta\gamma\delta ij \tag{3.16}$$

since every index except l is either fixed or contracted we end up with a linear constraint on the parameters a_l. Playing the same game for every i, j and Cartesian index we end up with a matrix equation describing the constraints on the parameters

$$A^{(\alpha\beta\gamma\delta ij)l} a_l = 0 \tag{3.17}$$

In the same manner as with the eigensymmetries we can find the solutions to the above equation and the linear combination of those describe the possible combinations of the parameters still fulfilling the constraints. The parameters which descibe this linear combinations of solutions we call the hyper parameters and the solutions we call the hyper vectors.

3.5 Mapping to super cells and force calculations

3.5.1 Map clusters to super cell

Once the orbits are found for the primitive cell the procedure of mapping them to a supercell is straightforward. First the supercell must be oriented in the same way as the original primitive cell. This can be done with e.g. the Kabsch algorithm and a suitable translation. The two lattices should now match and the cell metric of the supercell should be changed to the same as the primitive cell. Since both the atoms in the atom list of the supercell and the one belonging to the orbits uses the same site-offset representation clusters can be exactly mapped over. By looping over all the offsets of the atoms in the super cell the clusters can be translated to different parts of the lattice. Care must however be taken to use the boundary conditions of the original supercell. In the following sections the orbits of the supercell are assumed to be appropriate populated.

3.5.2 Calculate forces

Lets go through the procedure of calculating the forces stemming from a specific cluster. Suppose the cluster belong to some orbit and this orbit has a force constant Φ constructed by the linear combinations of the orbits eigentensors and parameters. The clusters in this orbit also have some multiplicity associated with them as we don't consider all the possible permutations of the clusters. E.g. cluster $[0, 1, 1, 3, 3, 3]$ would have a multiplicity of $6!/(1!2!3!) = 60$ but combined with the usual prefactor from the Taylor expansion $1/n!$ we just get

$$\texttt{prefactor} = \frac{1}{1!2!3!} \tag{3.18}$$

Next the cluster will be oriented in some way relative to the orbits prototype cluster. The relevant rotation is stored in the orientation family that the cluster belongs to. So to get the force constant of the orientation family we must rotate (contravariant) the the orbits force constant

$$\Phi_{\text{of}} = \Phi_{\text{orbit}}^{\alpha'\beta'\dots}(R^{-1})^{\alpha'\alpha}(R^{-1})^{\beta'\beta}\dots \tag{3.19}$$

Note that if the rotation matrices is in scaled coordinates they must first be transformed by $R_{\text{cart}} = V^T R_{\text{scaled}} V^{-T}$.
Since we store the clusters as ordered sequences it might happen that a permutation is needed to get the correct force constant for a cluster. E.g. $[1, 3]$ was mapped to $[4, 2]$ but we store it as $[2, 4]$. This is why to every cluster in the orientation family there is an associated permutation vector. If the cluster is transposed with this vector the correct force constant is retrieved.
One further speed up is possible by noting that if an atom appears more than once in the cluster we don't need to evaluate the clusters force for every appearance of the atom. Instead we just use the appropriate multiplicity of the atom in the cluster as an extra prefactor. E.g. for a on site cluster

$$F_0^\gamma = -\frac{1}{2}\frac{2!}{2!}\left(\Phi_{00}^{\gamma\beta}u_0^\beta + \Phi_{00}^{\alpha\gamma}u_0^\alpha\right) = -\frac{1}{2}\frac{2!}{2!}2\left(\Phi_{00}^{\gamma\beta}u_0^\beta\right) \tag{3.20}$$

So the minus sign is from $F = -\nabla E$, the $1/2$ is from the Taylor expansion, the $2!/2!$ is from the multiplicity of the orbit and the 2 is from the multiplicity of atom 0 in the cluster.

```python
for orbit in fcm.orbits:
    fc_orbit = orbit.fc
    prefac = orbit.prefac
    for of in orbit.orientation_families:
        rotation = of.rotation
        fc_of = rotate_fc(fc, rotation)
        for cluster in of:
            fc_cluster = fc_of.transpose(cluster.permutation)
            unique_atoms, unique_atom_indices, counts = np.unique(
    cluster)
            for a, i, c in zip(unique_atoms, unique_atom_indices,
    counts):
                tensor_list = [fc_cluster, list(range(orbit.order))]
                for j, atom in enumerate(cluster):
                    if j == i: continue
                    tensor.extend([displacements[a], [j]])
                forces[a] = - prefac * c * np.einsum(*tensor_list)
```

3.5.3 Fitting the parameters

Once the model is set up and able to calculate forces given displacements and a set of hyperparameters the procedure of fitting is straightforward. As demonstrated earlier the calculated forces will be a linear expression in the hyperparameters for a fixed displacement configuration. Suppose that the forces were to be calculated for some fixed displacement configuration and one hyperparameter, say parameter number i, set to one and all the other set to zero. The resulting forces can then be put in the ith column of an $m \times n$ matrix M where m is the number of force components and n is the number of hyperparameters. This matrix is called the fit matrix and repeating this procedure for all hyperparameters will build up the M matrix. Now the resulting forces for the given displacement and some vector of hyperparameters a will be given as the dot product of the fit matrix with the hyperparameter vector

$$F_{\text{predicted}} = Ma \tag{3.21}$$

Now if we were to be given displacements and forces from some source, e.g. dft code, we could fit the hyperparameters a by solving the equation

$$F_{\text{dft}} = M_{\text{dft}}a \tag{3.22}$$

with the fit matrix M calculated from the displacements of the dft data.
For small cutoffs and large systems the number of forces to fit against is large and the number of parameters is small so ordinary L2 methods are expected to work well. Should the number of parameters be large compared to the number of forces which would typically be the case when the cutoff is large and the input data is expensive to calculate as is often the case for ab initio codes, some kind of regularized L2 method could be needed.

```python
fcm = ForceConstantModel(atoms, cutoffs)
fcm.displacements = data.displacements
M = np.zeros((len(data.forces), number_of_hyperparameters))
for i in range(number_of_hyper_parameters):
    hyperparameters = [0] * number_of_hyper_parameters
    hyperparameters[i] = 1
    fcm.hyperparameters = hyperparameters
    M[:, i] = fcm.calc_forces()
fitted_hyperparameters = np.linalg.lstsq(M, data.forces)
```

4 Applications

In this chapter, I will consider two simple examples to demonstrate the extraction of force constant matrices and their subsequent analysis. First, the approach developed in this thesis is applied to obtain the fourth-order force constants of silicon as described by a widely used empirical potential. This illustrates the excellent accuracy that can be achieved in this fashion even at elevated temperatures.

In the second example the method is deployed to analyze phonon softening in tantalum, which showcases the extraction of finite temperature properties from molecular dynamics (MD) simulations based on an anharmonic force constant potential.

4.1 Anharmonicity in silicon

Silicon is widely used as a standard benchmark, see e.g., [21], and will be also considered here. It adopts the cubic diamond structure, belonging to space group Fd3m (ITC A no. 227), which tests the ability of the present implementation to handle asymmorphic space groups. While in actual applications one would commonly employ reference data from density functional theory (DFT) calculations, in the present case the atomic interactions are described using an empirical potential developed by Tersoff [28].

First, to generate configurations for training, MD simulations were carried out for $3\times3\times3$ supercells (216 atoms) at temperatures of 100, 300, 600, 900, and 1200 K. For each temperature one configuration each was extracted for training and validation, equivalent to $3 \times 216 = 648$ force components in the training and validation sets.

Both second-order (harmonic) and fourth-order models were generated using a cutoff range of 4.2,Å, located between the second and third nearest neighbor shell. This choice was motivated by the fact that for larger cutoffs the interaction strength was found to be very small. While the harmonic models comprised 6 independent parameters, the fourth-models contained 123 parameters.

The force constant potentials (FCPs) were fitted using a least-squares optimization algorithm to a subset comprising N_t force components that were randomly drawn from the training set and tested against the full validation set. This procedure was repeated 100 times for each value of N_t. While the training of the harmonic model converges for $N_t \approx 100$, the fourth-order model requires about 200 components for convergence (Fig. 4.1, left). Here, the relative error is defined as

$$\Delta = ||\boldsymbol{f}_p - \boldsymbol{f}_t||_2/||\boldsymbol{f}_t||_2, \tag{4.1}$$

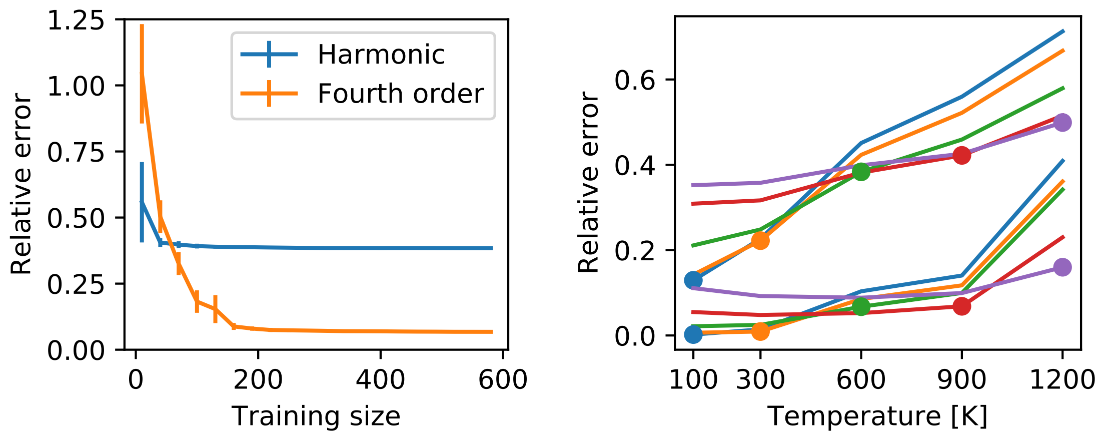


Figure 4.1: *(Left)* Convergence of fit with respect to the number of force components for the harmonic and fourth order model. *(Right)* The relative error of the prediction for the harmonic and fourth order model. The lower lines belong to the fourth order model. Dots indicate which temperature the parameters were fitted to. The lines describe the performance of the model away from its training region. Note how the lines crosses each other, indicating that the fits are of an effective nature.

where $\boldsymbol{f}_p$ and $\boldsymbol{f}_t$ are vectors of the predicted and target force components, respectively.
The training procedure was repeated using training data for each of the five temperatures indicated above and subsequently validated against data for each of the other temperatures (Fig. 4.1, right). From this analysis it is apparent that the second-order models are sensitive to the training data and thus, as expected, can only provide temperature dependent effective harmonic models. By contrast, fourth-order models trained at temperatures $\lesssim$ 900 K predict the forces observed at other temperatures with good accuracy. Only at the highest temperature of 1200 K, do the deviations become notable, which could be related to the training set being very small for a higher-order model and/or that terms beyond fourth-order becoming important.
In any case, the results are remarkable in so far as the underlying FCs were obtained using a single configuration for training. For comparison, computing the third-order FCs using a direct approach such as the one implemented in PHONOPY requires already six distinct calculations.
The good accuracy of the fourth-order models is also evident from a direct comparison of predicted and target forces (Fig. 4.2), where including higher-order terms yields a notable improvement compared to the second-order fits, especially at higher temperatures.
One of the big advantages of the present approach is that it allows us in principle to extract second (and higher) order force constants independently of the training data given. In Figure 4.3 the contributions and interaction strengths of different orbits are plotted. The second-order force constants do not change with temperature as they would do in an (effective) harmonic model. It is also noteworthy that the contributions derive mainly from nearest neighbor interactions and that even higher-

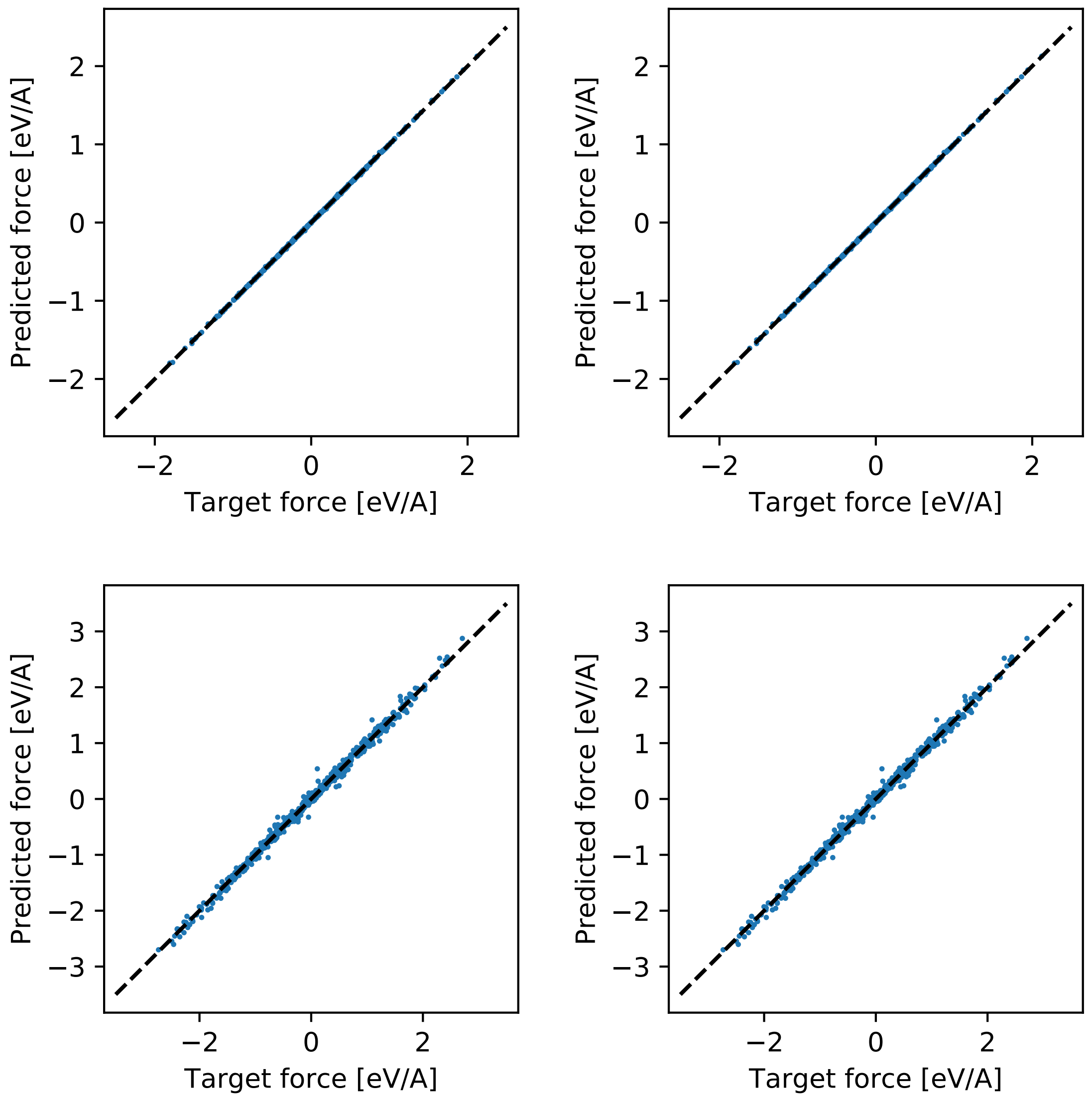


Figure 4.2: Comparison of harmonic (left) and fourth order (right) models. Upper row is 300K and lower is 900K.

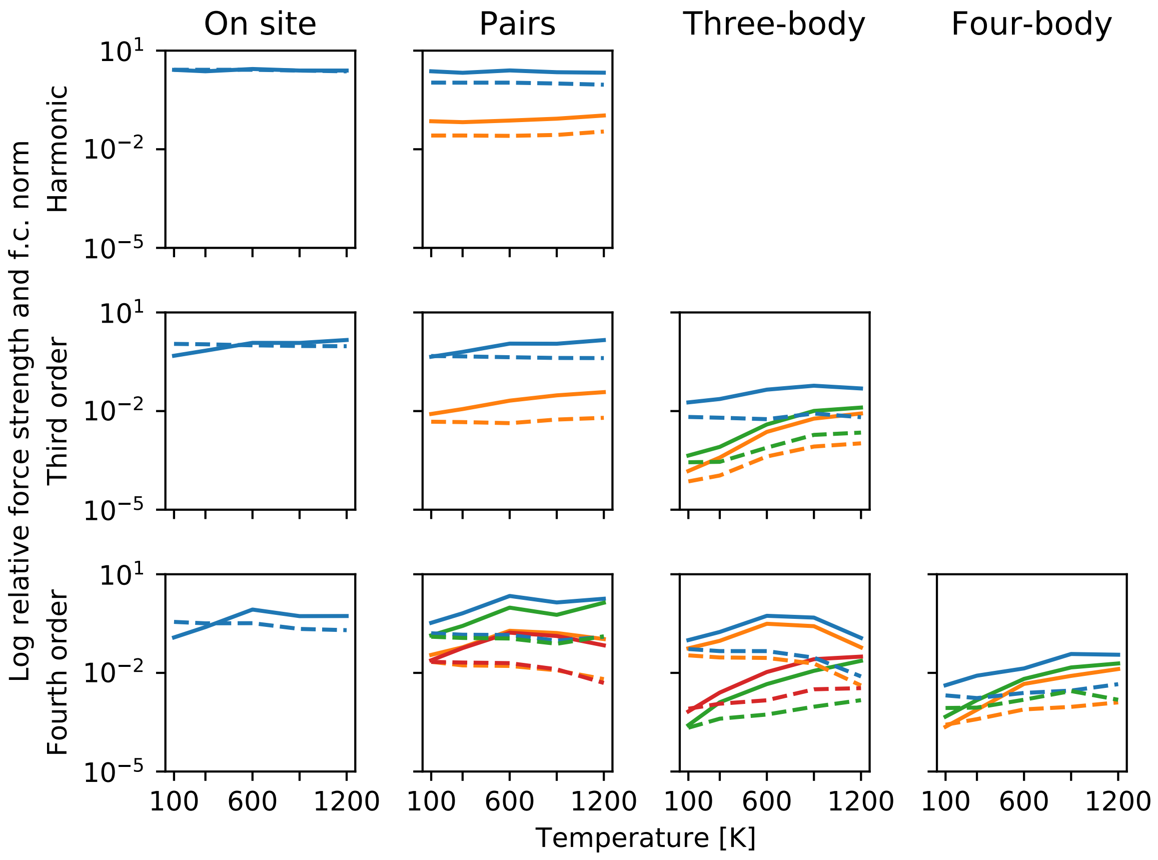


Figure 4.3: From left to right increasing many-body-interaction. From top to bottom increasing order. Solid lines represents normalized contribution to the total forces. Dashed lines represents normalized norm of the corresponding force constant.

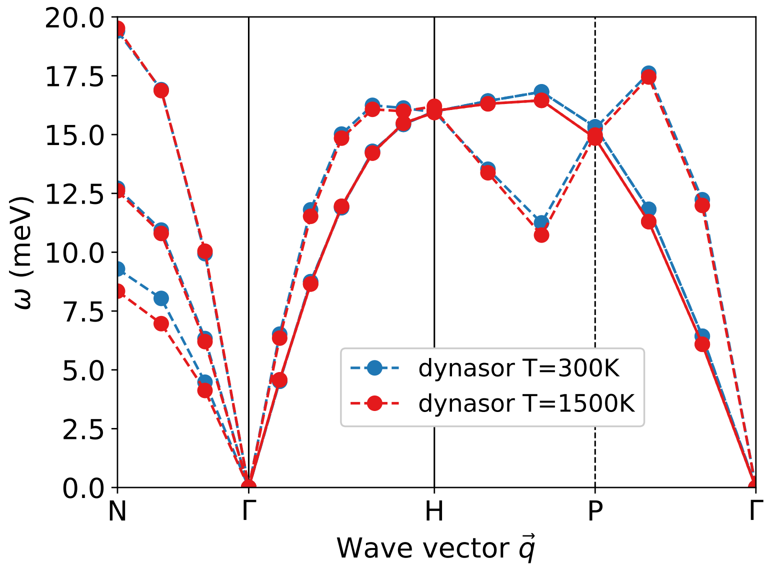


Figure 4.4: A demonstration of the phonon softening of Tantalum at the N point.

order contributions can be quite large for e.g. fourth order interactions among two atoms.

4.2 Phonon softening in tantalum

The vibrational properties of materials are very sensitive to temperature, which is crucial for understanding their thermodynamic properties. This behavior is evident from the variation of phonon frequencies and even more so life times with temperature. In this context, it will now be demonstrated how the approach and code developed in this work can be employed to extract the temperature dependence of the phonon frequencies by explicit simulation.
A fourth-order model was constructed for body-centered cubic (BCC) tantalum following a similar approach as in the case of silicon and using an embedded atom method potential [29]. The model was subsequently sampled using molecular dynamics (MD) simulations. The phonon frequencies and life times (not shown here) were extracted by recording velocity auto-correlation functions during the course of the MD simulation as implemented in the DYNASOR code.
The softening of the frequencies with increasing temperature is apparent in the thus obtained phonon dispersions, in particular at the N point (Fig. 4.4). Since this analysis was carried out at constant volume, the softening is purely the result of increased phonon-phonon coupling. If volume expansion was taken into account the softening would be even more pronounced.
This example serves as a simple illustration for the potential of the present approach (and code) for analyzing the vibrational features and properties derived thereof. This includes for example the extraction of free energy landscapes for of stable and metastable phases or the thermal conductivity.

5
Conclusions and outlook

In this thesis I have developed a general and efficient algorithm for obtaining higher order force constants for crystalline materials. The approach has been implemented in Python and can be easily interfaced with optimization algorithms available via external libraries such as scikit-learn and scipy.
The method has been extensively tested for stability and reliability considering a variety of different crystal structures, including both single and multi-component systems. While there is notable potential for further improvements in computational efficiency, already the current version allows one to deal with systems of low symmetry and large unit cells.
In this thesis, the performance of the method has been demonstrated and verified by applications to silicon and tantalum. Applications to various other systems can be anticipated including e.g., the study of vibrationally stabilized systems such as BCC-Ti, cubic zirconia, as well as high-temperature phases of ferroelectric, anti-ferroelectric, and multi-ferroic perovskites. Another area that deserves further exploration is the application to studies of thermal conduction, both in bulk materials, nanostructures, and across interfaces.
As suggested by the analysis of the FCs in silicon, the present approach promises to notably reduce the number of calculations required for extracting FCs, not only for homogeneous bulk systems but even more so for systems containing surfaces, interfaces, dislocations, or other defects as well as non-periodic configurations such as nanoparticles or wires. In order to realize this potential, further work is required to systematically establish the convergence behavior and develop optimal protocols for constructing training set structures.
These efforts will benefit from further performance enhancements pertaining in particular to the sampling of force constant models in MD simulations.

Bibliography

1. Dewandre, A. *et al.* Two-Step Phase Transition in SnSe and the Origins of its High Power Factor from First Principles. *Phys. Rev. Lett.* **117,** 276601 (2016).
2. Erhart, P. *et al.* Analytic bond-order potential for atomistic simulations of zinc oxide. *Journal of Physics: Condensed Matter* **18,** 6585 (2006).
3. Carlsson, A. in *Solid State Physics* 1–91 (Academic Press, New York, 1990).
4. Brockhouse, B. N. & Shull, C. G. *The Nobel Prize in Physics* 1994.
5. Raman, S. C. V. *The Nobel Prize in Physics* 1930.
6. Daw, M. S. & Baskes, M. I. Semiempirical, Quantum Mechanical Calculation of Hydrogen Embrittlement in Metals. *Phys. Rev. Lett.* **50,** 1285–1288 (1983).
7. Baskes, M. I. Modified embedded-atom potentials for cubic materials and impurities. *Phys. Rev. B* **46,** 2727–2742 (1992).
8. Stillinger, F. H. & Weber, T. A. Computer simulation of local order in condensed phases of silicon. *Phys. Rev. B* **31,** 5262–5271 (1985).
9. Bartók, A. P., Payne, M. C., Kondor, R. & Csányi, G. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **104,** 136403 (2010).
10. Behler, J. & Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **98,** 146401 (2007).
11. Thompson, A., Swiler, L., Trott, C., Foiles, S. & Tucker, G. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics* **285,** 316–330 (2015).
12. Togo, A., Chaput, L. & Tanaka, I. Distribution of phonon lifetime in Brillouin zone. *Phys. Rev. B* **91,** 094306 (2015).
13. Tadano, T. & Tsuneyuki, S. Self-consistent phonon calculations of lattice dynamical properties in cubic $SrTiO_3$ with first-principles anharmonic force constants. *Phys. Rev. B* **92,** 054301 (2015).
14. Thomas, J. C. & der Ven, A. V. Finite-temperature properties of strongly anharmonic and mechanically unstable crystal phases from first principles. *Phys. Rev. B* **88,** 214111 (2013).
15. Thomas, J. C. & der Ven, A. V. Elastic properties and stress-temperature phase diagrams of high-temperature phases with low-temperature lattice instabilities. *Phys. Rev. B* **90,** 224105 (2014).
16. Chen, M.-H., Thomas, J. C., Natarajan, A. R. & der Ven, A. V. Effects of strain on the stability of tetragonal ZrO_2. *Phys. Rev. B* **94,** 054108 (2016).
17. Kadkhodaei, S., Hong, Q.-J. & van de Walle, A. Free energy calculation of mechanically unstable but dynamically stabilized bcc titanium. *Phys. Rev. B* **95,** 064101 (2017).

18. Togo, A. & Tanaka, I. First principles phonon calculations in materials science. *Scripta Materialia* **108,** 1–5 (2015).
19. Andrade, X., Sanders, J. N. & Aspuru-Guzik, A. Application of compressed sensing to the simulation of atomic systems. *Proceedings of the National Academy of Sciences* **109,** 13928–13933 (2012).
20. Sanders, J. N., Andrade, X. & Aspuru-Guzik, A. Compressed Sensing for the Fast Computation of Matrices: Application to Molecular Vibrations. *ACS Cent. Sci.* **1,** 24–32 (2015).
21. Zhou, F., Nielson, W., Xia, Y. & Ozolins, V. Lattice anharmonicity and thermal conductivity from compressive sensing of first-principles calculations. *Phys. Rev. Lett.* **113,** 185501 (2014).
22. Souvatzis, P., Eriksson, O., Katsnelson, M. & Rudin, S. The self-consistent ab initio lattice dynamical method. *Computational Materials Science* **44,** 888–894 (2008).
23. Ziman, J. M. *Electrons and Phonons* (Oxford University Press, London, 1960).
24. Wallace, D. C. *Thermodynamics of Crystals* (Dover Publications, 1998).
25. Born, M. & Huang, K. *Dynamical Theory of Crystal Lattices* (Oxford University Press, 1998).
26. Togo, A. *spglib* 2009.
27. Larsen, A. H. *et al.* The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter* **29,** 273002 (2017).
28. Tersoff, J. Empirical Interatomic Potential for Carbon, with Applications to Amorphous Carbon. *Phys. Rev. Lett.* **61,** 2879–2882 (1988).
29. Ravelo, R., Germann, T. C., Guerrero, O., An, Q. & Holian, B. L. Shock-induced plasticity in tantalum single crystals: Interatomic potentials and large-scale molecular-dynamics simulations. *Phys. Rev. B* **88,** 134101 (2013).