Studying the Desorption of Krypton from Graphene Using Molecular Dynamics Simulations

Aaron Stromberg

Adsorption is the process in which a particle, known as an adsorbate, adheres to the surface of a substrate. The inverse of this process is desorption, which is when the particles break away from the substrate. This paper attempts to examine the dynamic properties of a system during the desorption process with use of the molecular dynamics software LAMMPS. This information is useful for developing applications such as Adsorbed Natural Gas Tanks and gas separation technologies. The results of these simulations show some encouraging similarities between real-world experiments, opening the door for future experiments of a similar nature.

Introduction

We use the method of molecular dynamics to study the time evolution of dynamic properties of many different systems and processes. In this paper, we examine the process of desorption in a *krypton-graphene system*. In order to simulate desorption, the system first had to be initialized to correctly reflect the adsorbed state. Then, energy had to be added to the adsorbed atoms in order to induce desorption. This energy was added in the form of heat, using Nose-Hoover thermosttating (*1*). The simulations were performed in a two stage process, the *initialization phase* and the *observation phase*, and during both phases computations of dynamic properties

are made, including the temperature of the adsorbate, pressure of the desorbed atoms as they create a vapor above the substrate, adsorbate-substrate interaction energy, and the number of particles adsorbed/desorbed.

The simulations were performed using LAMMPS (Large Scale Atomic/Molecular Massively Parallel Simulator), an open source molecular dynamics package developed by Sandia Labs. Molecular dynamics simulations work by time-evolving Newton's equations of motion in a virtual box called the *simulation box*. By setting the boundaries of the box as *periodic*, we are able replicate this box infinitely in dimensions of our choosing, which allows us to simulate seemingly large systems with much less computational power. Periodic boundary conditions work by remapping any particles that attempt to leave the simulation box through one face to appear as if they have actually just re-entered from the other side. This is an important technique used in molecular dynamics simulations, but many other factors must also be considered when designing a simulation. For example, the set of *units* that the simulation software uses must also be specified. For all the experiments in this paper, *Lennard Jones Reduced Units*, also just called reduced units, are used. When using these units, all quantities are actually unitless. Simple formulas are used to convert from these unitless quantities into any desired unit. They work by redefining various physical quantities to be measured in units of fundamental values like the σ and ϵ values of a substance's Lennard-Jones potential. For example, distances may be measured in units of the σ of Krypton. For a more detailed explanation and a description of the formulas used for converting between reduced and SI units, please see http://lammps.sandia.gov/doc/units.html.

As mentioned earlier, the simulations in this paper were performed in two phases: the *initialization phase* and *observation phase*. During the *initialization phase*, the system is created. This includes the definition of atoms, the creation of the simulation box, and the placement of the atoms within the simulation box. Thermostatting, in order to maintain or increase the

temperature of the system, was also sometimes applied during this phase. This phase is always executed only once, and generates what LAMMPS refers to as a `restart file`. A `restart file` contains almost all the information LAMMPS needs to successfully restart a simulation from whatever point in the simulation the `restart file` was written.

During the *observation phase*, the system is restarted from the `restart file` while maintaining the temperature of the system at this point. The *observation phase* can be executed as many times as desired, each time generating a new `restart file` to restart from. In some of the experiments, the *observation phase* will change a property of the system in order to observe the effects. For example, in one experiment, the system is initialized in a small system with almost no empty space. Then, during the *observation phase* the top of the simulation box was increased to observe what effects this might have on the system. Generally, however, the *observation phase* does not perturb the system and is meant to be executed as many times as desired.

Methods

In this section, we discuss in general the different types of the experiments conducted and the methods employed to do so. Many experiments were done, but the main focus of this paper relate to 7 experiments performed in a *krypton-graphene simulation*. For all of the following simulations, the x and y dimensions of the simulation box are defined to periodic, but the z dimension has different boundary conditions depending on the experiment. In addition, all simulations use LJ units normalized with respect to the σ and ϵ Krypton. The values for krypton were obtained from Sidi Maiga, and are listed in Table 1.

The following paragraphs discuss the *krypton-graphene system* at a high level. The basic initial conditions and the types of changes made during a simulation are discussed, but specific values and actual code has been deferred to the appendix. The code is commented, noting

Lennard-Jones Values		
Value	Krypton	Carbon (Graphene)
σ	3.6 Å	3.4 Å
ϵ	171 K	28 K
$Mass$	83.798 u	12.0107 u

Table 1: The fundamental values used to determine the reduced units in our experiments.

Simulation Box		
Dimension	Size	Boundary Condition
x	39.3522 Å	Periodic
y	38.46 Å	Periodic
z	70 Å	Depends

Table 2: The values used for the simulation box size and boundary conditions in the krypton-graphene system.

specific lines that were changed to perform different kinds of experiments. At the end of this section, a description of the directory structure and the correct way to run the code is provided.

Krypton-Graphene System

This system is defined in it's initial state by a monolayer of krypton adsorbed onto a monolayer of graphene. The initial coordinates of the particles in the adsorbed state were obtained using Gran Canonical Monte Carlo (GCMC) methods, and were performed by Sidi Maiga. We will look at both the *initialization phase* and the *observation phase*, and the roles they play in simulating experiments in this system.

Initialization Using the LJ values from Table 1 and the simulation box values from Table 2, we create a simulation box in LJ units normalized with respect to krypton. This is done simply by dividing the length of the simulation box by the σ of krypton (notice that this results in a unitless value, as expected for the reduced units). The values obtained using GCMC list the coordinates of both the Krypton and the Graphene in Angstroms, so the same operations must be performed to generate their reduced unit values as well. The GCMC simulations were performed on a system in equilibrium at 70K, and so the velocities of the krypton atoms are scaled to that temperature in LAMMPS as well. Using the AWK tool, several scripts were written to perform the conversions into LJ units and then format the data in a

manner that LAMMPS can use. These simple scripts are also included in the appendix. The output generated by these scripts can be used in the `Atoms` section of a LAMMPS data file. More about the specific format of Data files in LAMMPS can be read about here: `http://lammps.sandia.gov/doc/read_data.html`.

The system is sometimes heated during this phase as well. In most of the experiments performed, the temperature was raised from 70K to some higher temperature (such as 140K or 200K) in 100,000 timesteps. This is done using Nose-Hoover thermosttating, and is trivial to implement in LAMMPS (please see the actual code for details).

Observation After the first 100,000 timesteps, the temperature of the system is maintained – either at 70K, 140K, 170K, or 200K, depending on the final state of the initialization phase. Generally, this is the only parameter that's changed in the observation phase, though sometimes other factors, such as the height of the simulation box, were varied here as well.

There are many properties of the system that are measured during the simulation. One of the more complex values that was monitored was the pressure of the gas that was composed of any desorbed atoms. This was done by computing the stress tensor on each desorbed atom, summed across all dimensions, and divided by three times the volume of the gas (*2*). To implement this in LAMMPS, a `region` was created that extended infinitely in the x and y dimensions, and from 2σ above the substrate to the top of the box in the z dimension (which is not periodic). Only atoms in this region were considered desorbed, and thus only these atoms contributed to the pressure of the gas that we were calculating. Other values that we calculated were simpler to compute, such as the temperature of the krypton atoms, the interaction energy between the krypton and the graphene, interaction energy among the krypton atoms themselves, the number of particles currently desorbed, and the volume of the box (which only varies in some experiments). The *observation phase* is designed to be run many times in succession, each time

continuing the simulation from the last point it left off. In some experiments, this phase was executed for up to 5 million timesteps; another was only simulated for 1 million timesteps. The number of steps this phase was executed for was simply determined by when the system equilibrated or ceased to yield any relevant information.

Execution

This section gives an overview of the directory structure used during these simulations, and the correct order in which the code was run to avoid errors and preserve data. If these steps are not followed, it is very possible to overwrite your data, which extends simulation time tremendously as you will have to redo entire experiments. For this reason, it's important to separate out your data and have a workflow that prevents the user from accidentally deleting his or her progress.

Directory Structure The directory structure can be seen in Figure 1. Just below the root folder, we have a `Conf` directory. This stores the configuration and data files that LAMMPS might need to reference to create the simulation environment. This folder is symlinked to by many other directories, which allows many different simulations to use the same configuration files if necessary (for example, if a file describing the CO_2 molecule was needed in two very different simulations).

The other folder at the root level is the main `Simulations` directory, which stores the actual code that will be run, as well as the output generated by each command. Under the `Simulations` directory are directories which describe the types of experiments or systems that will be contained within them, such as the `Desorption` directory. These can have subdirectories as well that narrow the scope of the experiments even further, as `Desorption` does. Finally, a specific experiment directory contains two subdirectories: `Conf` and `Data`. `Conf` is simply a symlink to the main `Conf` directory. `Data`, on the other hand, stores subdirec-

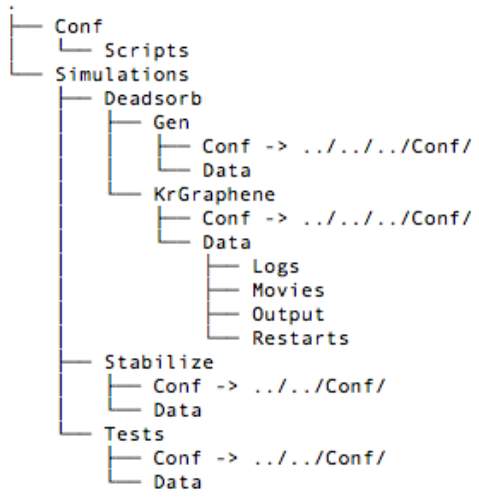

Figure 1: This figure shows the directory structure used by the authors while performing these simulations. By separating LAMMPS scripts, configuration files, and the data output by LAMMPS, one creates a safer and more efficient workflow.

tories that contain different types of output that LAMMPS might generate. All files in DATA should be kept read-only. This will prevent you from accidentally overwriting any data that might come in handy (LAMMPS will throw an error stating that it is unable to open the file if the file is read-only and you are attempting to overwrite the file, on accident or on purpose). A simple bash script that can be executed after running LAMMPS to ensure that all your files are read-only is provided in the appendix. The files provided in this document assume this is the directory structure, and may not work properly if this is not how your development environment looks.

Running the Code Running the LAMMPS script is relatively simple. The *initialization script* should be run only once. After this is run, one should make sure that their output has been generated successfully and has read-only permissions (this can be done with the provided script). The *initialization script* will also generate a `restart` file (which goes in the `Data/Restart` directory) that can be read the *observation script*, which can be run multiple times since it generates it's own `restart` file. After each run of the *observation script*, it is highly recommended you run the provided bash script to ensure that all the data is preserved. Any new data should be output to a new file, which is easily done simply by editing the LAMMPS script. Using this

method will generate a lot of different files. These files should follow a strict naming convention to differentiate one output from another. I often found it helpful to identify the starting and ending temperatures and the final timestep of a simulation, but the choice is arbitrary so long as it works for the user.

Experiments and Results

This section will discuss the specific experiments performed and the results obtained from them. As previously mentioned, the x and y dimensions are set to periodic for all simulations in this paper, but the z direction is handled differently. In all experiments, the z dimension is not periodic, meaning that any atoms that leave the system through a face perpendicular to the z dimension is lost. However, some simulations were initialized with a reflective wall on the top of the simulation box, thus preventing any atoms from exiting the system in the z direction. In fact, six of the eight simulations included this reflective wall; the other two simulations did not use this wall, and allowed the atoms to exit the system. First we will discuss the two simulations that did not use a reflective wall, and then turn the discussion to the six simulations that did use a wall.

No Reflective Wall These simulations were performed to see if we could induce total desorption on a system. By choosing not to have a reflective wall on the top of the simulation box, we allow the particles to entirely leave the system if they exit the simulation box through the top face of the box. This typically throws an error, but LAMMPS allows users to suppress this error and simply issue a warning if desired, which is what was done for this experiment.

There were two experiments done that had no reflective wall. The first experiment initialized the temperature to 70K, and maintained this temperature during the entire simulation. The second experiment raised this temperature to 140K over the first 100,000 timesteps, and then

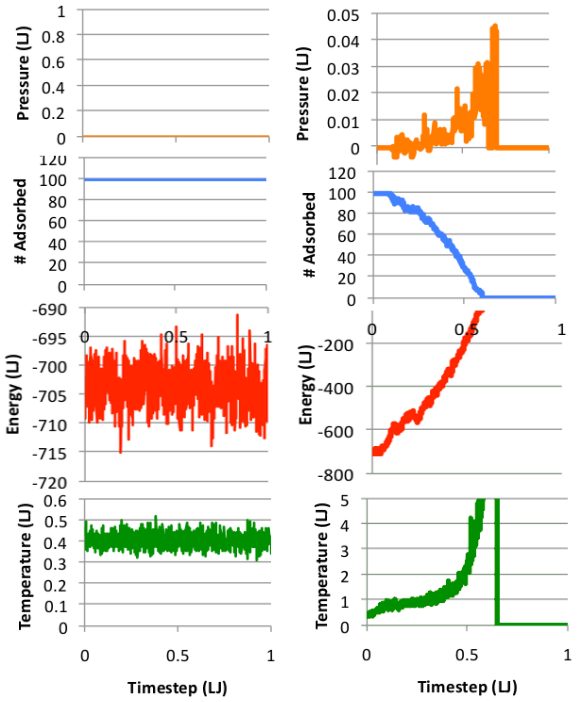

Figure 2: The above graphs show (from bottom to top) temperature, interaction energy between the substrate and the adsorbate, count of atoms adsorbed, and pressure of the vapor of the desorbed atoms. From left to right: 70K, 140K. The timesteps are measured in millions.

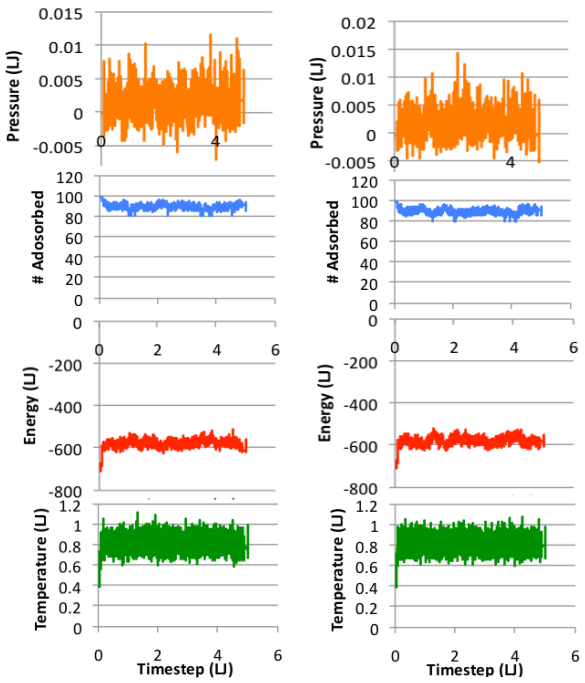

Figure 3: These graphs show the dynamics of the 140K system with a reflective wall. The graphs on the left are taken from the system that was equilibrated at a small volume, and the graphs on the right were simulated at a constant height of 70 Å.

maintained that temperature. Each of the systems was restarted in the *observation phase* until the krypton totally desorbed or reached equilibrium; for the system at 140K, this took less than 1 million timesteps, while the system at 70K was simulated for 3 million timesteps. The data from these simulations can be seen in Figure 2.

The simulation performed at 70K remained in equilibrium during the entire simulation. This was slightly unexpected. As there was no pressure simulated by any gas above the adsorbed particles, we expected to see desorption. However, in the 140K system, complete desorption was observed in less than 1 million timesteps. This is clearly shown in the graphs in Figure 2.

With Reflective Wall With a reflective wall, the system changes significantly. There were five experiments performed with a reflective wall. However, three of those experiments were performed in a four-times larger simulation box than the other simulations, while the other two have the same size box of 70 Å. The three experiments performed in larger simulation box are referred to as the *tall* system, and are discussed in the next paragraph. The first of the experiments performed in the same size of 70 Å raised the temperature of the box from 70K to 140K in 100,000 timesteps, as the previous experiment without a reflective wall did. This simulation was run for 5 million timesteps. The second experiment first equilibrated the system in a small box that forced the adsorbate to stay in close proximity with the substrate. Then, the top of the simulation box was raised to 70 Å. This system also begins at 70K and is increased to 140K in the first 100,000 timesteps, and was also simulated for 5 million timesteps. The purpose of these two experiments was to analyze the effects of the volume change on the dynamics of the system. Graphs of the results are shown in Figure 3. We see the system reaches the same final equilibrium state in both cases.

The remaining three *tall* experiments were performed to see how a larger system would affect the rate of desorption, and also to see how increasing temperatures would affect it. The height of the box was increased just over four times to a height of 288 Å. It was expected that increasing the volume of the simulation box would increase the amount of desorption, as would increasing the temperature. These trends were in fact observed, and can be seen in Figure 4. The *tall* experiments were performed at 140K, 170K, and 200K.

Comparison to GCMC Results With the data that was generated by LAMMPS, we are able to compare to our results to the results of GCMC methods performed by Sidi Maiga. Sidi's simulations yielded isotherms that took the form of plots of *Film Density vs Pressure*, where *Film Density* was number of adsorbed particles divided by the area of the bottom of the simulation

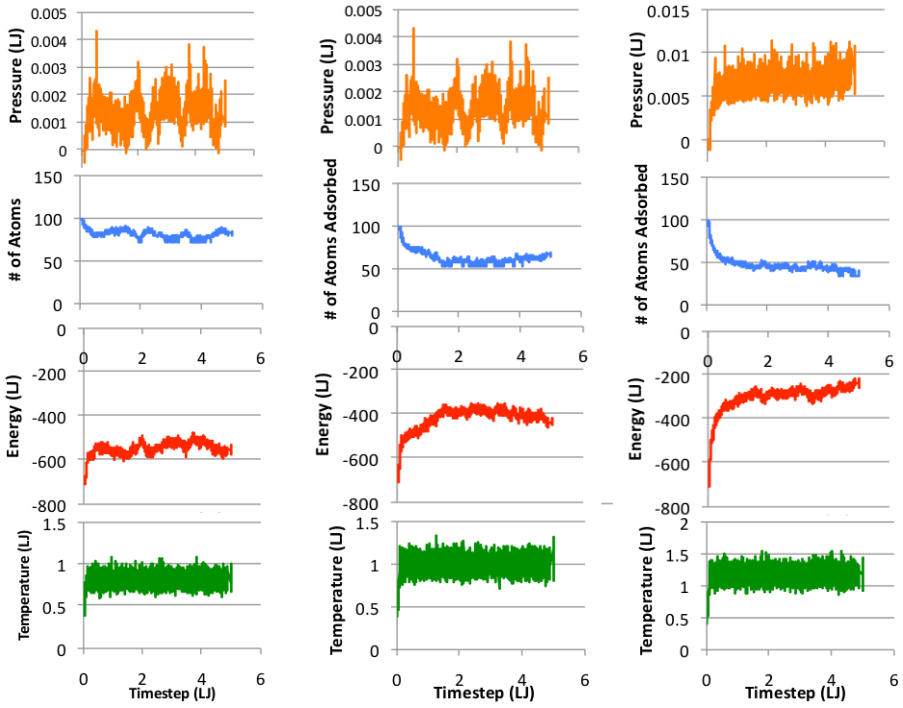

Figure 4: The above graphs show the time evolution of various dynamic properties as the system is evolved through time. Each column of vertically stacked graphs is relates to a single experiment with a reflective wall maintained at a certain temperature. From left to right: 140K, 170K, and 200K. The timesteps are measured in millions, and each experimented was simulated for 5 million timesteps.

box. While our simulations were not performed at a constant pressure, we were able to get a rough idea of how consistent the molecular dynamics methods were with the GCMC methods. We were able to generate a point that could by compared against the isotherms by taking the average number of particles in the film after the temperature had been held constant and the average pressure of the vapor at this time as well. It should be noted this is an extremely rough estimate, since the system is quite small and the number of particles in the vapor that actually contribute to the pressure is even less. However, it can give a rough order of magnitude indication of how realistic these simulations are compared to GCMC methods. The isotherms and the points generated by the method described above can be seen in Figure 5. We certainly did

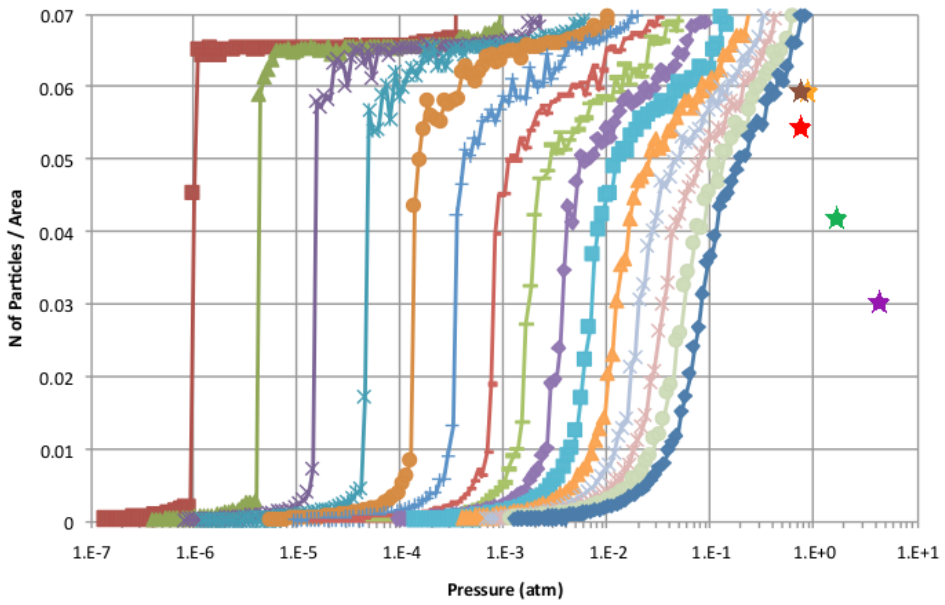

Figure 5: The isotherms shown above were taken between temperatures 70K and 140K with a 5K interval. The stars represent the points generated by our rough estimate from the molecular dynamics simulations. Only the simulations with a reflective wall were plotted. The orange star corresponds to the 140K system at the original 70 Åsimulation box height, the brown star is the 140K system equilibrated in the smaller box, the purple star is the 200K *tall*, the green star is the 170K *tall* system, and the red star is the 140K *tall* system.

not expect perfect agreement between these two methods, and although our estimates were not extremely precise, we believe it shows enough agreement to merit further investigation. Unfortunately, GCMC simulations at temperatures higher than 140K are not available for comparison with our MD results.

Conclusions and Final Remarks

The simulations conducted in this paper were able to replicate the desorption process closely enough to yield results that, qualitatively, appear similar to those generated by GCMC methods. In addition, we were able to generate a large amount of data from seven experiments that provide some insight into the dynamics of desorption of krypton from a graphene monolayer. Future

work could include simulating other types of gases, such as CO_2, methane, or a mixture.

Acknowledgments

The author(s) would like to acknowledge the National Science Foundation for financial support of the Research Experience for Undergraduates (REU) summer program and the Howard University REU in Physics Site (NSF Grant PHY-1358727). I would like to thank Dr. Silvina Gatica, Mr. Sidi Maiga, Ms. Jazzmen Johnson, and Mr. Anton Nekhai for their help this summer.

References

1. W. G. Hoover, *Phys. Rev. A* **31**, 1695 (1985).

2. Sandia Labs, Compute stress/atom (2015).

Appendix

The code that was used to perform these experiments is copied below. Only one generic file was used for both the *initialization phase* and the *observation phase*. The *initialization phase* script is presented first, and is called `in.kr-mono`. The *observation phase* script is presented second, and is called `in.kr-mono-restart`. These files only had to be modified slightly in order to conduct each of these experiments, since each was different from each other in only a few ways, commonly as little as one line different. I have attempted to comment the code so that it is understandable and logical. After these main scripts, some of the helper scripts are presented. First, the file `protect.sh` is listed, a simple script that was run after every experiment to ensure the data was protected and prevent accidental overwrites by LAMMPS. Then, the file `convert-lj-graphene.awk` is shown, which was used by the AWK tool

to convert the data generated by Sidi into a format that is usable in the `Atoms` section of a LAMMPS data file.

in.kr-mono

```
# Initialization script for kr-graphene system

#Set constants
variable affected_height equal 2

#Set up units
units lj
atom_style atomic
boundary p p f

neigh_modify delay 0

# Read data and assign to groups
read_data Conf/kr-graphene.data
# change_box all z final 0 80
group krypton type 1
group graphene type 2

# Initialize velocities
velocity krypton create 0.409356725 7589179

# Define interactions
pair_style lj/cut 2.5
pair_coeff 1 1 1 1
pair_coeff 1 2 0.40465131911 0.972222222
pair_coeff 2 2 0 0 #doesn't matter because we wont time evolve them

# Define energy computes
compute ads_subs krypton group/group graphene
compute ads_ads krypton group/group krypton
compute adsKE krypton ke

# define region that in which the gas pressure will be measured.
region gas block INF INF INF INF ${affected_height} INF
region affected block INF INF INF INF 0 ${affected_height}

# Pressure computes
compute peratom krypton stress/atom NULL
compute  p krypton reduce/region gas sum c_peratom[1] c_peratom[2] c_peratom[3]
compute adsTemp krypton temp
compute_modify adsTemp dynamic yes

#variables
variable vg equal "xhi * yhi * (zhi - v_affected_height)"
variable p equal "-(c_p[1] + c_p[2] + c_p[3]) / (3 * v_vg)"
variable c equal count(krypton,affected)
variable cg equal count(krypton,gas)
variable eas equal c_ads_subs
variable eaa equal c_ads_ads
variable eak equal c_adsKE
variable t equal temp
variable s equal step
```

```
variable et equal etotal
variable v equal vol

# Initialze thermo information
thermo_style custom step temp etotal v_eas v_eaa v_p v_c v_cg vol pe
thermo_modify temp adsTemp lost warn norm no
thermo 1000

# Apply output fixes
fix print all print 100 "$s $t ${et} ${eas} ${eaa} $p $c ${cg} $v" &
file Data/Output/Unnormalized/nvt.70-140.ppf.nw.dyn-temp.100k.out screen no title &
"step temp etotal E_ads_subs E_ads_ads p_press count cg vol"

dump 1 all movie 100 Data/Movies/nvt.70-140.ppf.nw.dyn-temp.100k.mpg type type &
axes yes 0.8 0.02 view 80 -30
dump_modify 1 pad 5


# Apply system fixes
fix 1 krypton nvt temp 0.409356725 0.81871345 .5

run  100000

write_restart Data/Restarts/nvt.70-140.ppf.nw.dyn-temp.100k.restart
```

in.kr-mono-restart

```
# Restart script for in.kr-mono

#Set up constants
variable affected_height equal 2

neigh_modify delay 0

# Read data and assign to groups
read_restart Data/Restarts/nvt.200-200.ppf.tall.shock.7mil.restart

# Change the box height
#change_box all z final 0 19.4444444

# Define energy computes
compute ads_subs krypton group/group graphene
compute ads_ads krypton group/group krypton
compute adsKE krypton ke

# define region that in which the gas pressure will be measured.
region gas block INF INF INF INF ${affected_height} INF
region affected block INF INF INF INF 0 ${affected_height}

# Pressure computes
compute peratom krypton stress/atom NULL
compute  p krypton reduce/region gas  sum c_peratom[1] c_peratom[2] c_peratom[3]
compute adsTemp krypton temp
compute_modify adsTemp dynamic yes

#variables
```

```
variable vg equal "10.9311667 * 10.6833333 * (zhi - v_affected_height)"
variable p equal "-(c_p[1] + c_p[2] + c_p[3]) / (3 * v_vg)"
variable c equal count(krypton,affected)
variable cg equal count(krypton,gas)
variable eas equal c_ads_subs
variable eaa equal c_ads_ads
variable eak equal c_adsKE
variable eat equal c_adsTot
variable t equal temp
variable s equal step
variable et equal etotal
variable v equal vol

# Initialze thermo information
thermo_style custom step temp etotal v_eas v_eaa v_p v_c v_cg vol pe
thermo_modify temp adsTemp lost warn norm no
thermo 1000

# Apply system fixes
fix 1 krypton nvt temp 1.16959064 1.16959064 0.5

# Apply output fixes
fix print all print 100 "$s $t ${et} ${eas} ${eaa} $p $c ${cg} $v" &
file Data/Output/nvt.200-200.ppf.tall.shock.8mil.out screen no title &
"step temp etotal E_ads_subs E_ads_ads p_press c cg vol"
#fix hist krypton ave/histo 1 1 100 0 4 20 z mode vector file &
# Data/histo.kr-mono.out
fix wall krypton wall/reflect zhi EDGE

dump 1 all movie 100 Data/Movies/nvt.200-200.ppf.tall.shock.8mil.mpg type type &
axes yes 0.8 0.02 view 80 -30
dump_modify 1 pad 5


run 1000000

write_restart Data/Restarts/nvt.200-200.ppf.tall.shock.8mil.restart
```

protect.sh

```
#!/bin/bash

chmod -w Data/Restarts/* Data/Output/Unnormalized/* Data/Output/Normalized/* log.lammps
echo "Please move and save the log.lammps!"
```

convert-lj-graphene.awk

```
BEGIN {count=1; xhi=x/sigma; yhi=y/sigma}
$2/sigma >= 0 && $2/sigma <= xhi && $3/sigma >= 0 && $3/sigma <= yhi \
{print count++, 1, $2/sigma, $3/sigma, 0}
END {}
```