Thermal Rectification in Graphene based Structures using the Large-scale Atomic/Molecular Massively Parallel Simulator "LAMMPS"

W. Dobbenberg*, R. Wiltvank, T.P.E. Swoboda, A. Jarray, M. Munoz Rojo

University of Twente, Faculty of Engineering Technology, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands
**corresponding author w.dobbenberg@student.utwente.nl*

ABSTRACT: In this paper, the thermal rectification effects in graphene will be examined using the Large-scale Atomic/Molecular Massively Parallel Simulator "LAMMPS". The goal is to obtain recitification in monolayer graphene samples and compare them to other works. Two methods have been analysed in combination with the Tersoff and Airebo potential. Namely, the langevin thermostat method and the enhanced heat exchange algorithm. The results show that Tersoff is able to accurately predict the thermal conductivity of pristine graphene. However, Tersoff was incapable of correctly simulating single vacancy defects. On the contrary, Airebo accurately simulated vacancy defects but showed a relatively low thermal conductivity. Consequently, to accurately measure the thermal recitification ratio of graphene, further improvements should be made to either the Tersoff or Airebo to fix their shortcomings.

Key words: Thermal-rectifier; Graphene; Molecular dynamics; Langevin; Enhanced heat exchange algorithm

1 INTRODUCTION

Thermal rectification involves an asymmetry in the heat transfer through a lattice or ribbon. This implies that the heat transfer propagates easier in one direction and is inhibited in the opposite direction. Graphene, a single-layer carbon honeycomb structure, is a material, commonly used for thermal rectification research. The experimental study of Wang et al.,[1] is one of many examples. Graphene has shown to have a high thermal conductivity in both theoretical and experimental studies [2, 1].
On the other hand, simulating the thermal properties of graphene can be rather difficult and time extensive. Programs such as the Large-scale Atomic/Molecular Massively Parallel Simulator ("LAMMPS")[3] can still require many days in order to finish a single simulation. In addition, these simulations have a large dependency on computer hardware, and initial conditions of the simulation.
In this paper, the thermal rectification effects in graphene will be examined using the previously mentioned program "LAMMPS". The main objective is to obtain rectification in monolayer graphene samples and compare them to previous works [1, 4, 5, 6]. Rectification will be obtained through the use of two different non-equilibrium molecular dynamics methods. The first method is known as the langevin thermostat method. The second is a more direct approach falling into the reverse non-equilibrium molecular dynamics category. Namely, the enhanced heat exchange algorithm developed by Wirnsberger et al.[7] The two "NEMD" methods will combined with both the Tersoff [8] and Airebo [9] potentials. Likewise, the potentials will be compared.

2 MODEL

In order to study the thermal properties of graphene and achieve thermal rectification, a simulation model has been constructed around the previously mentioned non-equilibrium molecular dynamics methods. The model describes the basic principles behind the two "NEMD" methods and contains the parameters to the nanoribbon data file used in the simulations.

2.1 Graphene nanoribbon papameters

"LAMMPS" possesses several tools to produce material structures to be as input in the molecular dynamics simulations. However, to gain greater control over the material structure, an alternative program named Matlab is used to write the atom coordinates to a $.dat$ file. Subsequently, the "LAMMPS" executable script loads the $.dat$ file when executed. The Matlab code generates a 2-dimensional graphene nanoribbon (example shown in Figure 1 and 3) of 450 nm by 5 nm which is referred to as L_x and L_y respectively. On both ends of the nanoribbon a thermostat region is defined in the "LAMMPS" executable. The

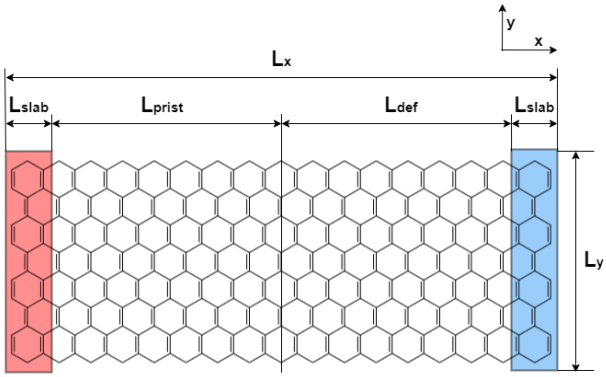


Figure 1: Geometry example of the graphene nanoribbon used in the LAMMPS simulations. Only a small proportion of the strip is shown (not scaled). The two thermostats regions are shown by the two-coloured boxes.

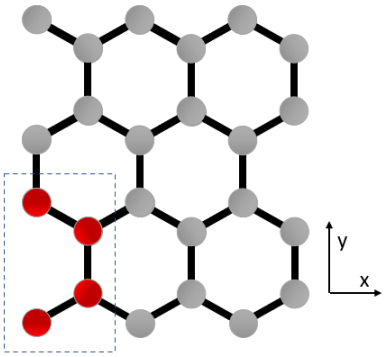


Figure 2: A zoomed in portion of the monolayer graphene with the unit cell in red. The example consists of three (x direction) times two (y direction) unit cells.

thermostat regions are 5 nm wide in the x direction and will be referred to as L_{slab}. When a asymmetrical nanoribbon is used to analyse the rectification effects in graphene, the asymmetry is achieved by applying vacancy defects to one half of the nanoribbon, excluding the thermostat regions. This length of the pristine and defected regions are defined by the terms L_{pris} and L_{def} respectively. Generally, the direction of heat flow is defined from left to right. Which will be referred to this as the normal direction. To obtain the thermal rectification via the "NEMD" methods, it is required to run the simulation a multiple of two times. The first run will have the heat flow go from left to right. The second will have its thermostats swapped to create a heat flow from right to left. This second direction of heat flow will be referred to as the inverse direction. The nanoribbon is generated by copying a graphene unit cell, consisting of four atoms, replicated a certain number of times in both the x and y direction to match the length of the L_x and L_y parameters. The unit cell shown in Figure 2 leads to a zigzag and armchair orientation along the x and y axis respectively. Two different sets of boundary conditions have been used in this project. These sets consist of either fixed (shrink wrapped) and periodic boundary conditions. The periodic boundary condition allows for free movement and interaction across the simulation boundary while the fixed boundary condition prohibits this behaviour. Both the x and z directions remain fixed for all simulation runs. The y direction has the periodic boundary condition applied for the pristine graphene samples and the fixed condition for the asymmetrical graphene samples. In both the x and z direction a clearance of 20Å is used between the most outer atom and the fixed boundary. This allows for a sufficient movement of the carbon atom while preventing them from crossing the fixed boundary. The periodic y direction has a clearance equal to the bond distance between carbon atoms divided by two. The bond distance depends on the potential used during the simulation and is described in Table 1. When the fixed boundary condition is applied to the y direction, the clearance is the same as for the x and z dimensions.

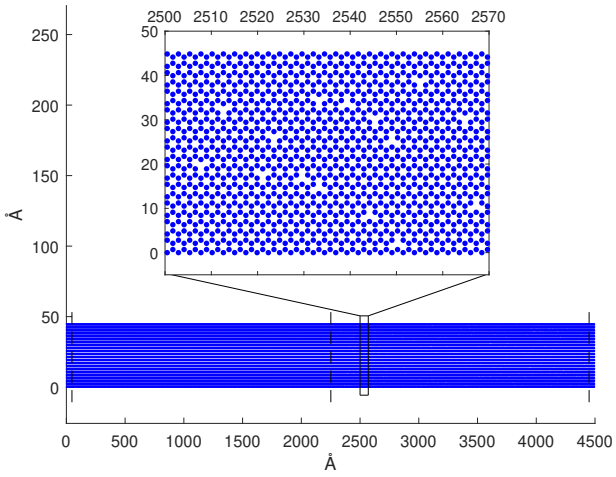


Figure 3: Plot of the generated graphene sheet using the Matlab generation script. The script creates single-vacancy defects on the right hand side of the nanoribbon. The vacancies account for 1% of the total atoms in the right hand side of the nanoribbon and are randomly distributed.

2.2 LAMMPS

All simulations performed in "LAMMPS" are ran for a total length of 1 ns with a time step size equal to 0.5 fs. In order to describe the interactions between atoms an interatomic potential is used. "LAMMPS" possesses several build in potentials. The potentials examined in this paper are shown in Table 1. Afterwards, the graphene structure is energy minimized using "LAMMPS" build in command.

Potential	dt	CPU per dt	$d_{bond}/2$
SiC.Tersoff	1 fs	$3.74e^{-6}$	0.7275 Å
CH.Airebo	0.5 fs	$3.25e^{-5}$	0.7 Å
CH.Rebo	0.5 fs	$3.18e^{-6}$	0.7 Å*

Table 1: The table shows the per potential recommended minimal timestep size. The CPU performance per time-step by LAMMPS [10] is listed. finally, it shows the resulting interatomic radius between two carbon atoms after the minimisation step in LAMMPS. All potentials in the table output in LAMMPS "metal" units and require the minimal data style "atomic" for the $.dat$ file. *Estimation based on the Airebo potential.

By iteratively adjusting the atom positions, the command approaches a local minimum in the system potential energy. Hence, slightly contracting or stretching the graphene nanoribbon. The minimising method used is the conjugate gradient (CG) algorithm.
After minimisation, the atoms in the thermostat regions are constrained by the "spring/self" command. The command locks each atom to its original location at the time of execution. The spring constant used is equal to 0.05 eV/Å^2. With the constraints in place, the first equilibrium run performing time-integration on Nose-Hoove [11, 12] style non-Hamiltonian equations of motion, is performed to gain an average system temperature of 300K.
When removing the part responsible for adding or removing energy into the system, the Nose-Hoove integration method reduces to the velocity Verlet integration method [13]. Verlet integration is a numerical method of integrating Newtons equation of motion. Newtons equation of motion for individual particles can be described as:

$$\ddot{x}(t) = m \cdot F(\mathbf{x}(t)) = -m\nabla_{\vec{x}} V(\mathbf{x}(t)) \tag{1}$$

where:

- $\ddot{x}$ being the acceleration of the particle,
- m the mass of a particle,
- F the ensemble of forces influencing the particle,
- $\mathbf{x}(t)$ the position vector of the particle at time t,
- V the scalar potential function.

During the velocity Verlet integration, the energy flow through the nanoribbon is induced by either one or two thermostatting methods. Both methods are based on Fourier's law. [14]

$$J_x = -\kappa \frac{\delta T}{\delta x} = \frac{\Delta Q}{A \Delta t} \tag{2}$$

where:

- J_x is the heat flux in direction x in terms of $\frac{\Delta Q}{A\Delta t}$,
- κ the thermal conductivity,
- $\frac{\delta T}{\delta x}$ the temperature gradient in direction x.

This equation can be rewritten to obtain an equation for the thermal conductivity of the material:

$$-\kappa = \frac{\Delta Q}{\Delta t A} \cdot \frac{\Delta x}{\Delta T} \tag{3}$$

Due to asymmetries in the thermal conductivity across the asymmetric nanoribbons, the thermal conductivity will be different when switching the thermostats. Originally, the term used for determining thermal rectification, was based on the difference in heat flux [15]. However, since the enhanced heat exchange algorithm uses a constant heat current, would always default to zero rectification. Hence we concluded, that this is an inaccurate way to measure and compare the thermal rectification between both the langevin and the enhanced heat exchange algorithm methods. Instead, the thermal conductivity in Equation 3 is used for determining the thermal rectification ratio. It assumed that the heat current remains constant for the enhanced heat exchange algorithm and that the temperature remains constant for the langevin method. This gives the thermal rectification ratio described as:

$$rr = \frac{\kappa_{\xleftarrow{x^+}} - \kappa_{\xrightarrow{x^-}}}{\kappa_{\xrightarrow{x^-}}} \cdot 100\% \tag{4}$$

2.3 NEMD Langevin thermostat method

Non-equilibrium molecular dynamics is a technique based on the concept of controlling the temperature of two different reservoirs to induce heat flow between them. In this paper, temperature control is achieved through the use of the Langevin equations. The Langevin method modifies the force value used in the velocity Verlet integration to add or remove energy into the reservoir. The force value can be described by three components [16]:

$$F = F_c + F_f + F_r \tag{5}$$

$$F_f = -\frac{m}{d} v \tag{6}$$

$$F_r \propto \sqrt{\frac{k_B T m}{\Delta t \cdot d}} \tag{7}$$

where:

- F_c is the conservative force computed via the interparticle interactions defined by the simulation potential.
- F_f the frictional drag or viscous damping component that is proportional to the atom's velocity by a factor

$\frac{m}{d}$. With m equal to the mass of the atom and d being the damping coefficient for which 0.5 is used in this paper.

- F_r a force proportional to the solvent of particle's randomly bumping into each other at a temperature T, where k_B is the Boltmann constant, m the mass of the atom, Δt the timestep size, and d the damping factor.

Consequently, the reservoir will approach temperature T, defined in the third force component, over a time inversely proportional to $\Delta t \cdot d$.

2.4 RNEMD enhanced heat exchange algorithm

Instead of controlling the temperature of the reservoirs, the reverse non-equilibrium molecular dynamics method achieves heat flux through the nanoribbon by adding and subtracting energy at a constant rate from the reservoirs. The equation controlling this progress is the enhanced heat exchange algorithm by Wirnsberger et al. [7] As the name suggests, it is a second generation of the heat exchange algorithm by Ikeshoji et al. [17] The equations scale the velocity output of the velocity Verlet integration to add or subtract the energy to the reservoir. In Equation 8, the new translational velocity ($\mathbf{v}_{i,new}$) is determined by scaling the old translational velocity ($\mathbf{v}_{i,old}$) and non-translational velocity ($\mathbf{v}_{j,old}$). Where, E_k is the initial kinetic energy of the atom shown in Equation 9 with m equal to the corresponding mass.

$$\mathbf{v}_{i,new} = \mathbf{v}_{i,old}\sqrt{1+\frac{\Delta Q}{E_k}} + \left(1-\sqrt{1+\frac{\Delta Q}{E_k}}\right)\mathbf{v}_{j,old} \quad (8)$$

$$E_k = \frac{1}{2}\sum_i m\mathbf{v}_{i,old}^2 - \frac{1}{2}\sum_j m\mathbf{v}_{j,old}^2 \quad (9)$$

3 DISCUSSION AND RESULTS

The simulation outputs data in $.txt$ file format. Since this is uneasy to the eyes, the ouput data was post-processed using Matlab script.

3.1 Pristine Graphene

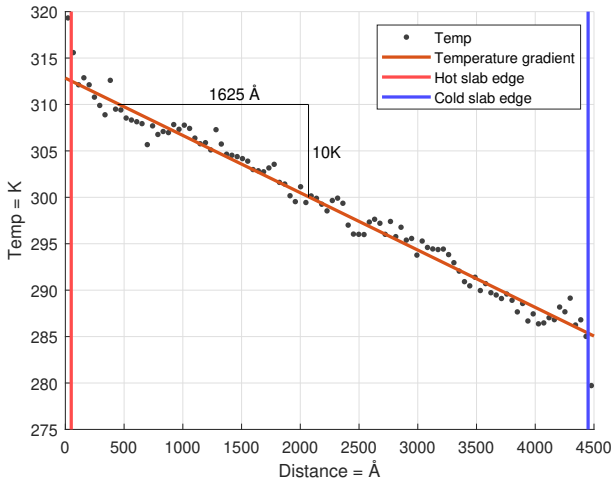
The first molecular dynamics run was performed on pristine graphene using the tersoff potential and langevin thermostat method. The temperature of the reservoirs are set at 300 ± 20 K. The resulting temperature profile in the x direction can be seen in Figure 4(a). The temperature profile assumed a near linear profile in accordance with Fourier's law. An exception can be observed around the temperature reservoirs. This is most likely caused by the damping factor of 0.5. Decreasing the damping factor will increase the effect of the Langevin thermostat force described in Section 2.3.
The heat flux was calculated using "LAMMPS" build in compute method. The resulting heat flux in the positive x direction can be seen in Figure 4(b). The resulting heat flux is not a good measure for determining the thermal conductivity since it is based on the simulation box volume and not on the actual graphene. In addition, the large fluctuations make it difficult to get an accurate average flux value. Instead, the energy accumulation of the langevin thermostats is used as they represent the energy going in and out of the thermostats (Figure 5). The gradient of the accumulated energy is found to be equal to 1.320 eV/ps. The resulting thermal conductivity for pristine graphene with the corresponding Tersoff potential is equal to 1471 W/mK. A reasonable result when comparing it to the findings of Cao [4], with an estimated thermal conductivity of 1600 W/mK for an 500 nm sample, when accounting for the phonon mean free path. The work of Cao shows that the thermal conductivity of graphene approaches an value of 2360 W/mK at 2.8 μm. A value relatively close to the experimental value of 2100 W/mK shown by Wang et al. [1]

3.2 Rectification

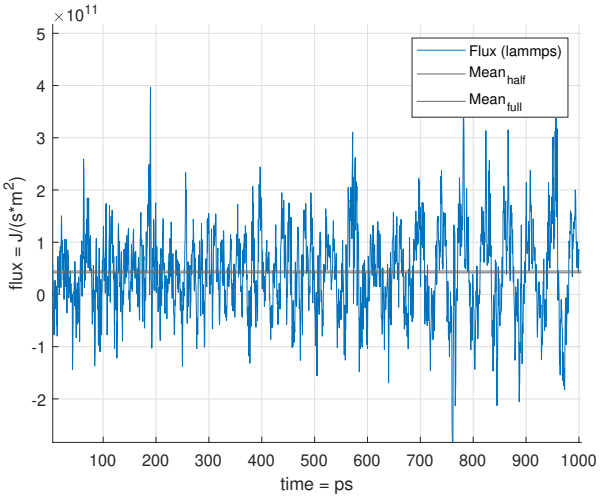
The introduction of single vacancy defects to one side of the graphene nanoribbon, while using the Tersoff potential, caused the overall system energy to increase over time. Consequently, the system temperature increased as well. The effect vacancies have on the temperature can be observed in Figure 6. To counteract this phenomenon, the potential was changed to the Airebo potential for all of the following simulation runs.

3.2.a NEMD

Using the Airebo potential fixed the previously mentioned energy accumulation problem. The resulting temperature profile for both the positive and negative x direction can be seen in Figure 8. The average system temperature increased or decreased, depending on the direction of the flow. The corresponding heat current is obtained using the energy accumulation in the thermostat regions. From Figure 7(a) and (b), the slope is used as an accurate measure for the energy going in and out of the reservoirs. Figure 7(a) has reached steady state and has a slope of 0.0830 eV/ps. Figure 7(b) has not yet reached steady state and thus, can not be used for an accurate value. Nevertheless the value is expected to be in between the two current slopes with the values 0.11 and 0.0435 eV/ps. For now is assumed that the actual heat current is exactly in between the two limits. The resulting rectification ratio is calculated using Equation 4 and is found to be 8.21%.



(a)



(b)

Figure 4: (a) Temperature profile for a fully pristine graphene sample of 450 nm by 5 nm in x and y direction respectively using the Tersoff potential. (b) The computed heat flux data points are measured per 1000 time steps. The boundary conditions are fixed, periodic, and fixed in the x,y,and z direction respectively. The total simulation time is 1 ns at an average temperature of 300 K.

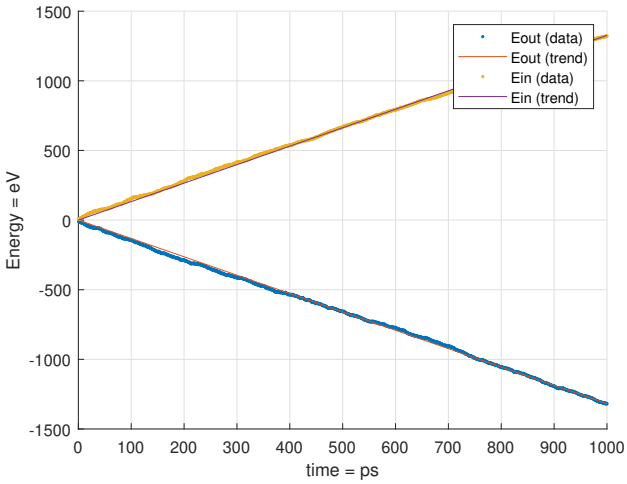


Figure 5: Energy accumulation of the langevin thermostats for the fully pristine graphene sample of 450 nm by 5 nm in x and y direction respectively using the Tersoff potential. The values show that no energy leakage is present as the inlet energy is equal to the outlet energy.

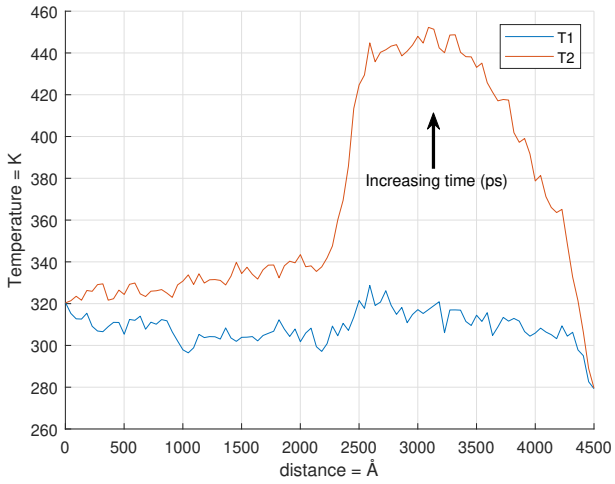
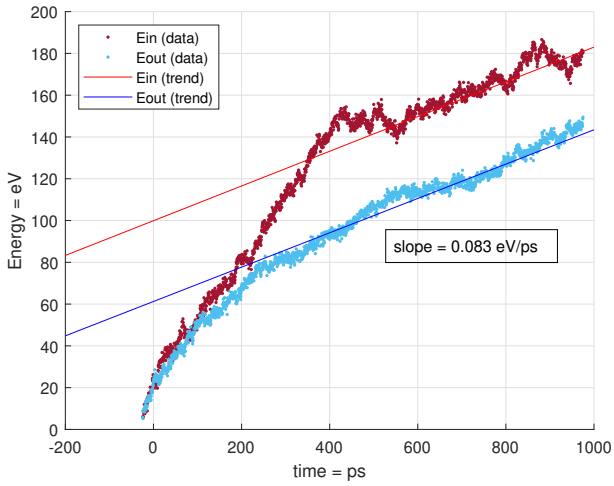


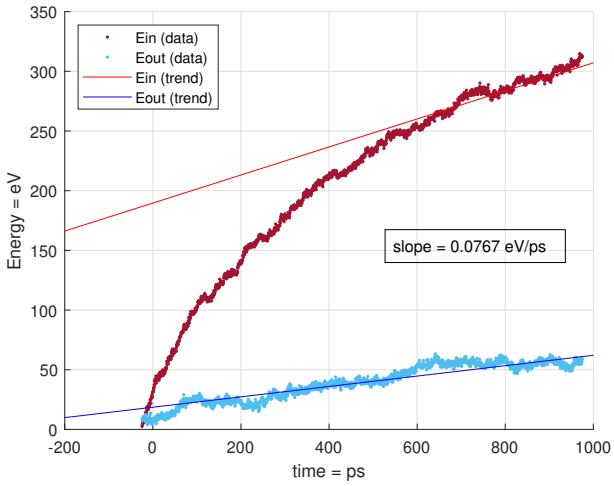
Figure 6: Temperature profile in x direction for the asymmetric graphene sample with 1% single vacancy defects. The single vacancy defects are randomly distributed among the right hand half of the nanoribon. The two profiles are taken at different timesteps along the simulation.

3.2.b RNEMD

Unlike the "NEMD" method, the average system temperature of the "RNEMD" enhanced heat exchange algorithm method remains around the original system temperature of 300 K. This can be observed in Figure 9. Additionally both the "NEMD" and "RNEMD" cross their corresponding average temperatures at around 61.1% of the total length of the nanoribbon. The energy going in and out of the thermostat regions is defined to be 0.2 eV/ps in both the normal and inverse run. This corresponds to a thermal conductivity of 219.6 and 202.4 W/mK, respectively. The final rectification ratio was found to be 8.56%, similar to the result of the "NEMD" run. Where the "NEMD" uses the accumulated energy in the thermostats to determine the thermal conductivity, the "RNEMD" method uses the temperature of the thermostats. In Figure 10(a) and (b), the temperature of both thermostats are plotted against time. Similar to the "NEMD" method, it is observed that steady state is



(a)



(b)

Figure 7: Energy accumulation of the langevin thermostats for the asymmetric graphene sample with 1% single vacancy defects. With (a) having the flux flow from left to right, and (b) from right to left. Both the energy in and out have been plotted as their absolute values.

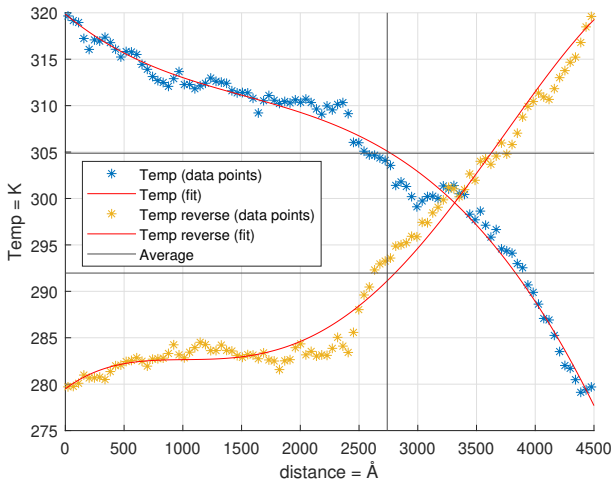


Figure 8: Temperature profile in x direction for the asymmetric graphene sample with 1% single vacancy defects for both the left to right and right to left heat flux directions in the "NEMD" run.

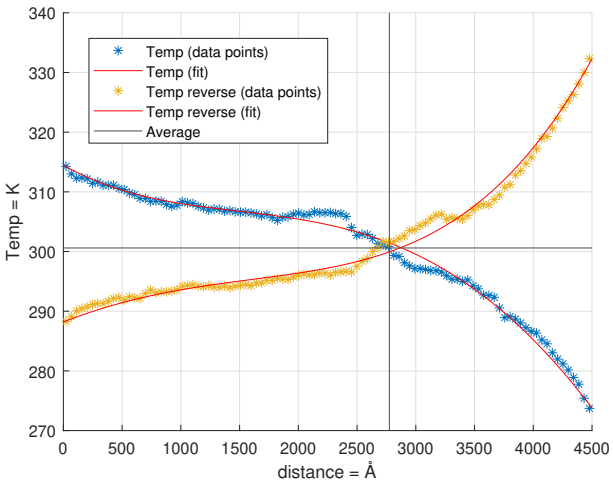


Figure 9: Temperature profile in x direction for the asymmetric graphene sample with 1% single vacancy defects for both the left to right and right to left heat flux directions in the "RNEMD" run.

not yet achieved. The study by Wang et al., [15] showed that, similar to the thermal conductivity of graphene, thermal rectification is also length dependent. They attribute this phenomenon to the phonon mean free path. Their results show that for single vacancy defects they can obtain a thermal rectification ratio of up to 80% for a 14 nm long sample. The rectification ratio decreases towards 3 to 5% with a logarithmic curve for increasing length. The 3 to 5% is achieved at 775 nm.

4 SUMMARY AND CONCLUSION

To summarise, all simulation were performed on a monolayer graphene nanoribbon of size 5 by 450 nm at a system temperature of 300 K. Each simulation has ran at 0.5 fs per timestep for a total of 1 ns. The results showed a peak thermal conductivity of 1471 W/mK for a fully pristine graphene sample (zigzag orientation) using the Tersoff potential and langevin thermostat method. The Tersoff potential however, did not accurately simulate single vacancy defects present in the asymmetric graphene nanoribbon.

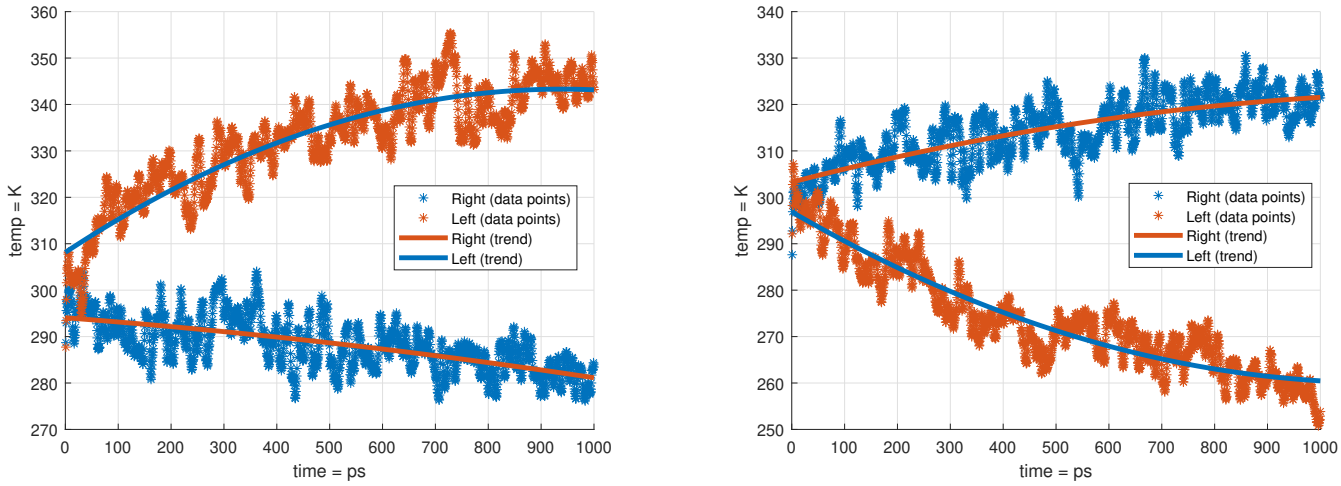


Figure 10: Thermostat temperatures over time for the asymmetric graphene sample with 1% single vacancy defects in the "RNEMD" run. With (a) the heat flux flowing from left to right, and (b) flowing from right to left.

Method	Defects	Potential	rr	κ_{max}	$E_{in/out,max}$	$B_{conditions,x,y,z}$
eHEXa	Singe-Vacancy	Airebo	8.56 %	219.68 W/mK	0.20 eV/ps	fs fs fs
Langevin	Singe-Vacancy	Airebo	8.21 %	92.45 W/mK	$8.22 \cdot 10^{-2}$ eV/ps	fs fs fs
Langevin	none	Tersoff	NaN	1471.28 W/mK	1.32 eV/ps	fs p fs

Table 2: A summation of the results for the thermal rectification ratio using the two different methods in combination with several settings. A more in-depth explanation to the boundary conditions can be found on the "LAMMPS" website. [3]

However, when applying the Airebo potential, a thermal rectification ratio of 8 to 9% has been achieved (Table 2). The drawback being that it also resulted in a lower thermal conductivity that can not only be attributed to the difference between the pristine and asymmetric graphene nanoribbon. And finally, a rise or fall in the system temperature was observed for the asymmetric graphene nanoribbon using the Langevin thermostat method.

The thermal conductivity for the pristine graphene with the Tersoff potential falls well within range of experimentally found results [1] when accounting for the phonon mean free path. We assume that the thermal conductivity will continue to increase similar to the results of Cao. [4] Secondly, we conclude that the Tersoff potential in its current state was not able to properly handle vacancies introduced in the graphene nanoribbon. In contrast to the Tersoff potential, the Airebo potential greatly underestimates the thermal conductivity of graphene. However, the Airebo potential proved better in handeling the single vacancy defects introduced. The resulting thermal rectification ratio showed to be quite similar to the results of Wang et al. [15], Similar to the thermal conductivity slight difference in rectification can be attributed to the phonon mean free path and is thus reasonable.

When comparing the two non-equilibrium molecular dynamics methods we conclude that both methods are appropriate for different applications. The reverse non-equilibrium molecular dynamics method is considered to be more accurate when determining the thermal rectification ratio for a set temperature independent of heat current. However, the non-equilibrium molecular dynamics method is considered to be more useful for real life implementation due to its controllable thermostats mimicking real life applications.

5 RECOMMENDATIONS

Due to the hardware limitations and a solid deadline, the simulations performed were only 1 ns long. It is highly recommended to run the simulations for a longer amount of time. As it is quite clear that the simulations have not fully reached steady state. Except for this oversight, we highly recommended to research the behaviour surrounding the Tersoff potential in combination with vacancy defects. We recommend to distribution the vacancies more evenly. As we consider the possibility that neighbouring vacancies resulting from the random distribute might be the cause. A likely second possibility would be the absence of hydrogen atoms binding to the carbon atoms

surrounding the vacancy. The third and final cause could lie in the data file used for the simulation as it is of the type "atomic". Including parameters like bonds and dihedrals might change the result.

Another recommendation would be the Green-Kubo method. Green-Kubo can be used to estimate the thermal conductivity of a material. It does however, still suffer from the effect of the phonon mean free path. If the resulting thermal conductivity's of the Green-Kubo formulation are scaled to match the compensate for the phonon mean free path, the temperature profile could be plotted using the inverse of Equation 2. Similar to the reverse non-equilibrium molecular dynamics model, the heat current should be assumed constant. Also, one of the two thermostat regions should be assumed constant. When these two assumptions are applied, the temperature profile along the length of a nanoribbon can be calculated through the use of integration of the inverse Equation 2.

REFERENCES

1. Haidong Wang, Shiqian Hu, Koji Takahashi, Xing Zhang, Hiroshi Takamatsu, and Jie Chen, Experimental study of thermal rectification in suspended monolayer graphene, *Nature Communications*, (2017), 8.
2. Eric Pop, Vikas Varshney, and Ajit K. Roy, Thermal properties of graphene: Fundamentals and applications, *MRS Bulletin*, (2012), 37(12).
3. S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, documentation, `https://lammps.sandia.gov/doc`, (1995).
4. Ajing Cao, Molecular dynamics simulation study on heat transport in monolayer graphene sheet with various geometries, *Journal of Applied Physics*, (2012), 111(8).
5. Jiuning Hu, Xiulin Ruan, and Yong P. Chen, Thermal conductivity and thermal rectification in graphene nanoribbons: A molecular dynamics study, *Nano Letters*, (2009), 9(7).
6. Xiangfan Xu, Luiz F.C. Pereira, Yu Wang, Jing Wu, Kaiwen Zhang, Xiangming Zhao, Sukang Bae, Cong Tinh Bui, Rongguo Xie, John T.L. Thong, Byung Hee Hong, Kian Ping Loh, Davide Donadio, Baowen Li, and Barbaros Özyilmaz, Length-dependent thermal conductivity in suspended single-layer graphene, *Nature Communications*, (2014), 5.
7. P. Wirnsberger, D. Frenkel, and C. Dellago, An enhanced version of the heat exchange algorithm with excellent energy conservation properties, *Journal of Chemical Physics*, (2015), 143(12).
8. Sayyed Jalil Mahdizadeh and Golnoosh Akhlamadi, Optimized Tersoff empirical potential for germanene, *Journal of Molecular Graphics and Modelling*, (2017), 72.
9. Gurjot Dhaliwal, Prasanth B. Nair, and Chandra Veer Singh, Uncertainty analysis and estimation of robust AIREBO parameters for graphene, *Carbon*, (2019), 142.
10. S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, Potential Performance, `https://lammps.sandia.gov/bench.html`, (1995).
11. Shuichi Nosé, A unified formulation of the constant temperature molecular dynamics methods, *The Journal of Chemical Physics*, (1984), 81(1).
12. William G. Hoover, Canonical dynamics: Equilibrium phase-space distributions, *Physical Review A*, (1985), 31(3).
13. Anna Lincoln, Arnero, Beasticles, BenFrantzDale, Berland, Charles Matthews, and et al. Colinb, Verlet integration, *Creative Commons Attribution, https://www.saylor.org/site/wp-content/uploads/2011/06/MA221-6.1.pdf*, (1967).
14. I. Shih Liu, On Fourier's law of heat conduction, *Continuum Mechanics and Thermodynamics*, (1990), 2(4).
15. Yan Wang, Siyu Chen, and Xiulin Ruan, Tunable thermal rectification in graphene nanoribbons through defect engineering: A molecular dynamics study, *Applied Physics Letters*, (2012), 100(16).
16. P. Mazur, On the theory of brownian motion, *Physica*, (1959), 25(1-6).
17. Tamio Ikeshoji and Bjørn Hafskjold, Non-equilibrium molecular dynamics calculation of heat conduction in liquid and through liquid-gas interface, *Molecular Physics*, (1994), 81(2).

A MATLAB CODE FOR GENERATING THE GRAPHENE SHEET

```
clc
clear all
close all

format shortG

%% ====== initiate ====== %%

filename = ...
    'C:\Users\wodo9\Documents\LAMMPS\Thermal_Conductivity_Reservoirs\Version_20_ehexa_prist\
%filename = 'D:\graphene_no_bonds.dat';
writeresults = false;
showresults = true;
FillVacancies = false;
seed = 52525;
L = 4500;

%% ====== Defect Settings ====== %%

a         = 0.01 % defect atom percentage
vac_size  = 1    % vacancy size
def_width = 0.5  % percent

%% ====== Atoms ====== %%

%      mass     r_bond
C   = [12.0107  0.7008];
H   = [1.00784  0.5];
Si  = [28.0855  0.7873];

%% ====== Unit Cells ====== %%

Unit_pris = [ 0.0              , 0.0                          , 0.0 ;
              cos(pi/6)*2*C(2) , sin(pi/6)*2*C(2)             , 0.0 ;
              cos(pi/6)*2*C(2) , sin(pi/6)*2*C(2) + 2*C(2)    , 0.0 ;
              0.0              , sin(pi/6)*4*C(2) + 2*C(2)    , 0.0 ];

%% ====== Sheet Dimensions ====== %%

xyz = [L 50 3.45]; % angstrom

Units_x = floor(xyz(1)/(2*max(Unit_pris(:,1))))
Units_y = floor(xyz(2)/(2*C(2) + max(Unit_pris(:,2))))

x_lim = [ (-cos(pi/6)*C(2))-2 , (4*(cos(pi/6)*C(2))*Units_x)-(cos(pi/6)*C(2))+2]'
y_lim = [ (-C(2)) , (((sin(pi/6)*4*C(2) + 4*C(2))*Units_y)-C(2))]'
z_lim = [ (-3.45/2)-20 , (3.45/2)+20 ]'

Res_width = 50;  % angstrom

%% ====== Generate Atom Positions ====== %%

% Total number of atoms
N_atoms = 4*Units_x*Units_y

% Calculate the coordinates of the atoms in the layer
```

```
coords = zeros(N_atoms,3);
id = 0;
for ix=0:(Units_x-1)
    for iy=0:(Units_y-1)
        for iatom=1:length(Unit_pris)
            id = id + 1;
            coords(id,:) = ...
                Unit_pris(iatom,:)+[ix*(2*max(Unit_pris(:,1))),iy*(2*C(2) + ...
                max(Unit_pris(:,2))),0];
        end
    end
end

%% ====== Define Regions ====== %%

r_hot     = find(coords(:,1) ≤ Res_width);
r_cold    = find(coords(:,1) ≥ max(coords(:,1)) - Res_width);
r_between = find( (coords(:,1) ≥ Res_width) & (coords(:,1) ≤ max(coords(:,1)) - ...
    Res_width) );
r_pris    = find( (coords(:,1) ≥ Res_width) & (coords(:,1) ≤ ((max(coords(:,1)) + ...
    min(coords(:,1)))*(1-def_width))) );
r_def     = find( (coords(:,1) ≤ max(coords(:,1)) - Res_width) & (coords(:,1) ≥ ...
    ((max(coords(:,1)) + min(coords(:,1)))*(1-def_width))) );

group = table([min(r_hot) max(r_hot)]',[min(r_cold) max(r_cold)]',[min(r_between) ...
    max(r_between)]',[min(r_pris) max(r_pris)]',[min(r_def) max(r_def)]');
group.Properties.VariableNames = {'Hot','Cold','Between','pristine','defected'}

length(r_hot)

%% ====== Search & Destroy ====== %%

if seed ≠ 0
    rng(seed)
end
randomi = randi([1 1000],length(r_def),1);

ind_def = find(randomi ≤ round(a*1000));
for j = 1:length(ind_def)
    ind_def(j) = r_def(ind_def(j));
end
def_percentage = length(ind_def)/length(r_def)

row = Units_y*4;
for h = 1:5
    i = 0;
    for q = 1:length(ind_def)
        error1 = 0;
        move_count = 0;
        while q ≠ i
            i = i + 1;
            state = [0 0 0];
            %% Check Unit_cell position
            if mod(ind_def(i),4) == 1
                state(1) = 1;
            elseif mod(ind_def(i),4) == 2
                state(1) = 2;
            elseif mod(ind_def(i),4) == 3
                state(1) = 3;
            elseif mod(ind_def(i),4) == 0
```

```
110                 state(1) = 4;
111             end
112             %% Check if on top or bottom edge
113             if mod(ind_def(i),Units_y*4) == 0
114                 state(2) = 1;
115             elseif mod(ind_def(i),Units_y*4) == Units_y*4-1
116                 state(2) = 2;
117             elseif mod(ind_def(i),Units_y*4) == 1
118                 state(2) = 3;
119             elseif mod(ind_def(i),Units_y*4) == 2
120                 state(2) = 4;
121             end
122             %% Check if on left or right of defected region
123             if ind_def(i) ≤ min(r_def) + Units_y*4
124                 state(3) = 1;
125             elseif ind_def(i) ≥ max(r_def) - Units_y*4
126                 state(3) = 2;
127             end
128             %% state to checkable coords
129             if state(1) == 1
130                 check_coords = [(-1) (-2) (-row-2) (+row-1); (+1) (+2) (+row) ...
                        (+3); (-row+1) (-row) (-row+2) (-row-1)];
131             elseif state(1) == 2
132                 check_coords = [(-1) (-2) (-row) (-3); (+1) (+2) (+row+2) ...
                        (-row+1); (+row-1) (+row) (+row-2) (+row+1)];
133             elseif state(1) == 3
134                 check_coords = [(-1) (-2) (+row-2) (+3); (+1) (+2) (-row) ...
                        (-row-1); (+row+1) (+row) (+row+2) (+row-1)];
135             elseif state(1) == 4
136                 check_coords = [(-1) (-2) (+row) (-3); (+1) (+2) (-row+2) ...
                        (+row+1); (-row-1) (-row) (-row-2) (-row+1)];
137             end
138
139             if state(2) == 1
140                 check_coords(2,:) = check_coords(2,:)-row;
141                 check_coords(3,4) = check_coords(3,4)-row;
142             elseif state(2) == 2
143                 check_coords(2,2) = check_coords(2,2)-row;
144                 check_coords(3,3) = check_coords(3,3)-row;
145             elseif state(2) == 3
146                 check_coords(1,:) = check_coords(1,:)+row;
147                 check_coords(3,4) = check_coords(3,4)+row;
148             elseif state(2) == 4
149                 check_coords(1,2) = check_coords(1,2)+row;
150                 check_coords(3,3) = check_coords(3,3)+row;
151             end
152
153             if state(3) == 1
154                 if state(1) == 1
155                     check_coords(3,:) = 0;
156                     check_coords(1,3) = 0;
157                 elseif state(1) == 2
158                     check_coords(1,3) = 0;
159                     check_coords(2,4) = 0;
160                 elseif state(1) == 3
161                     check_coords(2,3) = 0;
162                     check_coords(2,4) = 0;
163                 elseif state(1) == 4
164                     check_coords(3,:) = 0;
165                     check_coords(2,3) = 0;
```

```
166                 end
167             elseif state(3) == 2
168                 if state(1) == 1
169                     check_coords(2,3) = 0;
170                     check_coords(1,4) = 0;
171                 elseif state(1) == 2
172                     check_coords(2,3) = 0;
173                     check_coords(3,:) = 0;
174                 elseif state(1) == 3
175                     check_coords(1,3) = 0;
176                     check_coords(3,:) = 0;
177                 elseif state(1) == 4
178                     check_coords(1,3) = 0;
179                     check_coords(2,4) = 0;
180                 end
181             end
182             %% Perform coord check
183             move_vector = [0 0 0 0; 0 0 0 0; 0 0 0 0];
184             for j = 1:3
185                 for k = 1:4
186                     if check_coords(j,k) ≠ 0
187                         if any((ind_def(i) + check_coords(j,k)) == ind_def(:))
188                             move_vector(j,k) = 0;
189                         else
190                             move_vector(j,k) = 1;
191                         end
192                     else
193                         move_vector(j,k) = 0;
194                     end
195                 end
196             end
197             move_vector;
198             %% Move defect
199             if any(any(not(move_vector == [1 1 1 1; 1 1 1 1; 1 1 1 1])')) == 1 && ...
                    error1 ≠ 1
200                 move_dirt(1) = ismember([1 1 1], move_vector(1,1:3), 'rows');
201                 move_dirt(2) = ismember([1 1 1], move_vector(2,1:3), 'rows');
202                 move_dirt(3) = ismember([1 1 1], move_vector(3,1:3), 'rows');
203                 move_dirt = [move_dirt(1) 2*move_dirt(2) 3*move_dirt(3)];
204                 move_dir = 0;
205                 if move_dirt == [0 0 0]
206                     disp("error2")
207                 else
208                     while move_dir == 0
209                         random_dir = randi([1 3],1);
210                         move_dir = move_dirt(random_dir);
211                     end
212                 end
213
214                 if move_dir(1) == 0
215                     disp("error2")
216                 else
217                     old = ind_def(i);
218                     new = ind_def(i) + check_coords(move_dir(1),1);
219                     if any(ind_def == new)
220                         disp("error3")
221                     end
222                     ind_def(i) = new;
223                     X = sprintf('defect point %d is moved to location %d , the old ...
                        index was %d and has moved back to %d',old,new,i,i - ...
```

```
                        abs(check_coords(move_dir)));
                    %disp(X)
                    i = i - 1;
                end
            end
            move_count = move_count+1;
            if move_count == 1000
                error1 = 1;
                disp("error1")
            end
        end
    end
end

%% remove the ind_def from coords
ind_def = sort(ind_def);
atom_id = ones(length(coords),1);
if FillVacancies == 1
    for i = 1:length(ind_def)
        j = length(ind_def)-i+1;
        if mod(ind_def(j),2) == 0
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)) ...
                (coords(ind_def(j),2)+0.5) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)+0.433) ...
                (coords(ind_def(j),2)-0.25) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)-0.433) ...
                (coords(ind_def(j),2)-0.25) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
        else
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)) ...
                (coords(ind_def(j),2)-0.5) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)+0.433) ...
                (coords(ind_def(j),2)+0.25) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
            coords(length(coords)+1,:) = [(coords(ind_def(j),1)-0.433) ...
                (coords(ind_def(j),2)+0.25) coords(ind_def(j),3)];
            atom_id(length(coords),:) = 2;
        end
        coords(ind_def(j),:) = [];
        atom_id(ind_def(j),:) = [];
    end
else
    for i = 1:length(ind_def)
        j = length(ind_def)-i+1;
        coords(ind_def(j),:) = [];
        atom_id(ind_def(j),:) = [];
    end
end

N_atoms = length(coords)

xlin = linspace(min(x_lim),max(x_lim),101);

%% Show results
if showresults == 1
    Lnorm = sum(atom_id(:) == 1)
    figure(1),clf(1),hold on
```

```
        plot(coords(1:Lnorm,1),coords(1:Lnorm,2),'.b','MarkerSize',30)
        plot(coords(Lnorm+1:end,1),coords(Lnorm+1:end,2),'.r','MarkerSize',20)
        axis equal
        for k = 1:length(xlin)
            if k == 81 || k == 88
                xline(xlin(k),'r')
            else
                xline(xlin(k))
            end
        end
        xline(0.555*(abs(min(x_lim)) + max(x_lim)),'g')
    hold off
end

%% Write results
if writeresults == 1
    fid = fopen(filename,'w');
    fprintf(fid,'graphene d=%g \n\n',1.455);

    fprintf(fid,'%g atoms \n',N_atoms);

    fprintf(fid,'%g atom types \n',1);

    fprintf(fid,'\n');
    fprintf(fid,'%g %g xlo xhi \n',x_lim(1),x_lim(2));
    fprintf(fid,'%g %g ylo yhi \n',y_lim(1),y_lim(2));
    fprintf(fid,'%g %g zlo zhi \n\n',z_lim(1),z_lim(2));

    fprintf(fid,'Masses \n\n');
    fprintf(fid,'%g %g \n',1,C(1));
    %fprintf(fid,'%g %g \n',2,H(1));
    fprintf(fid,'\n');

    fprintf(fid,'Atoms \n\n');
    for i=1:N_atoms
        fprintf(fid,'%g %g %g %g %g \n',i,atom_id(i),coords(i,:));
    end
    fprintf(fid,'\n');

    fprintf(fid,'\n');

    fclose(fid);
end
```

B LAMMPS CODE FOR LANGEVIN RUNS

```
################################################################################
#
#
#  Produced by: Wouter Dobbenberg
#
#
#  Description:
#  ------------
#
#  This file is a LAMMPS input script for carrying out a NEMD simulation using
#  the Tersoff potential field and langevin algorithm.
#
################################################################################

## File Specifications ##
units              metal
dimension          3
boundary           f s s
processors         * * *
atom_style         atomic
read_data          "graphene_no_bonds.dat"

## Run Parameters ##
variable dt   equal  0.0005        # Timestep
variable s    equal  10           # Nevery
variable p    equal  100           # Nrepeat
variable d    equal  $p*$s        # Nfreq

timestep ${dt}
thermo   $d
thermo_modify lost warn

## Simulation Box Specifications ##
variable T            equal 300                                  # Surrounding ...
    Temperature in Kelvin
variable V            equal vol                                  # Volume
variable Lx           equal xhi-xlo                              # Nanoribbon ...
    length (500nm)
variable Ly           equal yhi-ylo                              # Nanoribbon ...
    length (500nm)
variable Lslab        equal 50.0                                 # slab length ...
    (angstrom)
variable clear_x      equal 6.0                                  # clearance in x ...
    direction
variable devide       equal ((${Lx}-(${clear_x}*2))/${Lslab})    # = 10 for 504 Lx

variable Tlo          equal 320
variable Thi          equal 280

## Real to SI convergion ##
variable kB equal 1.3806504e-23                                  # Kolbertz constant
variable eV2J equal 1.602e-19                                     # ElectroVolt to joule
variable A2m equal 1.0e-10                                       # armstrong to Meter?
variable ps2s equal 1.0e-12                                      # picosecond to seconds
variable convert equal ${eV2J}*${eV2J}/${ps2s}/${A2m}            # Convert Factor

```

```
52 ## Potential Field Specifications ##
53 pair_style airebo 3.0 1 0
54 pair_coeff * * CH.airebo C
55 neighbor 2.0 nsq
56 neigh_modify every 1 delay 0 check yes
57
58 # Defining the regions
59     # hot slabs (at 225 to 275)
60     variable xlo1_Thi     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*0
61     variable xhi1_Thi     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*1
62
63         # Region x y z of the hot slabs
64         region  hot1   block  ${xlo1_Thi}  ${xhi1_Thi}   INF INF INF INF
65
66     # cold slabs (at 0 to 25 and 475 to 500)
67     variable xlo1_Tlo     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*1
68     variable xhi1_Tlo     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*0
69
70         # Region x y z of the cold slabs
71         region  cold1  block  ${xlo1_Tlo} ${xhi1_Tlo}  INF INF INF INF
72
73     # Between region 1
74     variable xlo1_T     equal xlo+${clear_x}
75     variable xhi1_T     equal xhi-${clear_x}
76
77         # Region x y z of the cold slabs
78         region  between1  block  ${xlo1_T} ${xhi1_T}  INF INF INF INF
79
80     # freeze left
81     variable xlo_left     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*0
82     variable xhi_left     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*1
83
84         # Region
85         region  fixleft  block  ${xlo_left} ${xhi_left}  INF INF INF INF
86
87     # freeze right
88     variable xlo_right     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*1
89     variable xhi_right     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*0
90
91         # Region
92         region  fixright  block  ${xlo_right} ${xhi_right}  INF INF INF INF
93
94     # left
95     variable xlo_right     equal xlo
96     variable xhi_right     equal xlo+(${Lx}/2)
97
98         # Region
99         region  left  block  ${xlo_right} ${xhi_right}  INF INF INF INF
100
101    # right
102    variable xlo_right     equal xlo+(${Lx}/2)
103    variable xhi_right     equal xhi
104
105        # Region
106        region  right  block  ${xlo_right} ${xhi_right}  INF INF INF INF
107
108 group FreezeLeft region fixleft
109 group FreezeRight region fixright
110 group hot_slab region hot1
111 group cold_slab region cold1
```

```
112
113 group gleft region left
114 group gright region right
115
116 ## Computing Temperature for the groupped regions per timestep
117 compute ThotR hot_slab temp
118 compute TcoldR cold_slab temp
119
120 ## Initial conditions
121 velocity all create $T 4928459 mom yes rot yes dist gaussian
122
123 dump 1d all xyz 1000 dump.graphene.xyz
124
125 # Minimization
126 #fix 1 all box/relax x 0.0 y 0.0 couple none vmax 0.01 nreset 100
127
128 min_style cg
129 minimize 1e-10 1e-10 50000 100000
130 reset_timestep 0
131
132 compute chunks  all chunk/atom bin/1d x lower 0.01 units reduced
133 fix     Tchunks  all ave/chunk 5 200 $d chunks temp file temp.profile ave window 10
134
135 fix CenterLeft  FreezeLeft  spring/self 0.05
136 fix CenterRight FreezeRight spring/self 0.05
137
138 ## NVT run
139 fix 1l gleft nvt temp $T $T 0.5
140 fix 1r gright nvt temp $T $T 0.5
141 run 50000
142 unfix 1l
143 unfix 1r
144
145 ## 2nd Equilibrium run
146 velocity all scale $T
147
148 unfix CenterLeft
149 unfix CenterRight
150
151 fix 1 all nve
152 fix hot11  hot_slab langevin ${Thi} ${Thi} 0.01 59804 zero yes tally yes
153 fix cold11 cold_slab langevin ${Tlo} ${Tlo} 0.01 28785 zero yes tally yes
154
155 fix_modify      hot11 temp ThotR
156 fix_modify      cold11 temp TcoldR
157
158 compute         myKE     all ke/atom
159 compute         myPE     all pe/atom
160 compute         myStress all stress/atom NULL virial
161 compute         flux     all heat/flux myKE myPE myStress
162
163 variable        Tdiff    equal (c_ThotR-c_TcoldR)
164 variable        Jx       equal (c_flux[1]/vol)*(${eV2J}/(${A2m}^2)/${ps2s}) #J/m2/s
165 #variable        tempatom atom  (c_myKE*${eV2J})/(3*${kB})
166
167 fix             ave_Tdiff all ave/time 1 $d $d v_Tdiff start 100000
168 #fix             Tave      all ave/atom 1 $d $d v_tempatom
169 fix             ave_Jx    all ave/time 1 $d $d v_Jx start 100000
170
```

```
171 #Output      #      1    2    3  4  5     6   7   8       9        10   11     12 ...
            13   14
172 thermo_style custom step temp pe ke press pxx pyy c_ThotR c_TcoldR v_Jx f_hot11 ...
       f_cold11 time spcpu
173
174 restart 10000 restart.one restart.two
175
176 # execution
177 run 2000000
178 write_restart Restart.final
```

C LAMMPS CODE FOR EHEXA RUNS

```
1  ################################################################################
2  #
3  #
4  #  Produced by: Wouter Dobbenberg
5  #
6  #
7  #  Description:
8  #  ------------
9  #
10 #  This file is a LAMMPS input script for carrying out a rNEMD simulation using
11 #  the Tersoff potential field and eHEX/a algorithm.
12 #
13 ################################################################################
14
15 ## File Specifications ##
16 units              metal
17 dimension          3
18 boundary           f s s
19 processors         * * *
20 atom_style         atomic
21 read_data          "graphene_no_bonds.dat"
22
23 ## Run Parameters ##
24 variable dt  equal  0.0005        # Timestep
25 variable s   equal  10           # Nevery
26 variable p   equal  100           # Nrepeat
27 variable d   equal  $p*$s        # Nfreq
28
29 timestep ${dt}
30 thermo   $d
31 thermo_modify lost warn
32
33 ## Simulation Box Specifications ##
34 variable T            equal 300                                  # Surrounding ...
      Temperature in Kelvin
35 variable V            equal vol                                  # Volume
36 variable Lx           equal xhi-xlo                              # Nanoribbon ...
      length (500nm)
37 variable Ly           equal yhi-ylo                              # Nanoribbon ...
      length (500nm)
38 variable Lslab        equal 50.0                                 # slab length ...
      (angstrom)
39 variable clear_x      equal 2.0                                  # clearance in x ...
      direction
40 variable devide       equal ((${Lx}-(${clear_x}*2))/${Lslab})    # = 10 for 504 Lx
41
42 variable Tlo          equal 320
43 variable Thi          equal 280
44
45 ## Real to SI convergion ##
46 variable kB equal 1.3806504e-23                                 # Kolbertz constant
47 variable eV2J equal 1.602e-19                                    # ElectroVolt to joule
48 variable A2m equal 1.0e-10                                      # armstrong to Meter?
49 variable ps2s equal 1.0e-12                                     # picosecond to seconds
50 variable convert equal ${eV2J}*${eV2J}/${ps2s}/${A2m}           # Convert Factor
51
```

```
## Potential Field Specifications ##

pair_style airebo 3.0 1 0
pair_coeff * * CH.airebo C
neighbor 2.0 nsq
neigh_modify every 1 delay 0 check yes

# Defining the regions
    # hot slabs (at 225 to 275)
    variable xlo1_Thi     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*0
    variable xhi1_Thi     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*1

        # Region x y z of the hot slabs
        region  hot1   block  ${xlo1_Thi}  ${xhi1_Thi}   INF INF INF INF

    # cold slabs (at 0 to 25 and 475 to 500)
    variable xlo1_Tlo     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*1
    variable xhi1_Tlo     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*0

        # Region x y z of the cold slabs
        region  cold1  block  ${xlo1_Tlo} ${xhi1_Tlo}  INF INF INF INF

    # Between region 1
    variable xlo1_T     equal xlo+${clear_x}
    variable xhi1_T     equal xhi-${clear_x}

        # Region x y z of the cold slabs
        region  between1  block  ${xlo1_T} ${xhi1_T}  INF INF INF INF

    # freeze left
    variable xlo_left     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*0
    variable xhi_left     equal xlo+${clear_x}+((${Lx}-(${clear_x}*2))/${devide})*1

        # Region
        region  fixleft  block  ${xlo_left} ${xhi_left}  INF INF INF INF

    # freeze right
    variable xlo_right     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*1
    variable xhi_right     equal xhi-${clear_x}-((${Lx}-(${clear_x}*2))/${devide})*0

        # Region
        region  fixright  block  ${xlo_right} ${xhi_right}  INF INF INF INF

    # left
    variable xlo_right     equal xlo
    variable xhi_right     equal xlo+(${Lx}/2)

        # Region
        region  left  block  ${xlo_right} ${xhi_right}  INF INF INF INF

    # right
    variable xlo_right     equal xlo+(${Lx}/2)
    variable xhi_right     equal xhi

        # Region
        region  right  block  ${xlo_right} ${xhi_right}  INF INF INF INF

group FreezeLeft region fixleft
group FreezeRight region fixright
group hot_slab region hot1
```

```
112 group cold_slab region cold1
113 group gleft region left
114 group gright region right
115
116 ## Computing Temperature for the groupped regions per timestep
117 compute ThotR hot_slab temp
118 compute TcoldR cold_slab temp
119
120 #delete_atoms overlap 0.1 all all
121
122 ## Initial conditions
123 velocity all create $T 4928459 mom yes rot yes dist gaussian
124
125 dump 1d all xyz 1000 dump.graphene.xyz
126
127 # Minimization
128 min_style cg
129 minimize 1e-13 1e-13 50000 100000
130 reset_timestep 0
131
132 compute chunks  all chunk/atom bin/1d x lower 0.01 units reduced
133 fix     Tchunks  all ave/chunk 5 200 $d chunks temp file temp.profile ave window 10
134
135 fix CenterLeft  FreezeLeft  spring/self 0.05
136 fix CenterRight FreezeRight spring/self 0.05
137
138 fix_modify CenterLeft energy yes
139 fix_modify CenterRight energy yes
140
141 ## NVT run
142 fix 1l gleft nvt temp $T $T 0.5
143 fix 1r gright nvt temp $T $T 0.5
144 run 50000
145 unfix 1l
146 unfix 1r
147
148 ## 2nd Equilibrium run
149 velocity all scale $T
150
151 fix 1 all nve
152 fix hot11  hot_slab ehex 1 -0.2
153 fix cold11 cold_slab ehex 1 0.2
154
155 compute         myKE     all ke/atom
156 compute         myPE     all pe/atom
157 compute         myStress all stress/atom NULL virial
158 compute         flux     all heat/flux myKE myPE myStress
159
160 variable        Tdiff    equal (c_ThotR-c_TcoldR)
161 variable        Jx       equal (c_flux[1]/vol)*(${eV2J}/(${A2m}^2)/${ps2s}) #J/m2/s
162 #variable        tempatom atom  (c_myKE*${eV2J})/(3*${kB})
163
164 fix             ave_Tdiff all ave/time 1 $d $d v_Tdiff start 100000
165 #fix             Tave      all ave/atom 1 $d $d v_tempatom
166 fix             ave_Jx    all ave/time 1 $d $d v_Jx start 100000
167
168 #Output      #      1    2    3  4  5     6   7   8       9        10   11     12 ...
            13   14
169 thermo_style custom step temp pe ke press pxx pyy c_ThotR c_TcoldR v_Jx f_hot11 ...
      f_cold11 time spcpu
```

```
170
171 restart 10000 restart.one restart.two
172
173 # execution
174 run 2000000
175 write_restart Restart.final
```