MolDyCaT*

Report

September 21, 2012

*Molecular Dynamics Carbon Tight Binding

Abstract

The successful implementation of the carbon tight binding potential of Xu *et al.* [1] is reported. All of their results were faithfully reproduced and the model's applicability to graphene and carbon nanotube systems was investigated. Elastic constants and phonon frequencies for diamond were calculated to within 10% of experimental values. The diamond and graphite vacancy formation energies were calculated, with the graphite found to be 7.26 eV, a value falling within contemporary experimental ranges [2].

Acknowledgements

We would like to thank Peter Haynes, James Spencer and Tom Poole for their advice and support. Also Mike Finnis for informative conversations on interesting systems for study and Matthew Foulkes for advice on dynamical matrices. Lastly we would like to thank team `TAMA` for useful discussions during debugging (despite them stealing all of our `MolDy`snacks).

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Theory 1

1.2.1 Tight Binding 1

1.2.2 Hamiltonian Construction 3

1.2.3 Forces 3

1.2.4 Calculating Phonons 4

1.2.5 Geometry Optimisation 6

1.3 Non-orthorhombic unit cell 7

1.4 Thermostats 8

1.4.1 Velocity Scaling 8

1.4.2 Andersen 9

1.5 Integrators 9

2 Method **11**

2.1 System Architecture 11

2.1.1 Modular Design 11

2.1.2 Building the System 11

2.1.3 Custom Objects 11

2.1.4 Testing 13

2.1.5 Optimisation 14

2.1.6 Program Flow 14

2.2 Radial Distribution Function 15

3 Results **17**

3.1 Verification 17

3.1.1 Energy-Atomic Separation Curves 17

3.1.2 Elastic Constants 21

3.1.3 Phonons and Grüneisen Parameters 23

3.1.4 Carbon Clusters 24

3.1.5 Liquid Carbon 26

3.2 Further Studies 26

3.2.1 Optimisers 26

3.2.2 Nanotubes 27

3.2.3 Point Defects 27

3.2.4 MD simulations 30

3.2.5 Bulk Modulus 32

4 Project Management **33**

4.1 Approach 33

4.2 Tools 33

5 Conclusion **35**

A User Documentation **36**

A.1 Installation Requirements 36

A.2 Running `MolDyCaT` 36

A.3 Input Files 37

A.4 Output Files 43

Chapter 1

Introduction

1.1 Introduction

The students on the 2011-2012 MSc course of the TSM CDT[1] at Imperial College London were given the challenge of reproducing the results of Xu *et al.* [1] who built a code implementing their empirical tight binding model of carbon. The students split into two teams and thus was born `MolDyCaT`.

With the characterisation of new carbon structures since the publication of the original paper, carbon nanotubes in 1991 [3] and graphene in 2007 [4], tight binding models of carbon are as relevant as ever. It was therefore decided to use `MolDyCaT` to test how well the model's parameterisation transferred to these recently discovered structures.

A full molecular dynamics code was written around the forces calculated from the tight binding model. This allowed the temporal evolution of carbon systems to be investigated and optimised carbon structures to be computed.

1.2 Theory

1.2.1 Tight Binding

The semi-empirical tight binding model implemented captures the quantum mechanical essence of bonding without recourse to the explicit electronic wavefunction. This results in an accurate picture of carbon bonding, in which quantum effects are significant, but without the computational expense of full *ab initio* methods. `MolDyCaT` calculations of several hundreds atoms can be performed on an ordinary laptop computer.

Tight binding uses a minimal basis of local orbitals which is both incomplete and non-orthogonal.

[1] Theory and Simulation of Materials Centre for Doctoral Training

Linear combinations of s and p orbitals centred on each atomic site result in hybridised sp^3 bonding between atoms. Assuming the interactions between orbitals to be short ranged ensures that such models have transferability between a range of systems [5] and allows the orbitals to be approximated as orthogonal. This corresponds to an overlap matrix between atomic orbitals, equal to the identity [5].

The effects of three centre integrals were neglected in this model, resulting in a parameterised solution to the Schrödinger equation which is written only as a function of the separation of atomic sites. The inaccuracies arising from excluding the three centre integrals were noted by Foulkes [6] and are compensated for in the parameterisation of the model.

The total energy is the sum of the tight binding energy and the repulsive energy,

$$E_{\text{total}} = E_{\text{rep}} + E_{\text{TB}}\,, \tag{1.1}$$

where E_{TB} is the sum of the energy eigenvalues of all the occupied electronic states. The Hamiltonian is constructed by considering the effective interaction of each orbital pair, scaled as a function of distance, $s(r)$, modified by a factor to account for relative angular orientations. The eigenvalues are then calculated by diagonalisation of the Hamiltonian.

E_{rep} is the short range repulsive energy, comprising of a sum over a pairwise potential between atoms. This tries to capture the effects of the Hartree energy (the mean field approximation of the coulombic repulsion of the electrons), exchange correlation energy and nuclei-nuclei repulsion:

$$E_{\text{rep}} = \sum_i f\left[\sum_{j\neq i}\phi(r_{ij})\right]\,, \tag{1.2}$$

where $\phi(r_{ij})$ is a GSP type scaling function, that is a function of the separation of atoms i and j and $f\,[x]$ is a fourth order functional with argument $x = \sum_{j\neq i}\phi(r_{ij})$. For the parameterisation please see the Code Specification [7].

The scaling functions, $\phi(r)$ and $s(r)$, chosen by Xu *et al.*, have the functional form suggested by Goodwin *et al.* [8],

$$f_{GSP}(r) = \frac{R_0}{r}^n \exp\left\{n\left[-\left(\frac{r}{R_c}\right)^{n_c} + \left(\frac{R_0}{R_c}\,,\right)^{n_c}\right]\right\}, \tag{1.3}$$

where f_{GSP} may be either $\phi(r)$ or $s(r)$, R_0 is the nearest neighbour separation and R_c is used as the matching point between the GSP function and the tail function. The constants n, n_c, R_0, R_c are specific to $\phi(r)$ and $s(r)$ and are listed in the Code Specification [7]. This form has been shown to be successful [5], as the exponential smooths the inverse power law that previous work had shown to be successful [9]. This is then coupled with a tail function to bring the GSP function and its gradient to zero smoothly.

Xu *et al.* fitted versus nearest neighbour separations for carbon graphite, diamond, linear chains, simple cubic and FCC (face-centred cubic) to local density approximation DFT calculations to parameterise the model [1].

1.2.2 Hamiltonian Construction

The Hamiltonian matrix H_{iajb} was constructed in 4×4 blocks, each block modelling the interactions between the s and p orbitals of atoms i and atom j:

$$H_{i=j} = \begin{array}{c} \\ b\downarrow \end{array} \begin{array}{c} a\rightarrow \\ \begin{pmatrix} E_s & 0 & 0 & 0 \\ 0 & E_p & 0 & 0 \\ 0 & 0 & E_p & 0 \\ 0 & 0 & 0 & E_p \end{pmatrix} \end{array}$$

$$H_{i\neq j} = \begin{array}{c} \\ b\downarrow \end{array} \begin{array}{c} a\rightarrow \\ \begin{pmatrix} V_{ss\sigma} & lV_{sp\sigma} & mV_{sp\sigma} & nV_{sp\sigma} \\ -lV_{sp\sigma} & l^2V_{pp\sigma} + (1-l^2)V_{pp\pi} & lm(V_{pp\sigma} - V_{pp\pi}) & ln(V_{pp\sigma} - V_{pp\pi}) \\ -mV_{sp\sigma} & lm(V_{pp\sigma} - V_{pp\pi}) & m^2V_{pp\sigma} + (1-m^2)V_{pp\pi} & mn(V_{pp\sigma} - V_{pp\pi}) \\ -nV_{sp\sigma} & ln(V_{pp\sigma} - V_{pp\pi}) & mn(V_{pp\sigma} - V_{pp\pi}) & n^2V_{pp\sigma} + (1-n^2)V_{pp\pi} \end{pmatrix} \end{array}$$

where l, m and n are the directional cosines between atom j and i, i.e. $\mathbf{r}_{ij} = |r|\begin{pmatrix} l \\ m \\ n \end{pmatrix}$. All these elements were then multiplied by the scaling function $s(r)$:

$$s(r) = \begin{cases} \left(\frac{r_0}{r}\right)^n \exp\left\{n\left[-\left(\frac{r}{r_c}\right)^{n_c} + \left(\frac{r_0}{r_c}\right)^{n_c}\right]\right\} & \text{if } r \leq r_1 \\ \sum_{n=1}^{4} c_n (r - r_1)^n & \text{if } r_1 < r < r_m \\ 0 & \text{if } r \geq r_m \end{cases}, \tag{1.4}$$

where r_m is the cut-off distance of 2.6 Å.

1.2.3 Forces

Repulsive Forces

The repulsive forces were calculated by differentiating equation 1.2,

$$F_i = \frac{\partial f\,[x]}{\partial x} \sum_{j \neq i} \frac{d\phi_{ij}(r_{ij})}{dr_{ij}} \hat{\mathbf{r}}_{ij}, \tag{1.5}$$

where F_i is the repulsive force on atom i, $\hat{\mathbf{r}}_{ij}$ is the unit vector between atoms i and j, r_{ij} is the distance between atoms i and j, $\phi(r_{ij})$ is the scaling function and $f[x]$ is a fourth order polynomial in $x = \sum_{j \neq \alpha} \phi(r_{\alpha j})$.

Hellman-Feynmann Forces

Assuming an orthogonal basis set, the α component of the Hellman-Feynmann force on atom i is given by:

$$F_{i\alpha} = -2 \sum_{ajb, j \neq i} \rho_{iajb} \frac{\partial H_{jbia}}{\partial R_{i\alpha}} \,. \tag{1.6}$$

where ρ_{iajb} is the density matrix:

$$\rho_{iajb} = \sum_s n_s c_{ia}^{(s)*} c_{jb}^{(s)} \,, \tag{1.7}$$

where s is a sum over the occupied states with occupation given by the vector n_s, and $c^{(s)}$ represents the eigenvector of state s. The idempotency constraint $\rho^2 = \rho$ proved a useful test during code development.

Degeneracy around the Fermi level (highest occupied state) was taken into account in the construction of the density matrix. In the absence of degeneracy at the Fermi level, $n_s = 1$ for $s \leq 2N$, where N is the number of atoms. If instead, there were levels with energy equal to the Fermi energy (to within a tiny tolerance), they were considered degenerate with the Fermi level. In this case, n_s among those levels was set equal to a value such that $\sum_s n_s = 2N$.

To simplify the code, a single "Hamiltonian derivative" matrix with respect to each Cartesian component was constructed, where the i^{th} block (of 4 rows) was the derivative with respect to r_i of the corresponding block in the original Hamiltonian. This implementation meant the force evaluation for atom i involved multiplications only with elements in the i^{th} block of the "Hamiltonian derivative". This saves computational time over an approach using a full matrix multiplication.

Both the repulsive and Hellman-Feynman forces were compared to forward difference numerical derivates and found to be in excellent agreement.

1.2.4 Calculating Phonons

Dynamical Matrix

An attempt was made to perform phonon calculations by constructing the full dynamical matrix for phonons of wavevector $\mathbf{k}$ [10]:

$$\mathbf{D}(\mathbf{k};\mu,\nu) = \frac{1}{M}\sum_{\mathbf{m}} \Phi\left(0,\mu;\mathbf{m},\nu\right) \exp\left\{-2\pi i \mathbf{k}\cdot\left[\mathbf{R}\left(0,\mu\right) - \mathbf{R}\left(\mathbf{m},\nu\right)\right]\right\} , \tag{1.8}$$

where μ and ν are atomic indices, M is the mass of carbon, $\Phi\left(0,\mu;\mathbf{m},\nu\right)$ is the force constant on atom μ due to atom ν in unit cell $\mathbf{m}$. The force constants were calculated by perturbing each atom in term along each of the three Cartesian directions, and taking a forward difference of the forces on each atom. Each force component was considered separately, making the dynamical matrix size $3N \times 3N$ for a system of N atoms.

This dynamical matrix was then diagonalised to compute the eigenvalues (spectrum of squared phonon frequencies) and normal mode eigenvectors (amplitudes and phases of each atom in a given normal mode).

Unfortunately implementation of this method in `MolDyCaT` was unable to reproduce the results of Xu *et al.* [1] within a reasonable accuracy. The frequency spectrum had the same number of degenerate frequencies but with optical mode frequencies several times larger than expected. Despite this discrepancy the dynamical matrices satisfied the necessary properties:

- Hermitian property $\mathbf{D}^{\dagger} = \mathbf{D}$.
- The frequency spectrum contained only positive values and recovered degenerate zero frequency modes at the Γ point.
- The acoustic sum rule was satisfied (diagonal elements are equal to the negative sum of the other elements of its row).

The main problem was that the computational expense of this calculation allowed only very small systems to be considered so it was not possible to converge the frequencies with respect to system size. With more time the code could be adapted to exploit the symmetries of the system to reduce the number of force evaluations and thereby bring the required system sizes into the *pur*view of `MolDyCaT`. Other problems may have existed within the code which will have become more obvious if this speed up was achieved. Code was written to calculate the phonon density of states $g(\omega)$

$$g(\omega) = \sum_{s}\int \frac{d\mathbf{k}}{(2\pi)^3}\delta(\omega - \omega_s(\mathbf{k})) \tag{1.9}$$

where s denotes a sum over the branches of the phonon spectrum and the integration is performed over all phonon wave vectors $\mathbf{k}$. However, without a reliable phonon spectrum this feature could not be used.

Instead of using the dynamical matrix it was decided to calculate the frequencies using the frozen-phonon super-cell approximation.

1.2.5 Geometry Optimisation

In order to perform structural relaxation on carbon structures it was necessary to implement a geometry optimisation algorithm. Schemes which worked robustly and efficiently for clusters were found to be unstable when applied to periodic bulk structures so a range of different optimisers were implemented around the `optbase` template.

Steepest descents (SD) was found to be stable but slow to converge, particularly for large systems unless the step was updated manually. Adaptive step size variations were found to be unsuited to this model.

By maintaining force information from previous iterations, the SD algorithm was evolved into a conjugate gradients (CG) scheme. This greatly accelerated convergence but at the expense of stability and this scheme was unable to cope with interstitials in diamond lattices.

An alternative approach based on damped molecular dynamics [11] (DMD) was implemented. Based on the assumption of a harmonic potential, the relaxation proceeds as a standard molecular dynamics simulation but with an extra damping force proportional to the velocity $F_{\text{damping}} = -\gamma v$. In the coupled modes (DMD-CM) approach, each degree of freedom is damped uniformly with a damping constant calculated from the energies of the first 3 steps, computed using SD, $E^{(i)}$, $i = 1, 2, 3$ and the time step Δt:

$$\gamma^{\text{CM}} = \frac{1}{\Delta t}\sqrt{\frac{1}{2}\ln\left(\frac{E^{(1)} - E^{(2)}}{E^{(2)} - E^{(3)}}\right)}\,. \tag{1.10}$$

The uncoupled modes approach (DMD-UM) of Probert [12] was also implemented. In this algorithm each atomic degree of freedom is treated independently and a separate damping coefficient is computed for each atom in each Cartesian direction. The damping coefficient of the α component of atom i is then given by:

$$\gamma_{i,\alpha}^{\text{UM}} = \frac{1}{\Delta t}\sqrt{\frac{F_{i,\alpha}^{(1)}}{F_{i,\alpha}^{(2)}}}\,. \tag{1.11}$$

In both the coupled and uncoupled methods the system was integrated using a velocity Verlet scheme once the damping forces were computed. The application of damping necessitated the use of an extra half step during integration. The time step of integration integration was chosen as $2\pi/N\gamma$ where $\gamma = \min\{\gamma_{i,\alpha}\}$ and $N = 8$. The assumption that all modes may be treated as independent resulted in faster convergence for some systems but caused stability issues for other systems. It was found that recomputing the damping constant during the simulation could further accelerate convergence without severely affecting stability.

1.3 Non-orthorhombic unit cell

Some systems are more naturally represented in terms of their primitive unit cells rather than conventional unit cells. Calculating shears is also more natural in a non-orthogonal basis as the lattice angles can simply be varied instead of having to derive complicated expressions for strain tensors. It was therefore decided to incorporate arbitrary non-orthorhombic unit cells into the functionality of `MolDyCaT`.

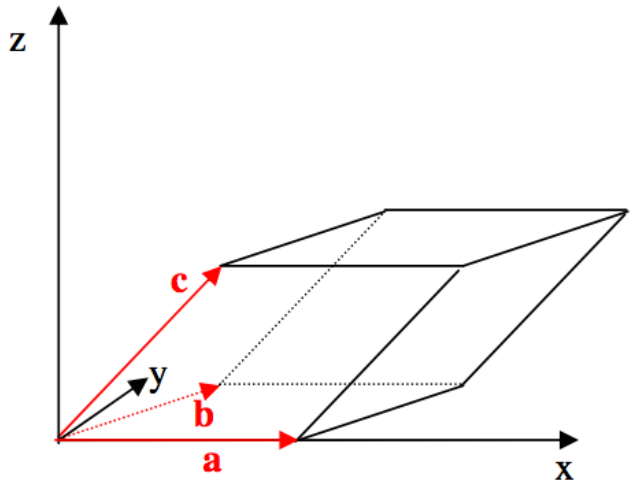


Figure 1.1: Non-orthorhombic unit cell. [13]

Internally, all coordinates are stored in the non-orthogonal basis $\mathcal{S}'$ specified in the user input file. Vector arithmetic involving periodic images therefore required a transformation into an orthogonal basis $\mathcal{S}$. This was achieved using a transformation matrix $\mathbf{T}$ precomputed at the initialisation of the simulation. The inverse $\mathbf{T}^{-1}$ is computed simultaneously for coordinate transformations $\mathcal{S} \to \mathcal{S}'$. To simplify the expression of this transformation, the systems were assumed to be aligned as in figure 1.1 such that $\mathbf{T}$ takes the form:

$$\mathbf{T} = \begin{pmatrix} 1 & \cos\gamma & \cos\beta \\ 0 & A & B \\ 0 & 0 & \pm\sqrt{(1-A-B)} \end{pmatrix}, \tag{1.12}$$

$$A = \pm\sqrt{(1-\cos^2\gamma)}\,, \tag{1.13}$$

$$B = \frac{\cos\alpha - \cos\gamma\cos\beta}{A}\,. \tag{1.14}$$

When adding vectors in a boundary corrected fashion, the result is first transformed into $\mathcal{S}$ where the correction may be applied straightforwardly. From here the result can be transformed back into the non-orthorhombic basis $\mathcal{S}'$ if required. This is currently only implemented for the energy calculation,

but could easily be extended to forces. The force calculation is currently performed with respect to energy derivatives along the orthorhombic basis. These forces would need to be resolved back into $\mathcal{S}'$ for integration.

1.4 Thermostats

Physical systems under experimental conditions are typically in contact with an external heat bath with which they exchange energy. Microcanonical NVE simulations, which necessarily conserve energy, are not suited to such systems as the temperature fluctuates with system size as $\Delta T \sim 1/\sqrt{N}$. For system sizes practically accessible to MD simulation, ΔT is too large to be ignored. This problem is addressed with thermostats which modify the NVE dynamics to ensure that the temperature at each time step falls within a specified tolerance.

1.4.1 Velocity Scaling

The simplest approach to thermostats is based around a simple rescaling of velocities [14]. The instantaneous temperature $T(t)$ of the system is defined to be:

$$T = \frac{1}{2}\sum_i \frac{M v_i^2}{Q k_B}\,, \tag{1.15}$$

where M is the mass of each atom, v_i is the velocity of atom i and Q is the number of degrees of freedom. As there are no external forces acting on the system, total linear momentum is conserved reducing Q to $3N-3$. In order to achieve the target temperature T_0, the velocities must be rescaled by a factor ξ such that

$$\Delta T = (\xi^2 - 1)T(t)\,, \tag{1.16}$$

$$\xi = \sqrt{(T_0/T(t))}\,. \tag{1.17}$$

This technique is straightforward to implement but does not remove local correlations in the system, a feature characteristic of thermal motion. This left velocity scaling unsuitable for melting periodic carbon structures as linear chains and clusters were not readily broken up. This also causes the dynamics of the velocity scaling thermostat to deviate from the canonical ensemble, introducing systematic errors into averaged quantities.

1.4.2 Andersen

The Andersen thermostat [15] breaks the local correlated motion by replacing the integrated velocities of atoms with values drawn from a Maxwell-Boltzmann distribution at the desired temperature. A configurable parameter λ specifies the strength of the thermostat, i.e. what fraction of atomic velocities are modified. As $\lambda \to 0$, NVE dynamics are recovered while a value of $\lambda = 1$ forces all velocities to be replaced. Each velocity component being modified is drawn from a zero centred Gaussian of variance $k_B T/M$.

1.5 Integrators

The workhorse of most MD simulations is the velocity Verlet integration scheme. This is relatively efficient, achieving $\mathcal{O}(\Delta t^2)$ accuracy using a single force evaluation, $\mathbf{F}$, per integration step. The update procedure on each atom is:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + \frac{1}{2M}\mathbf{F}_n \Delta t^2 \,, \tag{1.18}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{1}{2M}(\mathbf{F}_n + \mathbf{F}_{n+1})\Delta t^2 \,. \tag{1.19}$$

The Verlet scheme is the lowest order method of the more general family of symplectic partitioned Runge-Kutta methods. They all share the self-starting, time-reversal and symplectic properties which make Verlet attractive to MD but exist as higher order variants [16]. Each integration step is performed in several stages in which the positions and momenta are updated independently, based on their values at the previous stage. Each higher order method is constructed by combining several such stages. A typical integration step of an s-stage integrator looks as follows:

for $k = 1 \ldots s$,

$$\mathbf{v}_k = \mathbf{v}_{k-1} + \frac{b_k}{M}\mathbf{F}(\mathbf{r}_{i-1})\Delta t \,, \tag{1.20}$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} + a_k \mathbf{v}_k \Delta t \,, \tag{1.21}$$

end.

where a_k and b_k are coefficients defining the particular method being used. The `si2a`, `si3a`, `si3b`, `si4a`, `si4b`, `si4c` and `si6a` schemes described by Gray *et al.* [17] were implemented.

It was found that the order of the integrator could at times be a misleading indicator of performance as the absolute error was often very high for typical MD timestep sizes. Although the errors did scale

favourably with timestep, it was only for very small time steps (typically far lower than those required for MD), that higher order integrators begin to pay dividends. As most practical simulations require NVT ensembles where the dynamics are modified by the thermostats the higher order integration schemes proved to be of limited use and Verlet was used as the default integrator. Tests performed on Lennard-Jones systems suggested that these would benefit considerably from higher order integrators as much larger time steps could be used without introducing significant errors.

Chapter 2

Method

2.1 System Architecture

2.1.1 Modular Design

The `MolDyCaT` program was made with a modular design, as laid out in the Code Specification [7]. This modular design was chosen to allow testing of individual components rather than having to test and debug the program as a whole.

Modular design allows individual components to be interchanged without the need restructure other parts. This reduced complexity and interdependence was useful during testing as we could begin developing the MD code using simple forces derived from simple potentials, agnostic of the forces from the tight binding model. As soon as the tight binding forces were functioning correctly, only a single line of code needed to be changed for `MolDyCaT` to use the tight binding forces instead.

2.1.2 Building the System

The CMake[1] build system was chosen to produce the make file which links all the modules at compile time. It was chosen for cross-platform compatibility, detection of required libraries at compile time and simplicity.

2.1.3 Custom Objects

To facilitate code reuse and allow the `C++` code to follow the mathematical expressions as closely as possible, an object orientated approach was used. The following classes were written at the beginning of the development stage.

[1] `http://www.cmake.org/`

The `vect3` and `atom` Classes

The `vect3` object was designed as a wrapper around an Armadillo[2] vector of fixed length. This design allowed reuse of Armadillo's arithmetic operations out of the box and gave custom `MolDyCat` classes access to the Armadillo expression template engine (allowing compound arithmetic operations to be evaluated efficiently).

To allow the usage of periodic boundary conditions, the `vect3` class included the possibility of calling boundary corrected addition and difference operators to facilitate operations for systems with arbitrary unit cells.

Throughout the program a *const correct* design approach was used to prevent disparate modules from unexpectedly modifying those simulations' states. Enforcing const correctness across the program was one method of preventing bugs being transferred across modules. This also reduced debugging time as many potential problems were caught at compile time rather than as nonsensical output at runtime.

The `atom` object was built around two `vect3` objects to store the position and velocity of an atom.

Corresponding list types were implemented for the `atom` and `vect3` types around Standard Template Library (STL) vectors to store arrays of defined lengths of atom coordinates and positions.

The `unitcell` Class

The `unitcell` objects store the simulation box lengths, the angles between the lattice vectors and the boundary conditions. Boundary corrected addition, difference and inner products for vectors in non-orthorhombic unit cells are handled in this class.

The `unitcell` objects are created on the initialisation of the simulation, at which point, matrices for the forward and backward transformation between the general non-orthorhombic unit cell specified in the input file and a fictitious orthorhombic unit cell are generated [18]. Before any vector operation, the operands are transformed into the fictitious orthorhombic unit cell, and the operation is performed with boundary corrections. If required, the result is then transformed back into the non-orthorhombic unit cell.

The `simulation` Class

At the start of the program the input file is parsed and used to generate a `simulation` object – a container to store the simulation parameters as constants. This prevents any accidental changes to user specified parameters after simulation initialisation and simplified code because long lists of

[2] `http://arma.sourceforge.net/`

arguments in function calls could be replaced by a single object. Allowing simulation parameters to be configured in input files facilitates the changing of simulations parameters without having to recompile. Default values for most parameters are used if they are not specified in the input file.

The `forceeval` Class

The `forceeval` class was used to allow easy access to modules B through to E (see Code Specification [7] for details). The details of the potential model were abstracted away allowing energies and forces of arbitrary atomic configurations to be calculated with a single line of code. This allowed easy recycling of Hamiltonian matrices and derived objects to prevent computationally expensive and unnecessary memory allocations.

One benefit of this approach is the ease with which `MolDyCaT` could be modified to use other potentials or force-fields. This proved useful when testing the molecular dynamics functionality on harmonic and Lennard-Jones potentials as it could be done independently of the tight binding modules.

The `stepperbase`, `thermobase` and `optbase` Classes

To equip `MolDyCaT` with the tools required to simulate a variety of different systems[3] flexibility was built into the design from the start. It was not known which thermostats, integrators or relaxation algorithms would prove most effective in advance so a framework allowing for the addition of algorithms to the existing code base was designed. Virtual base types were defined for integrators, thermostats and relaxation algorithms in the classes `stepperbase`, `thermobase` and `optbase`, respectively. The remaining code could work entirely irrespective of which particular base type was selected. At run time, the base type was initialised to the derived type specified and the associated code run. This enabled code reuse and allowed for quick implementation of further integrators, thermostats and relaxation algorithms. The integrator class was templated so that new symplectic integrators could be entirely specified by weights in the integration stages, for example the leap frog integrator could be defined by just two numbers in this scheme.

2.1.4 Testing

All modules were tested independently and `MolDyCaT` was found to have passed all module tests outlined in the code specification [7]. Lennard Jones and simple harmonic potentials, were used to test the molecular dynamics. Toy models were used to test the tight binding and repulsive energies and forces. Small two atom systems could be worked out by hand and larger four atoms systems

[3] Periodic 2D and 3D crystals, small clusters, linear chains and liquid carbon.

using Mathematica[4].

2.1.5 Optimisation

The code was profiled using Valgrind[5] to identify potential performance bottlenecks. The predominant computational expense was the $\mathcal{O}(N^3)$ diagonalisation of the Hamiltonian matrix. This was handled by an external library and could therefore not be optimised further. Other slow performing loops were identified and parallelised using OpenMP. This was particularly important for the density matrix calculation ($\mathcal{O}(N^3)$) and the construction of the dynamical matrix ($\mathcal{O}(N^4)$). The latter was further optimised by running several copies of the `forceeval` object in parallel to handle the force constant calculations of each atom independently. In the original code specification [7], an $\mathcal{O}(N)$ variant of the model was mentioned. This was considered unnecessary for system sizes of interest as the linear scaling overhead outweighs potential performance benefits (the crossover point between $\mathcal{O}(N)$ and $\mathcal{O}(N^3)$ is at around 230 atoms for liquid carbon [19]). Although the $\mathcal{O}(N^3)$ model implemented was suitable for systems of interest, the modular design makes it possible to add this functionality at a later stage without any code restructuring. This was the motivation behind explicitly building and working in terms of the density matrix, as this is the quantity determined by Li, Nunes and Vanderbilt's [20] linear scaling techniques.

2.1.6 Program Flow

The program is executed from the command line with a single argument to specify the seed name, for example: `./moldycat {seed}` (see user documentation in Appendix A).

The input file associated with the seed, `inputs/{seed}_params.in`, if successfully parsed, creates the `simulation` object to store all configurable simulation parameters. Some validation is carried out to catch spurious input values which could cause unphysical simulations or cause the program to crash. When possible a line number on which the input file parsing error occurred along with any additional information is output to the user.

If the restart flag is enabled, the end positions and velocities of any previous simulations with the same seed are parsed and used to initialise the atom list.

The type of simulation, energy or molecular dynamics, determines the subsequent flow of the program:

- If a structural relaxation is requested first, an `optbase` object is initialised to the correct derived type. This is then iterated until the maximum number of iterations is reached or the forces are

[4] `http://www.wolfram.com/mathematica/`
[5] `http://valgrind.org/`

converged within the specified tolerance. Supported relaxation algorithms are steepest descent, conjugate gradient [21] and coupled and uncoupled modes damped molecular dynamics [12].

- If the simulation task is set to energy, the forces are not calculated, thus saving on computational time. A `forceeval` object is initialised and the Hamiltonian created. The Hamiltonian is then diagonalised and the total energy is returned. A mixture of stand-alone executables and scripts based around the energy calculations were written to produce the energy per atom versus nearest neighbour separation curves and elastic constants.

- If the simulation task is molecular dynamics, a `forceeval` object is initialised which is used to compute energies and forces of a configuration of atoms. Internally, the `forceeval` object initialises the Hamiltonian and its derivatives with respect to the x, y and z components as Armadillo matrices. The Hamiltonian is then diagonalised using the Armadillo library (which interfaces with LAPACK[6]). The density matrix is constructed from the eigenvectors to evaluate the Hellman-Feynman contribution to the forces.

For molecular dynamics runs, the `forceeval` object is passed to the integrator which propagates the atoms to the next time step. The implemented integrators are Verlet and a family of higher order symplectic integrators [17].

If requested by the user in the input file, a thermostat (velocity rescaling or Andersen), is then applied to the atomic velocities. The new forces are used to populate a new Hamiltonian and the process repeats until the specified number of MD steps is reached. If specified, the positions are sampled to calculate the radial distribution function.

2.2 Radial Distribution Function

The following method was employed to calculate the radial distribution function (RDF) using the variables `RDF_BINS` and `RDF_CUTOFF` specified in the input file:

- A number of histogram bins specified by `RDF_BINS` are created with each bin corresponding to a segment of width `RDF_CUTOFF/RDF_BINS` along the range from 0 to `RDF_CUTOFF`.

- At every `RDF_STEPS` number of steps the distance between all pairs of atoms is calculated and, if below `RDF_CUTOFF`, the value of the appropriate histogram bin is incremented by 2.

- At the end of the simulation the value in each bin is normalised according to equation 2.1, where $n(R)$ is the unnormalised bin value for the bin at R, R is the mid-value of the length

[6] `http://www.netlib.org/lapack/`

range to which the bin corresponds, dR is the bin width, ρ is the density and N is the total number of particles.

- The normalised bin values and the corresponding value of R are then printed to the file `outputs/(seed_name)_rdf.dat`.

$$g(R) = \frac{1}{\texttt{RDF\_STEPS}} \frac{n(R)}{4\pi R^2 dR \rho N} \,. \tag{2.1}$$

To enable the RDF to be calculated up to distances greater than half the box length the `supercell` method was used to create 26 image cells around the simulation cell. These extra cells were used only for calculation of the RDF and not for MD.

Chapter 3

Results

3.1 Verification

The initial goal for `MolDyCaT` was to reproduce the results of Xu *et al.* [1], as laid out in the Code Specification [7, Ch 4].

3.1.1 Energy-Atomic Separation Curves

Firstly, this meant reproducing the energy per atom versus atomic separation curves for a variety of structures including a linear chain of carbon atoms, diamond, graphene, β-Sn, FCC, BCC and HCP. The first three are shown in figure 3.3 and all are shown together in figure 3.4.

To mitigate the artefacts created by finite size effects in study of small systems, all physical properties were converged with respect to system size. As the number of atoms used for a single point energy calculation was increased the energy converged to a limiting value. The convergence for the (3,3) nanotube in shown in figure 3.1. The energy converges sharply at 50 atoms, which was found to be a typical value for the systems studied.

The minimum energy and equilibrium bond length for each structure (represented by their coordination number) are presented in figure 3.2.

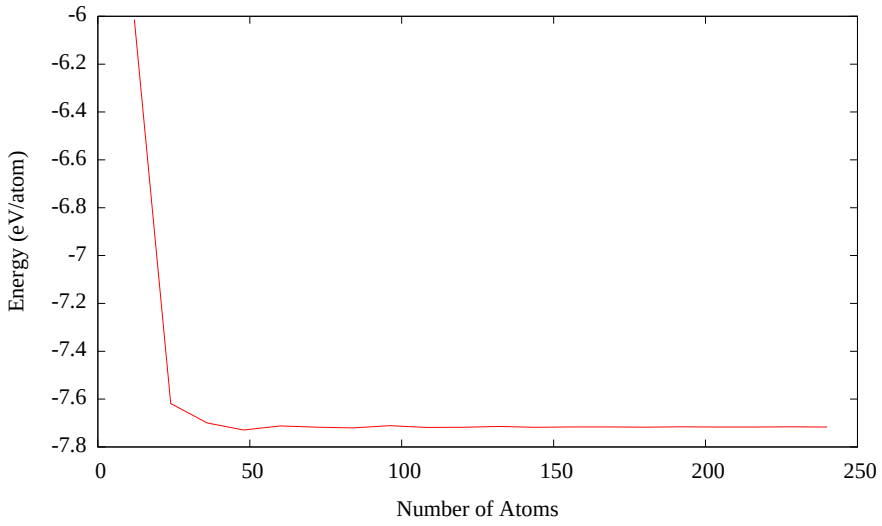
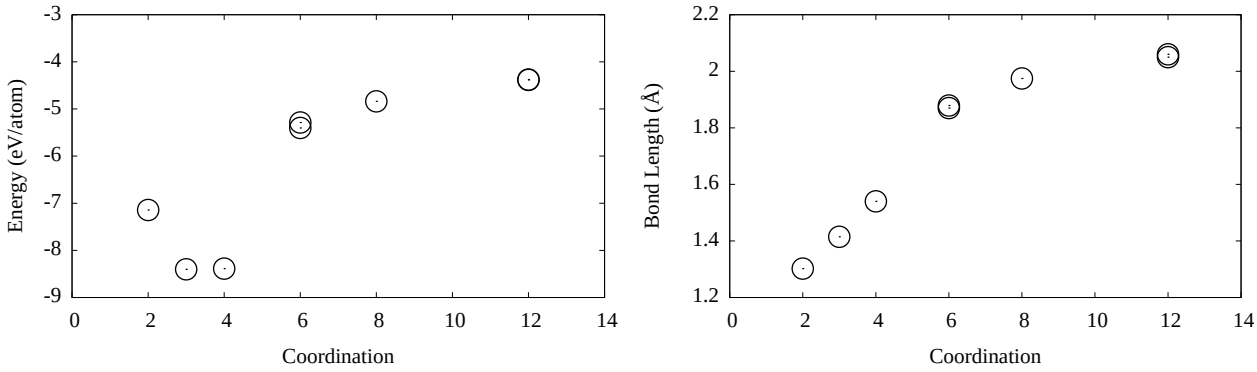


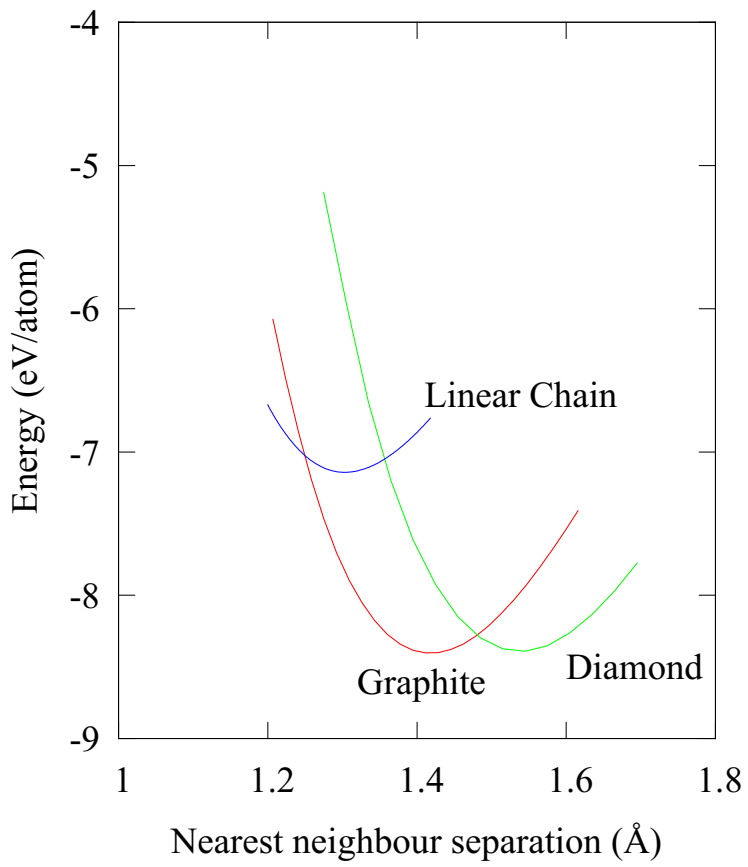
Figure 3.1: Graph illustrating rapid convergence of energy with system size for an armchair carbon nanotube.



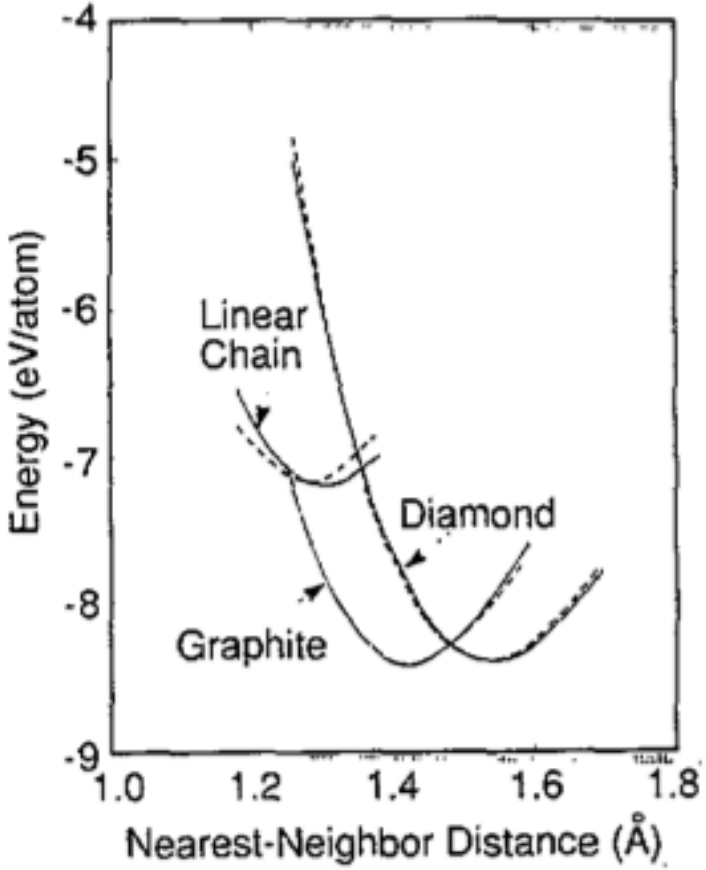
(a) Energy per atom versus coordination number.

(b) Bond length versus coordination number.

Figure 3.2: Plots showing how the energy per atom and bond lengths change depending on coordination number.

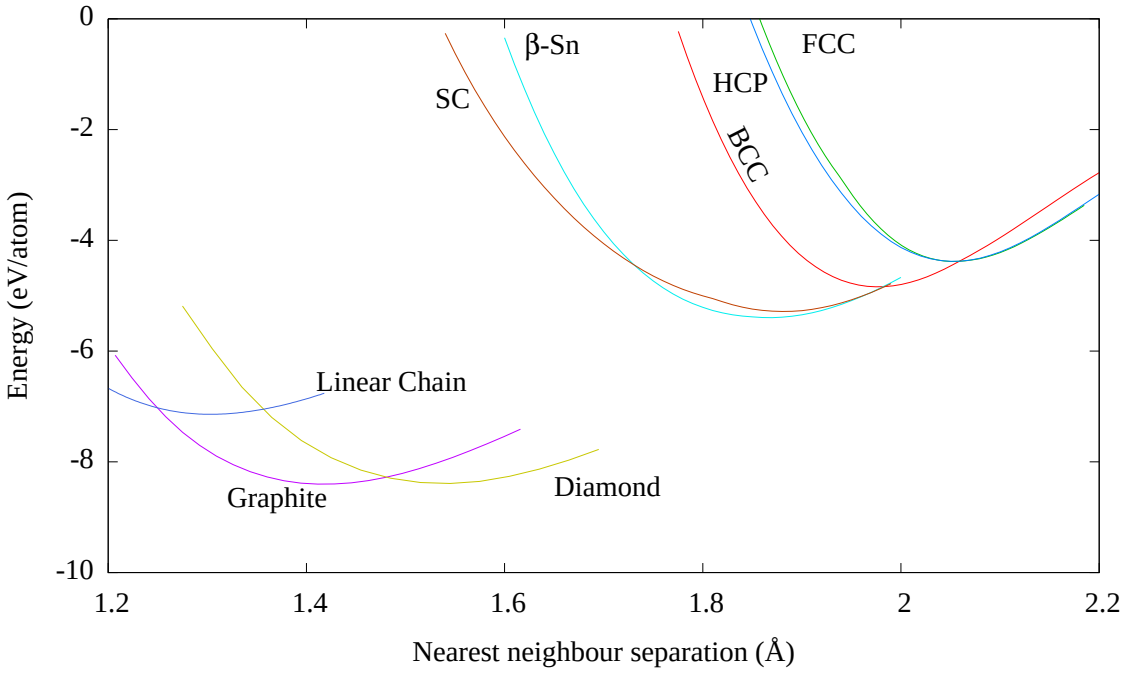


(a) `MolDyCaT`

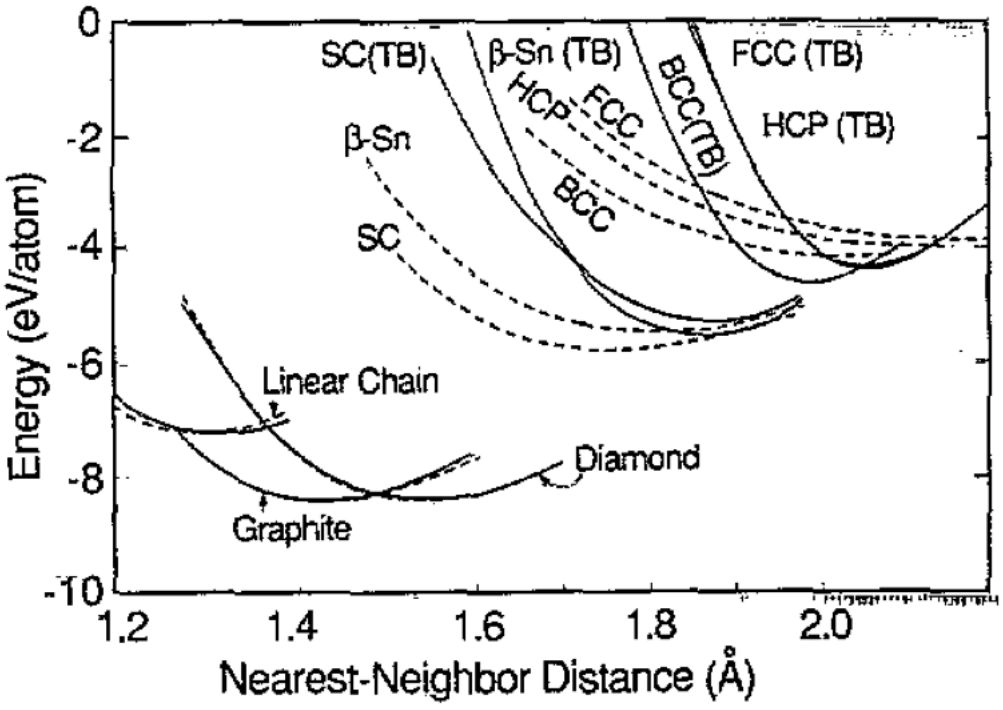


(b) Xu *et al.* [1]

Figure 3.3: Comparison of energy-nearest neighbour separation curves for the linear chain of carbon atoms, diamond and single layer graphite (i.e. graphene).



(a) `MolDyCaT`



(b) Xu *et al.* [1]

Figure 3.4: Comparison of all energy-nearest neighbour separation curves.

3.1.2 Elastic Constants

To work out the elastic constants the procedure outlined by Mehl *et al.* [22] was followed. This involved applying a volume conserving strain to the atomic co-ordinates (and lattice vectors), and expressing the energy as a function of this strain.

$$\epsilon_{ij} = \begin{pmatrix} \epsilon_1 & \epsilon_6/2 & \epsilon_5/2 \\ \epsilon_6/2 & \epsilon_2 & \epsilon_4/2 \\ \epsilon_5/2 & \epsilon_4/2 & \epsilon_3 \end{pmatrix}, \tag{3.1}$$

$$A'_i = (\delta_{ij} + \epsilon_{ij})\, A_j \,. \tag{3.2}$$

where A is the current position vector and A' is the new vector.

To calculate $c_{11} - c_{12}$ the following strain was applied:

$$\epsilon_1 = -\epsilon_2 = x \,, \tag{3.3}$$

$$\epsilon_3 = \frac{x}{1 - x^2} \,, \tag{3.4}$$

and all other elements were zero and $|x| < 0.05$ (as this is infinitesimal strain theory). The energy was then plotted as a function of x and $c_{11} - c_{12}$ found by applying the following fit [22]:

$$E(x)/V \approx (c_{11} - c_{12})x^2 + bx + c \,, \tag{3.5}$$

where b and c are fitting parameters.

For c_{44} a monoclinic strain was applied:

$$\epsilon_6 = x \,, \tag{3.6}$$

$$\epsilon_3 = \frac{x^2}{4 - x^2} \,, \tag{3.7}$$

and all other elements were zero. Care was taken as this involved deforming the lattice vectors to use non-orthorhombic unit cells. A fit was then applied to $E(x)$ [22]:

$$E(x)/V = c_{44}\frac{x^2}{2} + bx + c \,. \tag{3.8}$$

Figure 3.5 shows the data and the calculated fit on a 416 atom system of graphene to obtain $c_{11} - c_{12}$. As expected, the energies are symmetric in x. Table 3.6 shows the comparison between all `MolDyCaT` results, the results of Xu *et al.* and experimental values. The results of `MolDyCaT` generally agree with both, tending more towards experiment at times. This may simply be a reflection of the vast increases in computational power available now, 20 years on from the original paper, which allows `MolDyCaT` to study larger systems. As the results for elastic constant converge towards a final value with increasing system size - it is possible that `MolDyCaT` may have reached this convergence while Xu *et al.* did not.

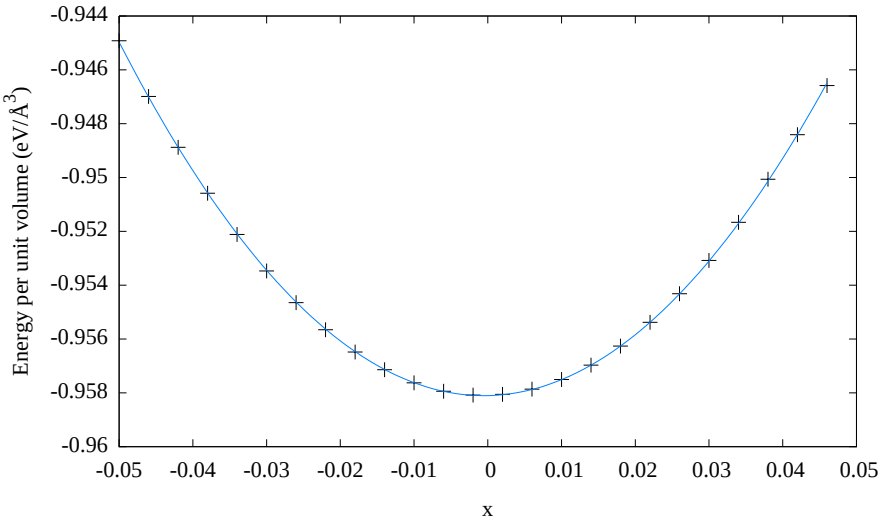


Figure 3.5: Graph of energy per unit volume of graphene under the strain applied to obtain $c_{11} - c_{12}$. The closeness of the fit indicates that the quadratic approximation was valid in the limit of an infinitesimal strain.

	`MolDyCaT`	Xu *et al.* .	Experiment
Graphene			
$c_{11} - c_{12}$	8.60	8.40	8.80
Diamond			
$c_{11} - c_{12}$	6.49	6.22	9.51
c_{44}	5.34	4.75	5.76

Figure 3.6: Comparison of elastic constants, in dyn cm^{-2}, obtained using `MolDyCaT` , from Xu *et al.* and experiment [1].

3.1.3 Phonons and Grüneisen Parameters

All phonon frequencies were calculated using the frozen-phonon super-cell approximation. For each mode, atoms were displaced from their equilibrium positions by a range of displacements, x, to represent a "frozen" phonon wave of that mode. The displacements were made small enough to enable the use of the harmonic approximation. The total energy is then

$$E = \frac{1}{2}NM\left(\frac{dx}{dt}\right)^2 + U(x=0) + \frac{1}{2}\frac{d^2U}{dx^2}\Big|_{x=0}x^2\,, \tag{3.9}$$

where N is the number of atoms in the unit cell, M is the mass of a carbon atom, and U is the energy obtained from the simulation. The first term on the right hand side represents the kinetic energy while the final two terms represents the potential energy.

Taking the derivative of equation with respect to t and invoking conservation of energy yields

$$NM\frac{d^2x}{dt^2} + \frac{d^2U}{dx^2}\Big|_{x=0}x = 0. \tag{3.10}$$

Therefore, by inspection, the angular frequency, ω, of the mode is

$$\omega = \left[\frac{d^2U}{dx^2}\Big|_{x=0}\frac{1}{NM}\right]^{\frac{1}{2}}\,, \tag{3.11}$$

where the second derivative of U is obtained from a parabolic fit of the energy around $x = 0$.

The Grüneisen parameter of mode i, γ_i, is defined by the equation:

$$\gamma_i = -\frac{V}{\omega_i}\frac{\partial\omega_i}{\partial V}\,, \tag{3.12}$$

for volume V and phonon frequency ω_i. They were determined by changing the volume of the unit cell and re-computing the phonon frequencies. This was done by applying a strain to each atom (and the lattice vectors) of the form:

$$\epsilon = \begin{pmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & x \end{pmatrix}\,, \tag{3.13}$$

for diamond, and

$$\epsilon = \begin{pmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & 0 \end{pmatrix}\,, \tag{3.14}$$

for graphite.

	MolDyCaT	Xu *et al.* .	Experiment
Diamond			
$\nu_{\mathrm{LTO}}(\Gamma)$	36.79	37.80	39.90
$\nu_{\mathrm{TA}}(X)$	22.44	22.42	24.20
$\nu_{\mathrm{TO}}(X)$	33.22	33.75	32.0
$\nu_{\mathrm{LA}}(X)$	34.61	34.75	35.5
$\gamma_{\mathrm{LTO}}(\Gamma)$	1.04	1.03	0.96
$\gamma_{\mathrm{TA}}(X)$	-0.17	-0.16	
$\gamma_{\mathrm{TO}}(X)$	1.09	1.10	
$\gamma_{\mathrm{LA}}(X)$	0.60	0.62	
Graphite			
E_{2g_2}	49.24	49.92	47.46
A_{2u}	29.18	29.19	26.04
$\gamma(E_{2g_2})$	2.02	2.00	
$\gamma(A_{2u})$	0.06	0.10	

Table 3.1: Comparison of phonon frequencies and Grüneisen parameters of diamond and graphite, in THz, obtained using MolDyCaT , from Xu *et al.* and experiment [1].

The phonon frequencies were then plotted as a function of volume around ω_i and a straight line was fitted to obtain $\frac{\partial \omega_i}{\partial V}$. The results are shown in Table 3.1. The results from MolDyCaT match well with those of Xu *et al.* , the small discrepancies could be due to statistical errors in the fits that were used. Alternatively, Xu *et al.* give no detail about how they calculated their values and so the discrepencies could be due to a different method being used.

3.1.4 Carbon Clusters

Geometry optimisations were carried out on small carbon clusters, C2 through C10, using the SD relaxation method. Starting from a equally spaced linear chain for C2, 3, 4, 5, 7, 9 and an approximate ring shape for C6, 8, 10 the structures were relaxed to find the lowest energy state, within a small 0.01 eV. The structures converged to those shown in figure 3.7. All ring shaped clusters show the same bond lengths as Xu *et al.* , however, for linear chains, the bond lengths do not match. This is because MolDyCaT does not contain a Hubbard correction for dangling bonds. The Hubbard correction could be added in future but is in general unnecessary because MolDyCaT is best suited to simulation of large periodic systems. Despite the non-inclusion of this term the bond lengths are still symmetric

across the chains for all linear clusters.

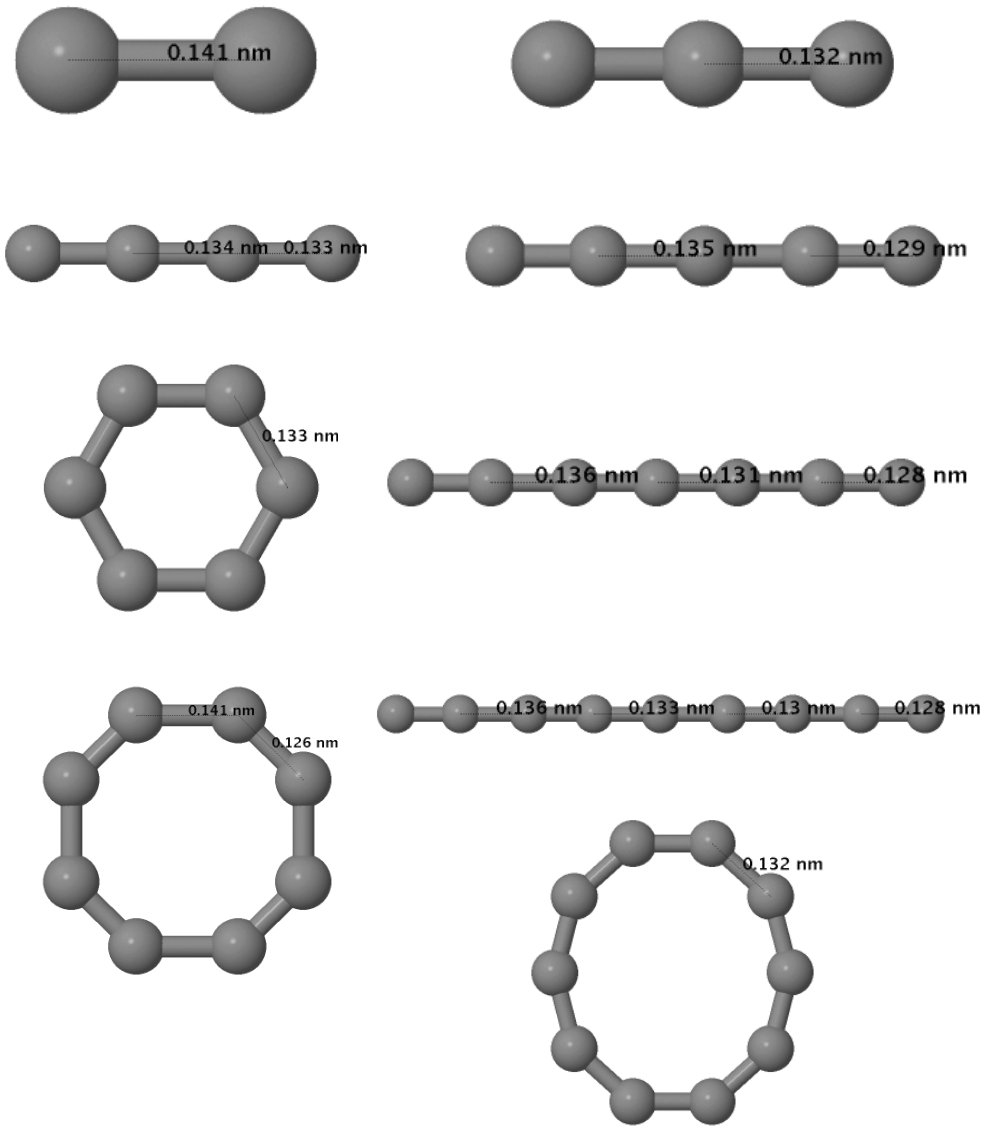


Figure 3.7: Optimised carbon cluster geometries. The linear chains are symmetric. Images produced using Jmol [23].

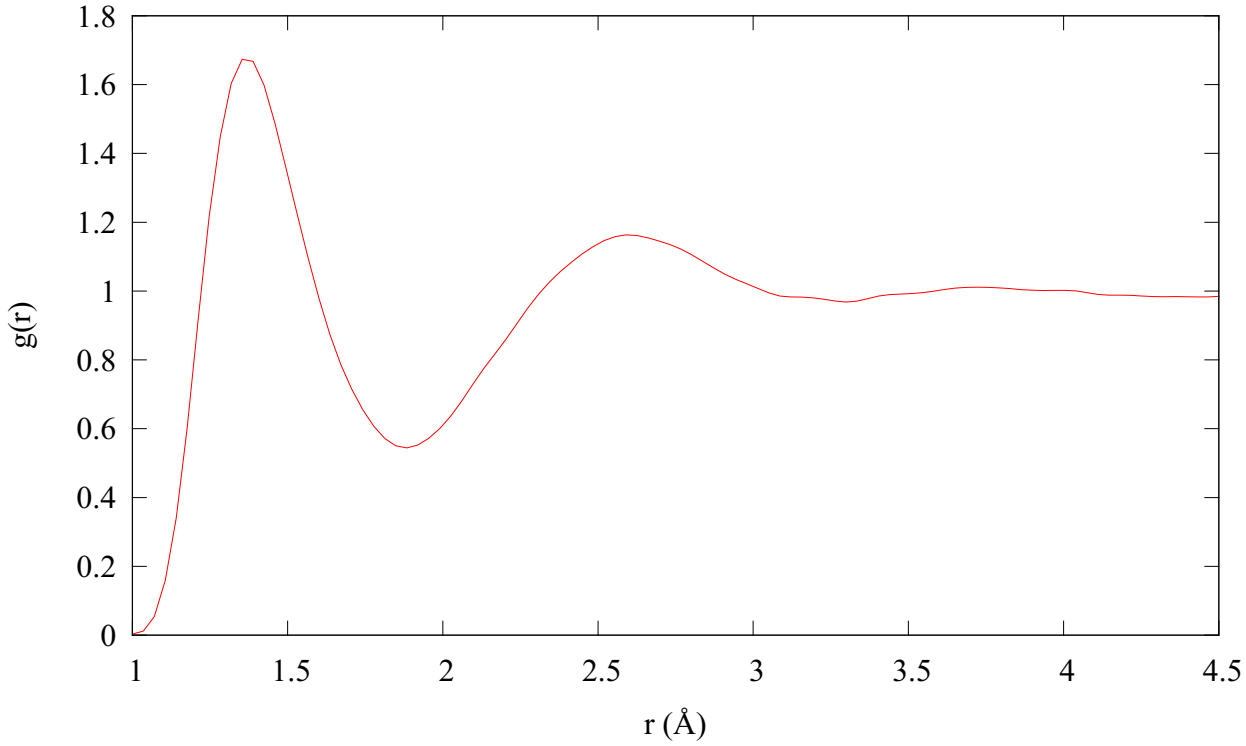


Figure 3.8: 5000 K Liquid Carbon RDF.

3.1.5 Liquid Carbon

A cell containing 108 atoms sized to fix the density at 2.0 gcm^{-3}, was arranged on a FCC lattice and equilibrated at 5000 K for 5000 steps with a time step of 0.5 fs using the Andersen thermostat. A simulation of 50000 steps with the same time-step was then run to produce the RDF for liquid carbon, figure 3.8. The RDF showed lower peaks than those in Xu *et al.* This may be attributed to finite size errors arising from studying smaller systems.

3.2 Further Studies

3.2.1 Optimisers

The convergence of the different geometry optimisation routines was tested on the C8 cluster, figure 3.9. Despite a significantly distorted initial structure, all optimisers were able to converge to the same final configuration, albeit requiring very different numbers of force evaluations. The `SD_STEP_SIZE` parameter was set to 0.001 for the SD solver and the `DMD_GAMMA_RECALC_STEPS` set to 60. All other parameters were left at their default values. The CG and DMD-CM methods were able to converge to lower force tolerances in the 75 step window shown.

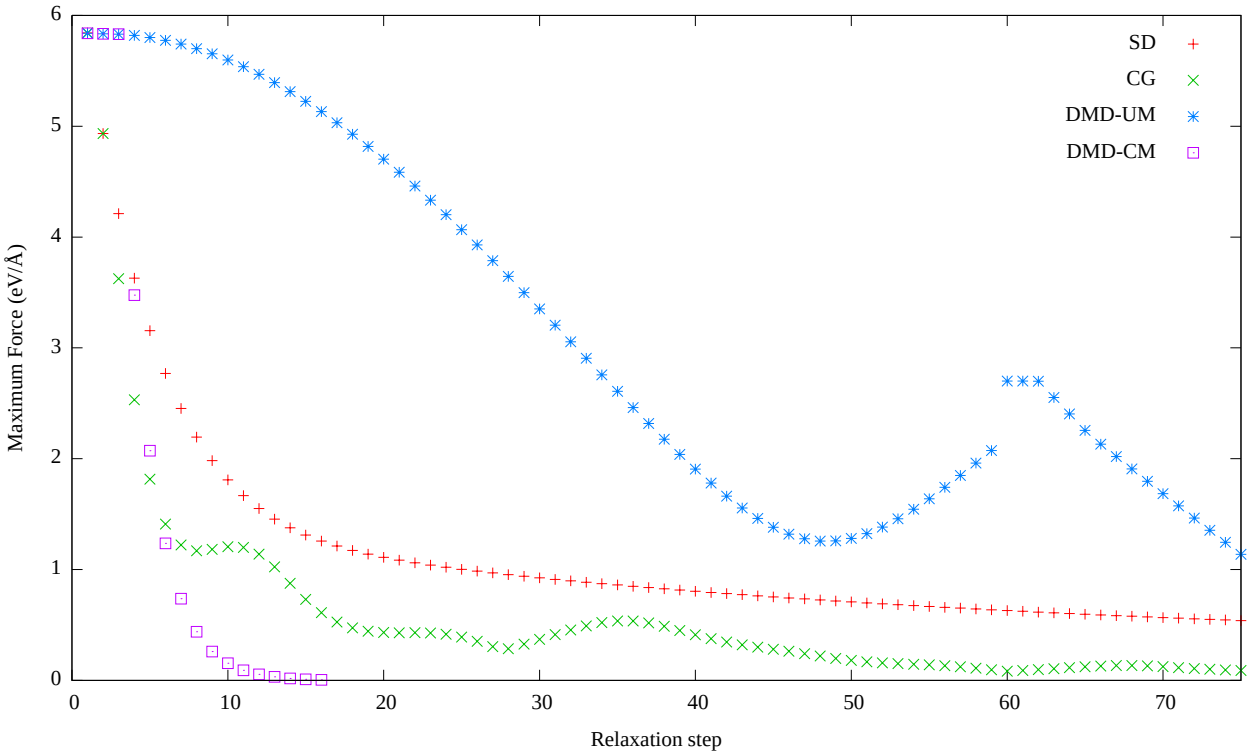


Figure 3.9: Convergence of different optimiser routines. The DMD-UM discontinuity at step 60 is caused by recalculation of the damping factors $\gamma_{i\alpha}$, thereby interrupting the usual dynamics.

3.2.2 Nanotubes

Energy versus atomic separation curves were produced for three different nanotubes, figure 3.10. Different nanotubes showed varying equilibrium bond lengths and energies, which `MolDyCaT` was comfortably able to distinguish. This means that `MolDyCaT` can be used to characterise different nanotubes and pick those with the required properties for particular applications. For example calculations of the bulk modulus could be carried out to identify which nanotube structures are strongest.

3.2.3 Point Defects

The energy of formation of vacancies was studied for diamond (216 atoms), graphite (modelled as a sheet of graphene with 160 atoms) and two different nanotubes, (10,5) (140 atoms) and (13,0) (156 atoms). For each system, the complete structure was relaxed using the SD method, with periodic boundary conditions, to find the lowest energy. Then a single vacancy was introduced and the structure relaxed to find the lowest energy. The relaxations were converged until the maximum force on any atom was lower than 0.02 eV/Å, beyond which there was no discernible change in energy. This relatively large tolerance was permissible as energies are variational - small errors in

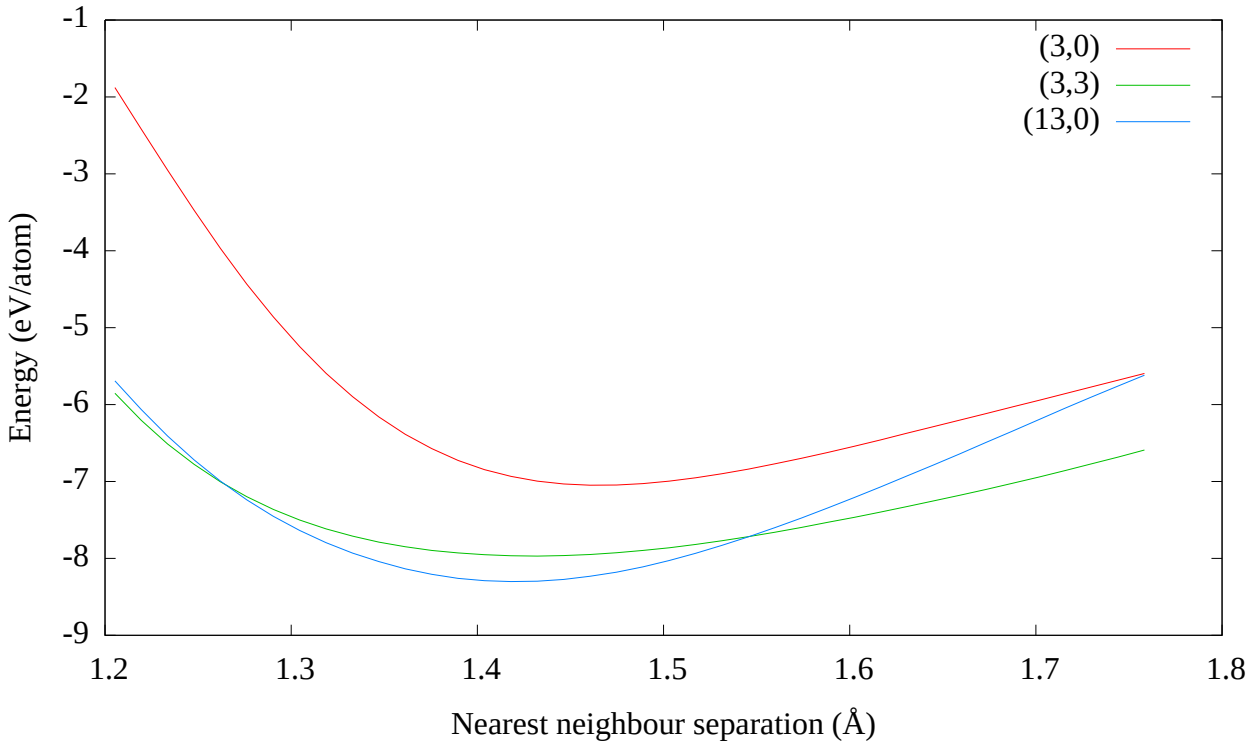


Figure 3.10: Energy-nearest neighbour separation curves for three different nanotubes.

equilibrium structure equate to very small second order errors in energy. The finite size corrected vacancy formation energies were then calculated according to equation 3.15 [24], where μ is the chemical potential calculated as the energy of a single atom in the pure system and acts as a finite size correction:

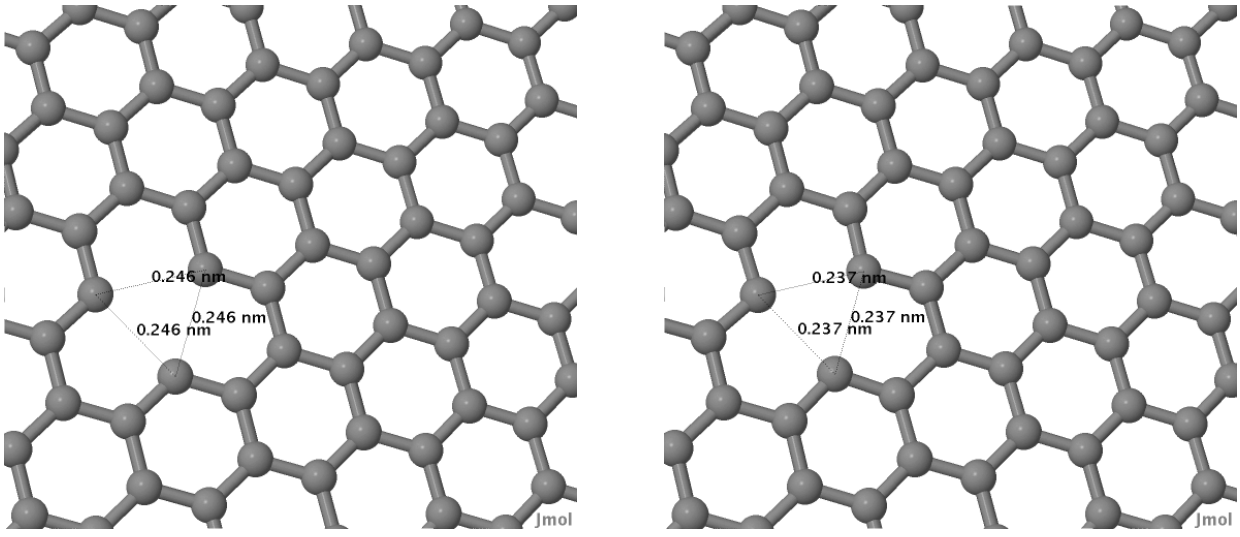
$$E_{\text{form}}\,[\text{vac}] = E_{\text{T}}\,[\text{vac}] + \mu\,[\text{C}] + E_{\text{T}}\,[\text{pure}]\ . \tag{3.15}$$

Table 3.11 shows the values calculated for the various energies of vacancy formation as well as those from DFT calculations and experiment. The energy for graphite agrees well with experiment, while that for diamond is closer to experiment than the DFT value. Those for nanotubes are close to those from DFT. These figures show that the tight-binding model employed in `MolDyCaT` is a good substitute for DFT when less computational power is available. Figure 3.12 shows the relaxation of the graphite sheet when the vacancy is introduced. The moving together of the atoms around the vacancy is uniform in this case. DFT has shown that in actuality two of the atoms move closer together [25]. This is a Jahn-Teller distortion and does not appear to be adequately described in this model.

Calculations were also carried out for the addition of a carbon interstitial to the 216 atom diamond lattice. It was found to have an energy of 12.82 eV underestimating the DFT value of 15.8 eV [26].

Material	`MolDyCaT`	DFT	Experiment
Graphite	7.26	7.6 [25]	7.0 ± 0.5 [2]
Diamond	8.74	7.2 [26]	9-15 [27]
(10,5) nanotube	6.83	6.59 [24]	
(13,0) nanotube	6.74	6.21 [24]	

Figure 3.11: Vacancy formation energies from `MolDyCaT`, DFT and experiment. Units are eV.



(a) Before Relaxation **(b)** After Relaxtion

Figure 3.12: Graphite sheet with vacancy before and after relaxation. Image produced using Jmol [23]

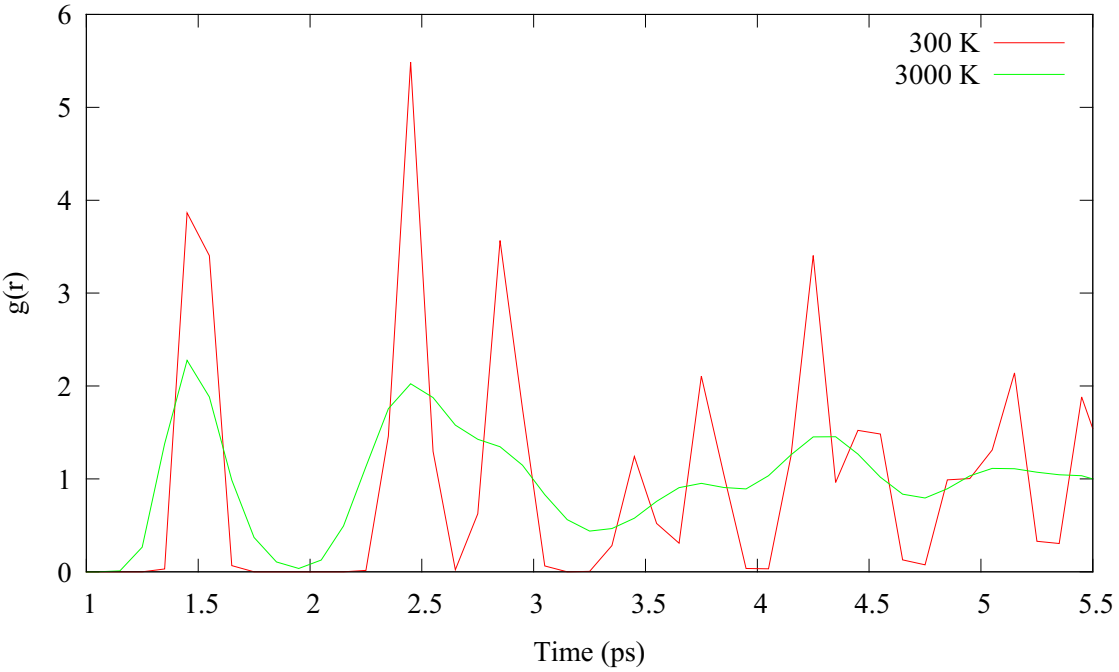


Figure 3.13: Radial distribution functions of diamond at 300 K (red) and 3000 K (green).

This discrepancy is likely due to the incomplete treatment of over- and under-coordinated bonds within the sp^3 hybridised model.

3.2.4 MD simulations

MD simulations of 64 atoms of diamond were carried out at 300 K and 3000 K using the Andersen thermostat. The simulations used a 0.5 fs timestep and were first equilibrated for 5000 steps before a further 'production run' of 5000 steps without thermostatic control. The RDFs for the two simulations are shown in figure 3.13. Significant thermal broadening can be seen at 3000 K as would be expected. The simulations conserved energy well, as shown by figure 3.14.

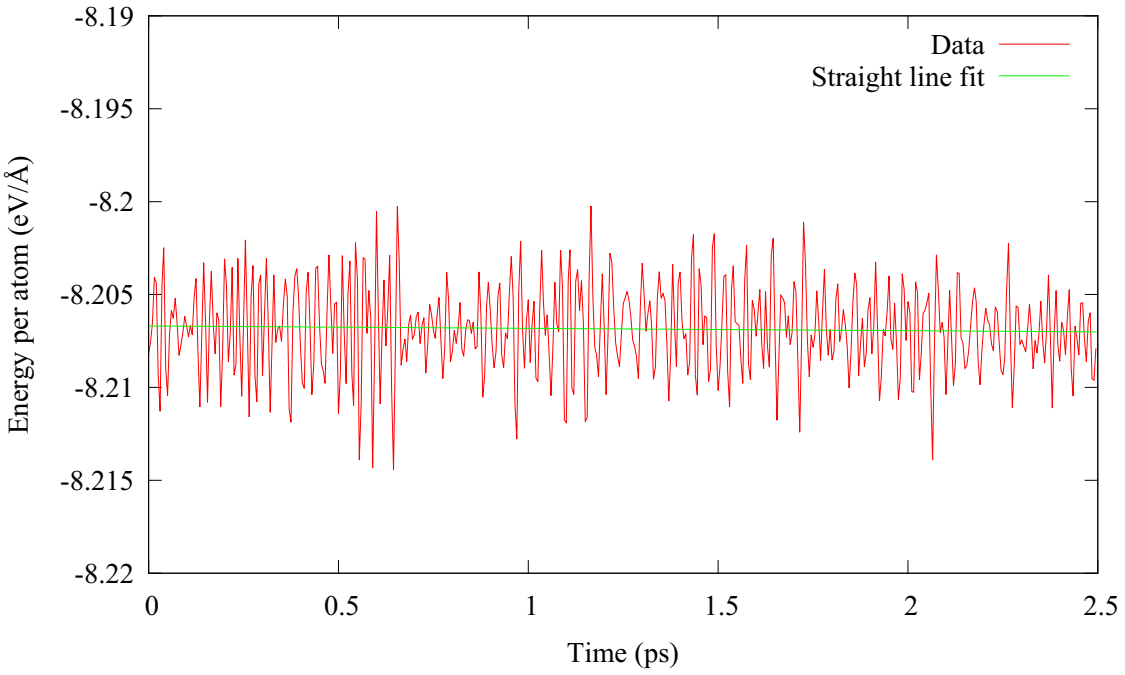


Figure 3.14: Energy per atom for diamond at 300 K over the course of a 2.5 ps simulation. As expected from a molecular dynamics simulation, there are minor energy fluctuations. The straight line fit performed indicates that the energy drift is negligible.

3.2.5 Bulk Modulus

The bulk modulus, K, is defined as:

$$K = -V\frac{dP}{dV}\,, \tag{3.16}$$

$$= V\frac{d^2E}{dV^2}\,. \tag{3.17}$$

Therefore the energy was computed for different structures as a function of volume, following a procedure analogous to that for the elastic constants. A (non-volume conserving) strain was applied to the positions, of the form:

$$\epsilon = \begin{pmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & x \end{pmatrix}\,. \tag{3.18}$$

The second derivative was computed by fitting a quadratic to the energy-volume curve, and V was taken as the volume at the minimum of this curve.

This method produced a value of 454 GPa for the bulk modulus of diamond. This compares well to both the DFT value of 422 GPa [28] and the experimental value of 442 GPa [29]. The main interactions between sheets of graphite are van der Waals forces. This model does not include such interactions and as such the bulk modulus of graphite was not calculated as it would not even have been close to the correct value. Methods exist for the incorporation of these interactions in tight binding models [30] and these could be implemented in `MolDyCaT` in future. Their implementation would also allow the study of arrays of nanotubes as nanotubes interact through van der Waals forces.

The procedure for finding the bulk modulus could have been improved by applying by fitting the Birch-Murnaghan equation of state:

$$E(V) = E_0 + \frac{9V_0K_0}{16}\left\{\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]K_0' + \left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^2\left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]\right\}\,, \tag{3.19}$$

where $K = K_0 + K_0'P$ or simply $K = K_0$ when $V = V_0$ [31].

Chapter 4

Project Management

4.1 Approach

The team adhered closely to the modular approach laid out in the code specification. Before starting, the pivotal features to be implemented were decided upon and the tasks were delegated to smaller subgroups within the team. This approach worked well on the whole, but occasionally caused issues when the progress of one subgroup depended on the results of another subgroup. The effects of such bottlenecks were mitigated by designing tasks to reduce the number of such interdependencies. They could have been further mitigated by including clear structures for the information inputted and outputted from each module in the code specification [7] to allow more independent operation of the subgroups.

Initially programming was planned to be done in pairs. The idea was that this would enable one person to work on writing the code, while the other person kept an eye on the big picture, looking for any trivial errors or logical flaws in the programming. As the project progressed, this technique was utilised less and less, with the members of the team preferring to go and work on their own, consulting with their team-mates only when needed. Often the whole team would assemble in one room for team programming sessions to make communication between subgroups easy.

4.2 Tools

Different software and web tools were continually evaluated throughout the course of this project. The first solution to sharing files between team members involved using a web based file storage system, Dropbox[1]. This was found to be useful for sharing papers, minutes of meetings etc., but too convoluted for collaborative work e.g. for writing the code specification. For all subsequent tasks it

[1] `http://www.dropbox.com`

was decided to use SVN.

For task management the team tried a web based tool called Pivotal Tracker[2]. This site allows team members to set up, assign and track tasks for various features required throughout a typical developmental cycle. However, it was quickly found to be more trouble than it was worth for such a relatively small project. The overhead of maintaining the tracker outweighed any potential productivity boost. Pivotal Tracker is adept at logging and tracking a large number of bugs; but as everyone had a role in development, bugs were dealt with as soon as they were noticed so there was very little back log. With regular meetings and emails most people were aware of everyone else's roles in the team, so the Pivotal Tracker account quickly fell into disuse.

Whenever possible, the team assembled on at least a weekly basis. Each team member held a brief presentation outlining any major developments in their part of the code. This provided a forum in which potential problems were discussed and conventions by which different parts of the code communicated were agreed upon. In theory this also equipped every team member with the knowledge to debug code anywhere in the program. A formal agenda was not required for meetings but written minutes were produced for each meeting. These outlined the tasks to be attempted by each team member before the next meeting and proved a useful reference for tracking the progress of the project.

A SVN repository was used for version control during the code developments. Frequent commits with detailed log messages were encouraged and proved indispensable in undoing changes which broke parts of the code. All commits were tested to ensured that they would not cause compilation problems for other team members. The team decided that members who committed code that did not compile should bring snacks to the next meeting to discourage poor testing practice.

In parallel to the development of `MolDyCaT` a second group of students, team `TAMA`, were assigned the same project of implementing the Xu *et al.* [1] potential. In the spirit of scientific collaboration there was mutual exchange of ideas and corroboration of results between the two teams. Supplementary to the intra-group meetings, weekly meetings were held with team `TAMA` and the two course supervisors.

Of the total marks assigned to the team, a portion is to be allocated according to the preferences of the team. A number of approaches were considered including an equal share of marks between group members but, as not all team members were able to contribute equally, this was not considered fair. Instead, a variant of the Knickrehm *World Series Method* [32, 33] was decided upon. Each team member will submit an anonymous ballot, to an external mediator, allocating a mark to each other member of the team. These will then be averaged and the marks allocated according to these proportions.

[2] `http://www.pivotaltracker.com/`

Chapter 5

Conclusion

As the result of many hours of hard work the tight binding carbon model of Xu *et al.* was successfully implemented and all the results of their paper faithfully reproduced. The transferability of the potential was tested by applying it to systems beyond those they studied, including to allotropes of carbon discovered in the 20 years since their paper was published. Material properties unaffected by van der Waals interactions were recovered with good agreement with DFT calculations and experimental values. Future work could focus on including semi-empirical corrections to facilitate accurate modelling of such systems. `MolDyCaT` can be extended to include the Hubbard correction for dangling and floating bonds for modelling carbon clusters [1].

Throughout the project the team worked together effectively to deliver a demanding project to an unaccommodating timetable. Initial designs proved overly ambitious and were soon focused down to more manageable objectives. This left time for skills transfer between team members as the more experienced coders were available to troubleshoot the programming problems of other team members as they arose. Frequent meetings and the ensuing discussions allowed the whole group to benefit from the lessons learnt by each team member.

Appendix A

User Documentation

A.1 Installation Requirements

To run `MolDyCaT` the library Armadillo version 2.4.2 will need to be installed. The code is compatible with version 2.4.2, however compatibility with other versions of Armadillo cannot be guaranteed. The build system CMake will also need to be installed. The code is compatible with CMake versions 2.6 and above. The code is compatible with the GNU[1] `c++` compiler, `g++`, versions 2.6.2 and higher; however CMake allows for cross platform compatability so, any `c++` compiler should be able to be used to compile the code.

A.2 Running `MolDyCaT`

Running the configuration script, `configure`, located in the root of the svn repository, in linux or unix environments will build and make the program and set up the directory structure if it is not already present.

To run `MolDyCaT` run the executable and give it the seed name of the input file:

`./moldycat {seed}`

This will then parse in the input file `inputs/{seed}_params.in` and the positions file indicated in the input file - if no input file is specified, the default positions file that is looked for is `{seed}_atoms.in` in the `inputs` folder.

[1] http://gcc.gnu.org/

A.3 Input Files

The code comes with two example input files located in the `inputs` folder. The first, `inputs/energy_params.in` shows the input parameters required for energy calculations, while the second `inputs/md_params.in` gives parameters for Molecular Dynamics simulations.

`energy_params.in`

```
#########################################################################
# TASK:
# ENERGY - Single energy calculation
# MD - Molecular dynamics
# PHONON - Calculate phonons
#########################################################################
TASK ENERGY


#########################################################################
# Restart flag: ON or OFF. If ON it will
# read in {seed}_positions.end, {seed}_velocities.end
# and {seed}_unitcell.out from the outputs folder
#########################################################################
RESTART OFF


#########################################################################
# CELL_DIMENSIONS X Y Z in Angstroms
# CELL_ANGLES alpha beta gamma in degrees
# BOUNDARY_TYPE can be open or periodic
#
# NB The length of any dimension of the cell should not be
#    be less than 5.2 after applying scaling to the unit cell
# NB CELL_ANGLES can only be 90 90 90
#
#
# The default values for BOUNDARY_TYPE are Open Open Open
#########################################################################
CELL_DIMENSIONS 3.4641 3.4641 3.4641
CELL_ANGLES 90.0 90.0 90.0
BOUNDARY_TYPE Periodic Periodic Periodic

#########################################################################
# SUPERCELL: Y(es) or N(o).
# MULTIPLY_CELL_DIRN  [integer] number of unit cells in the x, y
```

```
# or z direction
#
# The default value for SUPERCELL is N
# The default value for MULTIPLY_CELL_DIRN is 1 1 1
#####################################################################
SUPERCELL Y
MULTIPLY_CELL_DIRN 2 2 2

#####################################################################
# STRUCT_RELAX: ON or OFF
# Relaxation takes place prior to MD
# RELAXER: DMDUM (Damped MD uncopuled) or DMDCM (Damped MD coupled)
# or SD (Steepest Descent)
# FORCE_TOL Stops when force less than this
# RELAX_STEPS max number of iterations for relaxation
# SD_STEP_SIZE alpha for steepest descent
# DMD_GAMMA_RECALC_STEPS Number of steps to wait before periodically
#                        recomputing damping factor gamma
#                        (best left unchanged unless you have experience
#                        with the dark arts as rather temperamental)
# DMD_TIMESTEP_SCALE     Timestep for DMD calculations is calculated
#                        using 2*PI/(GAMMA*DMD_TIMESTEP_SCALE),
#                        decreasing may speed up convergence (or break it)
#
# The default value for STRUCT_RELAX is Off.
# The default value for RELAXER is CG.
# The default value for FORCE_TOL is 0.01.
# The default value of RELAX_STEPS is 100.
#####################################################################
STRUCT_RELAX OFF
RELAXER SD
FORCE_TOL 0.1
RELAX_STEPS 6000
SD_STEP_SIZE 0.001

DMD_GAMMA_RECALC_STEPS 10000
DMD_TIMESTEP_SCALE 7

#####################################################################
# The name of the positions file in the inputs folder.
#####################################################################
ATOMS_FILE_ADDRESS diamond.txt

#####################################################################
# TIME_STEP in femtoseconds
```

```
# MD_STEPS number of time steps
########################################################################
TIME_STEP 0.1
MD_STEPS 1000

########################################################################
# MD parameters for how often data is written out.
#
# If no parameters are specified the default values are 100 for
# each.
########################################################################
ENERGY_OUT_STEPS 10
VELOCITY_OUT_STEPS 100
POSITIONS_OUT_STEPS 100

########################################################################
# RDF_STEPS how often data is collected for rdf in time steps
# RDF_CUTOFF rdf is calculation up to this value in Angstroms
# RDF_BINS number of histogram bins for rdf
########################################################################

RDF_STEPS 5
RDF_CUTOFF 10
RDF_BINS 100

########################################################################
# TEMPERATURE in Kelvin
#
# TEMP_TOL Fraction of temperature
#
# THERMOSTAT ANDERSEN or VS (velocity scaling) or NOTHERMO for no thermostat
#
# The default value for TEMPERATURE is 350
# The default value for TEMP_TOL is 0.01
# The default value for THERMOSTAT is VS
########################################################################
TEMPERATURE 100
TEMP_TOL 0.01
THERMOSTAT ANDERSEN


########################################################################
# The default value for INTEGRATOR is VERLET
#
# Naming convention for the symplectic integrators follows:
```

```
# http://jcp.aip.org/resource/1/jcpsa6/v101/i5/p4062_s1
# where the symplectic integrator types are
#
# si2.a, si3.a, si3.b, si4.a, si4.b, si4.c and si6.a
#
# The numbers stand for the order and the letters coefficient
# choices.
#########################################################################
INTEGRATOR VERLET
```

md_params.in

```
#########################################################################
# TASK:
# ENERGY - Single energy calculation
# MD - Molecular dynamics
# PHONON - Calculate phonons
#########################################################################
TASK MD


#########################################################################
# Restart flag: ON or OFF. If ON it will
# read in {seed}_positions.end, {seed}_velocities.end
# and {seed}_unitcell.out from the outputs folder
#########################################################################
RESTART OFF


#########################################################################
# CELL_DIMENSIONS X Y Z in Angstroms
# CELL_ANGLES alpha beta gamma in degrees
# BOUNDARY_TYPE can be open or periodic
#
# NB The length of any dimension of the cell should not be
#    be less than 5.2 after applying scaling to the unit cell
# NB CELL_ANGLES can only be 90 90 90
#
#
# The default values for BOUNDARY_TYPE are Open Open Open
#########################################################################
CELL_DIMENSIONS 3.4641 3.4641 3.4641
CELL_ANGLES 90.0 90.0 90.0
BOUNDARY_TYPE Periodic Periodic Periodic
```

```
33
34 ######################################################################
35 # SUPERCELL: Y(es) or N(o).
36 # MULTIPLY_CELL_DIRN  [integer] number of unit cells in the x, y
37 # or z direction
38 #
39 # The default value for SUPERCELL is N
40 # The default value for MULTIPLY_CELL_DIRN is 1 1 1
41 ######################################################################
42 SUPERCELL Y
43 MULTIPLY_CELL_DIRN 2 2 2
44
45 ######################################################################
46 # STRUCT_RELAX: ON or OFF
47 # Relaxation takes place prior to MD
48 # RELAXER: DMDUM (Damped MD uncopuled) or DMDCM (Damped MD coupled)
49 # or SD (Steepest Descent)
50 # FORCE_TOL Stops when force less than this
51 # RELAX_STEPS max number of iterations for relaxation
52 # SD_STEP_SIZE alpha for steepest descent
53 # DMD_GAMMA_RECALC_STEPS Number of steps to wait before periodically
54 #                        recomputing damping factor gamma
55 #                        (best left unchanged unless you have experience
56 #                        with the dark arts as rather temperamental)
57 # DMD_TIMESTEP_SCALE     Timestep for DMD calculations is calculated
58 #                        using 2*PI/(GAMMA*DMD_TIMESTEP_SCALE),
59 #                        decreasing may speed up convergence (or break it)
60 #
61 # The default value for STRUCT_RELAX is Off.
62 # The default value for RELAXER is CG.
63 # The default value for FORCE_TOL is 0.01.
64 # The default value of RELAX_STEPS is 100.
65 ######################################################################
66 STRUCT_RELAX On
67 RELAXER SD
68 FORCE_TOL 0.1
69 RELAX_STEPS 6000
70 SD_STEP_SIZE 0.001
71
72 DMD_GAMMA_RECALC_STEPS 10000
73 DMD_TIMESTEP_SCALE 7
74
75 ######################################################################
76 # The name of the positions file in the inputs folder.
77 ######################################################################
```

```
78 ATOMS_FILE_ADDRESS diamond.txt
79
80 ##########################################################################
81 # TIME_STEP in femtoseconds
82 # MD_STEPS number of time steps
83 ##########################################################################
84 TIME_STEP 0.1
85 MD_STEPS 1000
86
87 ##########################################################################
88 # MD parameters for how often data is written out.
89 #
90 # If no parameters are specified the default values are 100 for
91 # each.
92 ##########################################################################
93 ENERGY_OUT_STEPS 10
94 VELOCITY_OUT_STEPS 100
95 POSITIONS_OUT_STEPS 100
96
97 ##########################################################################
98 # RDF_STEPS how often data is collected for rdf in time steps
99 # RDF_CUTOFF rdf is calculation up to this value in Angstroms
100 # RDF_BINS number of histogram bins for rdf
101 ##########################################################################
102
103 RDF_STEPS 5
104 RDF_CUTOFF 10
105 RDF_BINS 100
106
107 ##########################################################################
108 # TEMPERATURE in Kelvin
109 #
110 # TEMP_TOL Fraction of temperature
111 #
112 # THERMOSTAT ANDERSEN or VS (velocity scaling) or NOTHERMO for no thermostat
113 #
114 # The default value for TEMPERATURE is 350
115 # The default value for TEMP_TOL is 0.01
116 # The default value for THERMOSTAT is VS
117 ##########################################################################
118 TEMPERATURE 100
119 TEMP_TOL 0.01
120 THERMOSTAT ANDERSEN
121
122
```

```
123 ######################################################################
124 # The default value for INTEGRATOR is VERLET
125 #
126 # Naming convention for the symplectic integrators follows:
127 # http://jcp.aip.org/resource/1/jcpsa6/v101/i5/p4062_s1
128 # where the symplectic integrator types are
129 #
130 # si2.a, si3.a, si3.b, si4.a, si4.b, si4.c and si6.a
131 #
132 # The numbers stand for the order and the letters coefficient
133 # choices.
134 ######################################################################
135 INTEGRATOR VERLET
```

A.4 Output Files

The outputted files are detailed below.

{seed}_param.out

Lists all parameters as read in from {seed}_params.in.

Structural relaxations

{seed}_opt.xyz

Contains the positions at each step of a geometry optimisations. Compatible with Jmol for imaging.

{seed}_relax.out

Contains iteration number, total system energy, maximum force per atom.

Molecular Dynamics

{seed}_positions.xyz

Positions, at every POSITION_OUT_STEPS, compatible with Jmol for imaging.

{seed}_velocities.out

Velocities at every VELOCITY_OUT_STEPS.

{seed}_temp.out

Iteration number, temperature.

{seed}_energy.out

Iteration number and total energy at every ENERGY_OUT_STEPS.

{seed}_rdf.dat

Radial distribution function, if RDF_STEPS $\neq 0$.

{seed}_positions.end

Restart file, contains positions from end of simulation (or mid-way through if simulation did not end).

{seed}_velocities.end

Restart file, contains velocities from end of simulation, printed out every ENERGY_OUT_STEPS .

{seed}_unitcell.out

Restart file, contains details of the unitcell.

References

[1] C H Xu et al. A transferable tight-binding potential for carbon. J. Phys. Cond. Mat., 4:6047–6054, 1992.

[2] PA Thrower and RM Mayer. Point defects and self-diffusion in graphite. Physica status solidi (a), 47(1):11–37, 1978.

[3] K.K.S. Lau, J. Bico, K.B.K. Teo, M. Chhowalla, G.A.J. Amaratunga, W.I. Milne, G.H. McKinley, and K.K. Gleason. Superhydrophobic carbon nanotube forests. Nano Letters, 3(12):1701–1705, 2003.

[4] A.K. Geim and K.S. Novoselov. The rise of graphene. Nature materials, 6(3):183–191, 2007.

[5] M. Finnis. Interatomic forces in condensed matter. Oxford University Press Oxford, 2003.

[6] WMC Foulkes. Accuracy of the chemical-pseudopotential method for tetrahedral semiconductors. Phys. Rev. B, 48(19):14216, 1993.

[7] V Chen, M Coury, A Hammad, B Kaube, M Khawaja, and D Rathbone. Moldycat code specification. Theory and Simulation of Materials Doctoral Training Centre at Imperial College London, 2012.

[8] L. Goodwin, AJ Skinner, and DG Pettifor. Generating transferable tight-binding parameters: application to silicon. EPL (Europhysics Letters), 9:701, 1989.

[9] WA Harrison. New tight-binding parameters for covalent solids obtained using louie peripheral states. Physical Review B, 24(10):5835, 1981.

[10] M. Johnson, N. Malikova, M. Plazanet, and K. Parlinski. Phonons calculated from first-principles. École thématique de la Société Française de la Neutronique, 12:161–166, 2011.

[11] F. Tassone, F. Mauri, and R. Car. Acceleration schemes for ab initio molecular-dynamics simulations and electronic-structure calculations. Physical Review B, 50(15):10561, 1994.

[12] MIJ Probert. Improved algorithm for geometry optimisation using damped molecular dynamics. J. Comp.Phys., 191(1):130–146, 2003.

[13] Music - multipurpose simulation code. non-orthorombic cells. http://zeolites.cqe.northwestern.edu/Music/nonorthorhombic.pdf.

[14] P.H. Hünenberger. Thermostat algorithms for molecular dynamics simulations. Advanced Computer Simulation, pages 130–130, 2005.

[15] H.C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. The Journal of chemical physics, 72:2384, 1980.

[16] C. Schlier and A. Seiter. Symplectic integration of classical trajectories: A case study. The Journal of Physical Chemistry A, 102(47):9399–9404, 1998.

[17] S.K. Gray, D.W. Noid, and B.G. Sumpter. Symplectic integrators for large scale molecular dynamics simulations: A comparison of several explicit methods. J. Chem Phys., 101:4062, 1994.

[18] Music. These are some preliminary notes on non-orthorhombic unit cell simulations using music. http://zeolites.cqe.northwestern.edu/Music/nonorthorhombic.pdf.

[19] S.Y. Qiu, CZ Wang, KM Ho, and CT Chan. Tight-binding molecular dynamics with linear system-size scaling. Journal of Physics: Condensed Matter, 6:9153, 1994.

[20] X.P. Li, RW Nunes, and D. Vanderbilt. Density-matrix electronic-structure method with linear system-size scaling. Physical Review B, 47(16):10891, 1993.

[21] M.C. Payne, M.P. Teter, D.C. Allan, TA Arias, and JD Joannopoulos. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. Rev. Mod. Phys., 64(4):1045–1097, 1992.

[22] M.J. Mehl, B.M. Klein, and D.A. Papaconstantopoulos. First-principles calculations of elastic properties of metals. Intermetalic Compounds: Principles and Practice, 1, 1994.

[23] Jmol: an open-source java viewer for chemical structures in 3D. http://www.jmol.org/.

[24] J. Rossato, RJ Baierle, A. Fazzio, and R. Mota. Vacancy formation process in carbon nanotubes: First-principles approach. Nano Lett., 5(1):197–200, 2005.

[25] AV Krasheninnikov, PO Lehtinen, AS Foster, and RM Nieminen. Bending the rules: Contrasting vacancy energetics and migration in graphite and carbon nanotubes. Chem. Phys. Lett., 418(1):132–136, 2006.

[26] J. Bernholc, A. Antonelli, TM Del Sole, Y. Bar-Yam, and ST Pantelides. Mechanism of self-diffusion in diamond. Phys. Rev. Lett., 61(23):2689–2692, 1988.

[27] JC Bourgoin. An experimental estimation of the vacancy formation energy in diamond. Radiation effects, 79(1-4):235–239, 1983.

[28] Nicolas Mounet and Nicola Marzari. First-principles determination of the structural, vibrational and thermodynamic properties of diamond, graphite, and derivatives. Phys. Rev. B, 71:205214, May 2005.

[29] Marvin L. Cohen. Calculation of bulk moduli of diamond and zinc-blende solids. Phys. Rev. B, 32:7988–7991, Dec 1985.

[30] Luc Henrard, E. Hernández, Patrick Bernier, and Angel Rubio. van der waals interaction in nanotube bundles: Consequences on vibrational modes. Phys. Rev. B, 60:R8521–R8524, Sep 1999.

[31] FD Murnaghan. The compressibility of media under extreme pressures. Proceedings of the national academy of sciences of the United States of America, 30(9):244, 1944.

[32] R. Maranto and A. Gresham. Using "world series shares" to fight free riding in group projects. PS: Political Science and Politics, 31(4):789–791, 1998.

[33] G. Gibbs. The assessment of group work: lessons from the literature. URL: http://www. brookes. ac. uk/aske/documents/Brookes% 20groupwork% 20Gibbs% 20Dec, 200(9), 2009.