Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

Designs for a negative friction coefficient

Mikkel Metzsch Jensen

Thesis submitted for the degree of
Master in Computational Science: Materials Science
60 credits

Department of Physics
Faculty of mathematics and natural sciences

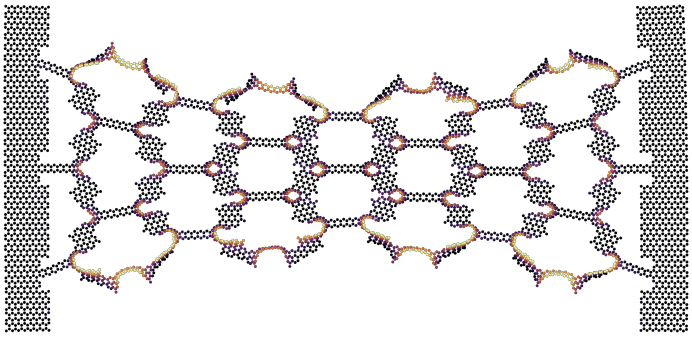
UNIVERSITY OF OSLO

Spring 2023

Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

Designs for a negative friction coefficient

Mikkel Metzsch Jensen



Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Various theoretical models and experimental results propose different governing mechanisms for friction at the nanoscale. We consider a graphene sheet modified with Kirigami-inspired cuts and under the influence of strain. Prior research has demonstrated that this system exhibits out-of-plane buckling, which could result in a decrease in contact area when sliding on a substrate. According to asperity theory, this decrease in contact area is expected to lead to a reduction of friction. However, to the best of our knowledge, no previous studies have investigated the friction behavior of a nanoscale Kirigami graphene sheet under strain. Here we show that specific Kirigami designs yield a non-linear dependency between kinetic friction and the strain of the sheet. Using molecular dynamics simulations, we have found a non-monotonic increase in friction with strain. We found that the friction-strain relationship does not show any clear dependency on contact area which contradicts asperity theory. Our findings suggest that the effect is associated with the out-of-plane buckling of the graphene sheet and we attribute this to a commensurability effect. By mimicking a load-strain coupling through tension, we were able to utilize this effect to demonstrate a negative friction coefficient on the order of -0.3 for loads in the range of a few nN. In addition, we have attempted to use machine learning to capture the relationship between Kirigami designs, load, and strain, with the objective of performing an accelerated search for new designs. Although this approach yielded some promising results, we conclude that further improvements to the dataset are necessary in order to develop a reliable model. We anticipate our findings to be a starting point for further investigations of the underlying mechanism for the frictional behavior of a Kirigami sheet. For instance, the commensurability hypothesis could be examined by varying the sliding angle in simulations. We propose to use an active learning strategy to extend the dataset for the use of machine learning to assist these investigations. If successful, further studies can be done on the method of inverse design. In summary, our findings suggest that the application of nanoscale Kirigami can be promising for developing novel friction-control strategies.

Acknowledgments

The task of writing a master's thesis is a demanding and extensive project which I could not have done without the support of many good people around me. First of all, I want to thank my supervisors Henrik Andersen Sveinsson and Anders Malthe-Sørenssen for the assistance in this thesis work. I am especially grateful for the weekly meetings with Henrik and the inspiring discussions had as we unraveled the discoveries related to the topic of this thesis. I remember that I initially asked for an estimate of how much time he had available for supervision and the answer was something along the lines of "There are no limits really, just send me an email and we figure it out". This attitude captures the main experience I have had working with Henrik and I am profoundly grateful for the time and effort he has put into this project. I hope that he did not regret this initial statement too much, because I have certainly been taken advantage of it. I also want to thank Even Marius Nordhagen for technical support regarding the use of the computational cluster. In that context, I also want to acknowledge the Center for Computing in Science Education (CCSE) for making these resources available.

I would like to express my gratitude to all the parties involved in making it possible for me to write my thesis from Italy. I am particularly grateful for the flexibility shown by my supervisors and for the support of Anders Kvellestad, who allowed me to work remotely as a group teacher. I would also like to thank Scuola Normale Superiore for providing me with access to their library.

I realize that it is a commonly used cliché to express gratitude for the support of loved ones. However, I want to highlight the exceptional role played by my fiancé, Ida, who deserves the main credit for enabling me to maintain a healthy state of mind. She has provided me with a solid foundation for a fulfilling life that enables me to pursue secondary objectives, such as an academic career. I look forward to spending the rest of my life with you.

In this thesis, I have used the formal pronoun "we" mainly as a customary habit related to the formalities of scientific writing in a team. Nonetheless, I have realized that this usage is more fitting as I have not been working alone on this project. I have received support all the way from colleagues and friends at the University of Oslo, my family residing in Denmark, and my life partner who slept beside me every night here in Italy. They are the "good people around me" who have made this thesis possible.

Acronyms

CM Center of Mass. 12

FFM Friction Force Microscopy. 9

GAN Generative Adversarial Networks. 2

MD Molecular Dynamics. 1, 2, 3, 4, 9, 10, 25

ML Machine Learning. 2

NN Nearest Neighbors. 14

SFA Surface Force Apparatus. 9

Contents

Chapter 1

Introduction

1.1 Motivation

Friction is the force that prevents the relative motion of objects in contact. In our everyday life, we recognize it as the inherent resistance to sliding motion. Some surfaces appear slippery and some appear rough, and we know intuitively that sliding down a snow-covered hill is much more exciting than its grassy counterpart. Without friction, it would not be possible to walk across a flat surface, lean against the wall without falling over or secure an object by the use of nails or screws [1, p. 5]. It is probably safe to say that the concept of friction is integrated into our everyday life to such an extent that most people take it for granted. However, the efforts to control friction date back to the early civilization (3500 B.C.) with the use of the wheel and lubricants to reduce friction in translational motion [2]. Today, friction is considered a part of the wider field *tribology* derived from the Greek word *tribos* meaning "rubbing". It includes the science of friction, wear and lubrication [2]. The most compelling motivation to study tribology is ultimately to gain full control of friction and wear for various technical applications. Especially, the reduction of friction is of great interest since this can be utilized to improve energy efficiency in mechanical systems with moving parts. Hence, it has been reported that tribological problems have a significant potential for both economic and environmental improvements [3]:

> "On global scale, these savings would amount to 1.4% of the GDP annually and 8.7% of the total energy consumption in the long term." [4].

On the other hand, the reduction of friction is not the only sensible application for tribological studies. Controlling frictional properties, besides minimization, might be of interest in the development of a grasping robot where finetuned object handling is required. While achieving a certain "constant" friction response is readily obtained through appropriate material choices, we are yet to unlock the full capabilities to alter friction dynamically on the go. One example from nature inspiring us to think along these lines is the gecko feet. More precisely, the Tokay gecko has received a lot of attention in scientific studies aiming to unravel the underlying mechanism of its "togglable" adhesion properties. Although the gecko can produce large adhesive forces, it retains the ability to remove its feet from an attachment surface at will [5]. This makes the gecko able to achieve a high adhesion on the feet when climbing a vertical surface while lifting them for the next step remains relatively effortless. For a grasping robot, we might consider an analog frictional concept of a surface material that can change from slippery to rough on demand depending on specific tasks; slippery and smooth when interacting with people and rough and firmly gripping when moving heavy objects.

In recent years an increasing amount of interest has gone into the studies of the microscopic origins of friction, due to the increased possibilities in surface preparation and the development of nanoscale experimental methods. Nano-friction is also of great concern for the field of nano-machining where the frictional properties between the tool and the workpiece dictate machining characteristics [3]. With concurrent progress in computational capacity and development of Molecular Dynamics (MD), numerical investigations serve as an invaluable tool for getting insight into the nanoscale mechanics associated with friction. This simulation-based approach can be considered as a "numerical experiment" enabling us to create and probe a variety of high-complexity systems which are still out of reach for modern experimental methods.

In materials science such MD-based numerical studies have been used to explore the concept of so-called *metamaterials* where the material compositions are designed meticulously to enhance certain physical properties [6–11]. This is often achieved either by intertwining different material types or removing certain regions completely. In recent papers by Hanakata et al. [6, 7], numerical studies have showcased that the mechanical properties of a graphene sheet, yield stress and yield strain, can be altered through the introduction of so-called *Kirigami* inspired cuts into the sheet. Kirigami is a variation of origami where the paper is cut additionally to being folded. While these methods originate as an art form, aiming to produce various artistic objects, they have proven to be applicable in a wide range of fields such as optics, physics, biology, chemistry and engineering [12]. Various forms of stimuli enable direct 2D to 3D transformations through the folding, bending, and twisting of microstructures. While original human designs have contributed to specific scientific applications in the past, the future of this field is highly driven by the question of how to generate new designs optimized for certain physical properties. However, the complexity of such systems and the associated design space makes for seemingly intractable[1] problems ruling out analytic solutions.

Earlier design approaches such as bioinspiration, looking at gecko feet for instance, and Edisonian, based on trial and error, generally rely on prior knowledge and an experienced designer [9]. While the Edisonian approach is certainly more feasible through numerical studies than real-world experiments, the number of combinations in the design space rather quickly becomes too large for a systematic search, even when considering the computation time on modern-day hardware. However, this computational time constraint can be relaxed by the use of machine learning (ML) which has been proven successful in the establishment of a mapping from the design space to physical properties of interest. This gives rise to two new styles of design approaches: One, by utilizing the prediction from a trained network we can skip the MD simulations altogether resulting in an *accelerated search* of designs. This can be further improved by guiding the search according to the most promising candidates. For instance, as done with the *genetic algorithm* based on mutation and crossing. Another more sophisticated approach is through generative methods such as *Generative Adversarial Networks* (GAN) or diffusion models. The latter is being used in state-of-the-art AI systems such as OpenAI's DALL·E2 [13] or Midjourney [14]. By working with a so-called *encoder-decoder* network structure, one can build a model that reverses the prediction process. This is often referred to as *inverse design*, where the model predicts a design based on physical target properties. In the papers by Hanakata et al. [6, 7] both the accelerated search and the inverse design approach was proven successful to create novel metamaterial Kirigami designs with the graphene sheet.

Hanakata et al. attribute the variation in mechanical properties to the non-linear effects arising from the out-of-plane buckling of the sheet. Since it is generally accepted that the surface roughness is of great importance for frictional properties it can be hypothesized that Kirigami-induced out-of-plane buckling can also be exploited for the design of frictional metamaterials. For certain designs, we might hope to find a relationship between the stretching of the sheet and frictional properties. If significant, this could give rise to an adjustable friction beyond the point of manufacturing. For instance, the grasping robot might apply such a material as artificial skin for which stretching or relaxing of the surface could result in a changeable friction strength.

In addition, the Kirigami graphene properties can be explored through a potential coupling between the strain and the normal load, through a nanomachine design, with the aim of altering the friction coefficient. This invites the idea of non-linear friction coefficients which might in principle also take on negative values. This would constitute a rarely found property which is mainly observed for the unloading phase of adhesive surfaces [15] or in the loading phase of particular heterojunction materials [16, 17].

To the best of our knowledge, Kirigami has not yet been implemented to alter the frictional properties of a nanoscale system. However, in a recent paper by Liefferink et al. [18] it is reported that macroscale Kirigami can be used to dynamically control the macroscale roughness of a surface through stretching. They reported that the roughness change led to a changeable frictional coefficient by more than one order of magnitude. This supports the idea that Kirigami designs can be used to alter friction, but we believe that taking this concept to the nanoscale would involve a different set of governing mechanisms and thus contribute to new insight in this field.

1.2 Goals

In this thesis, we investigate the prospects of altering the frictional properties of a graphene sheet through the application of Kirigami-inspired cuts and stretching of the sheet. With the use of molecular dynamics (MD)

[1] In computer science we define an *intractable* problem as a problem with no *efficient* algorithm to solve it nor any analytical solutions. The only way to solve such problems is the *brute-force* approach, simply trying all possible combinations, which is often beyond the capabilities of computational resources.

simulations, we evaluate the frictional properties of various Kirigami designs under different physical conditions. Based on the MD results, we investigate the possibility to use machine learning for the prediction of frictional properties and subsequently using the model for an accelerated search of new designs. The main goals of the thesis can be summarized as follows.

1. Design an MD simulation procedure to evaluate the frictional properties of a Kirigami graphene sheet under specified physical conditions.
2. Develop a numerical tool to generate various Kirigami designs, both by seeking inspiration from macroscale designs and by the use of a random-walk-based algorithm.
3. Investigate the frictional behavior under varying strain and load for different Kirigami designs.
4. Develop and train a machine learning model to predict the MD simulation results and perform an accelerated search of new designs with the goal of optimizing certain frictional properties.

1.3 Contributions

By working towards the goals outlined above (Sec. 1.2), I have discovered a non-linear relationship between the kinetic friction and the strain for certain Kirigami patterns. This phenomenon was found to be associated with the out-of-plane buckling of the Kirigami sheet but with no clear relationship to the contact area or the tension in the sheet. I found that this method does not provide any mechanism for a reduction in friction, in comparison to a non-cut sheet. However, the straining of certain Kirigami sheets allows for a non-monotonic increase in friction. The relationship to normal load was proven negligible in this context and I have demonstrated that a coupled system of load and strain (through sheet tension) can exhibit a negative friction coefficient in certain load ranges. Moreover, I have created a dataset of roughly 10,000 data points for assessing the employment of machine learning and accelerated search of Kirigami designs. I have found, that this approach might be useful, but that it requires an extended dataset in order to produce reliable results for a search of new designs.

During my investigations, I have built three numerical tools, in addition to the usual scripts for data analysis, which are available on Github [19]. The tools are summarized in the following.

- I have written a LAMMPS-based [20] tool for simulating and measuring the frictional properties of a graphene sheet sliding on a substrate. The code is generally made flexible with regard to the choice of sheet configuration, system size, simulation parameters and MD potentials, which makes it applicable for further studies on this topic. I have also built an automized procedure to carry out multiple simulations under varying parameters by submitting jobs to a computational cluster via an ssh connection. This was done by adding minor additions to the Python package developed by E. M. Nordhagen [21].
- I have generated a Python-based tool for generating Kirigami patterns and exporting these in a compatible format with the simulation software created. The generation of molecular structures is done with the use of ASE [22]. Our software includes two classes of patterns inspired by macroscale designs and a random walk algorithm which allows for a variety of different designs through user-defined biases and constraints. Given our system size of choice, the first two pattern generators are capable of generating on the order of 10^8 unique designs while the random walk generator allows for significantly more.
- I have built a machine-learning tool based on Pytorch [23] which includes setting up the data loaders, a convolutional network architecture, a loss function, and general algorithms for training and validating the results. Additionally, I have written several scripts for performing grid searches and analyzing the model predictions in the context of the frictional properties of graphene.

All numerical implementations have been originally developed for this thesis except for the libraries mentioned above along with common Python libraries such as Numpy and Matplotlib.

1.4 Thesis structure

The thesis is divided into two parts. In Part I we introduce the relevant theoretical background, and in Part II we present the numerical implementations and the results of this thesis.

Part I contains a description of the theoretical background related to Friction (**??**), Molecular Dynamics (**??**) and Machine Learning (**??**). In **??** we formulate our research questions in the light of the friction theory.

In Part II, we begin by presenting the system in Chapter 2 which includes a definition of the main parts of the system and the numerical procedures related to the MD simulation. Here we also present the generation of Kirigami designs. In **??** we carry out a pilot study where we evaluate the simulation results for various physical conditions and compare a non-cut sheet to two different Kirigami designs. In **??**, we further explore the Kirigami patterns through the creation of a dataset and the employment of machine learning and an accelerated search for new designs. In **??**, we use the results from the pilot study to demonstrate the possibility to achieve a negative friction coefficient for a system with coupled load and strain. Finally, in **??**, we summarize our results and provide an outlook for further studies. Additional figures are shown in **??**, **??** and **??**.

Part I

Background Theory

Part II

Simulations

Chapter 2

Creating a graphene kirigami system

The system geometry plays an essential role in the "friction experiment" that we are going to carry out through MD simulations. The purpose of the simulations is to quantify the friction that arises when a stretched Kirigami graphene sheet slides over a substrate. We aim to design the simplest possible system that allows for such a measurement under variations of Kirigami design, strain and load.

For this purpose, two approaches were considered as sketched in Fig. 2.1. One approach is simply to mimic a FFM type experiment where the graphene sheet is resting on a substrate and a moving body scans across the graphene surface as seen in Fig. 2.1b. This setup allows for a variety of tip designs, and we can even substitute the tip for a flat surface making the setup resemble a SFA experiment instead. For this setup, we would attach a pre-stretched sheet to the substrate and require the edges of the sheet to be fixated on the substrate to sustain the strain. Thus, the sheet and substrate would constitute the manufactured object and the moving body would represent the contact to the outside world. In this approach, the potential applications would relate to certain effects being associated with a constant strain value. Another approach is to have the sheet ends fixated on the moving body instead as shown in Fig. 2.1a. This switches the roles of the involved parts as we now view the moving body and the sheet together as the manufactured object, while contact with the substrate represents the outside world. This allows for the introduction of a nanomachine design that converts the loading of the manufactured object into a strain of the sheet. Thus, the possible applications allow for a dynamic effect with changing strain through the loading of the sheet. While both methods serve as novel approaches with prospects of providing valuable insight into a sparsely covered field, we choose the latter option (Fig. 2.1a) due to the increased application possibilities.

We do not attempt to model the nanomachine explicitly, but we will use the conceptual idea of a coupling between load and strain to motivate our study. Hence our system of choice consists of a 2D graphene sheet with locked ends, mimicking the attachment to a moving body, and a 3D silicon bulk substrate.

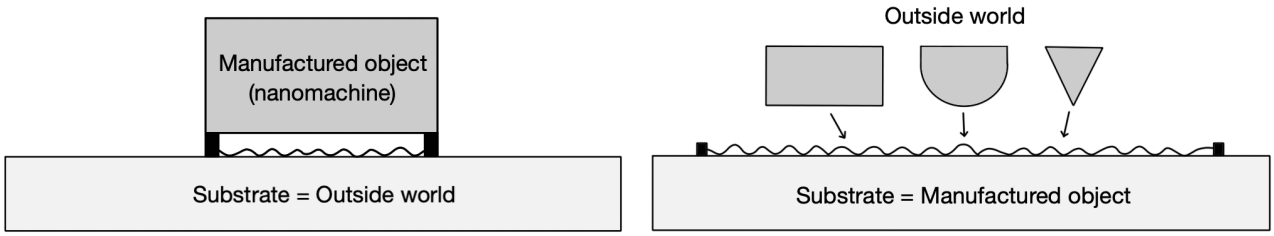


Figure 2.1: Conceptual visualization of two different system setups considered. The wiggly line represents the Kirigami sheet. (a) The chosen setup with a manufactured object connected to the sheet. The contact with the substrate represents the contact with the outside world. (b) An alternative setup where the sheet is fixed in the substrate constituting the manufactured object. The contact with the outside world is then represented through an indention by objects with various shapes and sizes.

2.1 Region definitions

We subdivide the two main parts of the system, the sheet and the substrate, into specific regions according to their functionality in the MD simulations. For the sheet, we denote a subsection of the ends, with respect to the sliding direction, as so-called *pull blocks*, which is reserved for the application of normal load, stretching, sliding of the sheet, and for applying the thermostat. The remaining *inner sheet* is left for the Kirigami cuts and is simulated as an NVE ensemble. This partitioning is motivated by the idea that we want to minimize direct manipulations on the inner sheet, given its presumed critical role in governing friction behavior. The pull blocks are equally split between a thermostat part and a rigid part. It should be noted that the rigidness of the pull blocks is enforced only after a relaxation period to ensure that the crystal structure is fully relaxed. This is further explained in Sec. 2.2. The substrate is equally divided into three parts: The *upper layers* (NVE) responsible for the sheet-substrate interaction, the *middle layers* being a thermostat (NVT), and the *bottom layers* being frozen, made rigid and fixed, in the initial lattice structure to ensure that the substrate stays in place. Fig. 2.2 shows the system with colors matching the three distinct roles:

1. Red: NVE parts which are governing the frictional behavior of interest.

2. Green: Thermostats (NVT) surrounding the NVE parts in order to modify the temperature without making disturbing changes to the interaction of the sheet and substrate.

3. Blue: Parts that are initially or eventually turned into rigid objects. For the substrate, this refers to an additional fixation as well.

The full sheet is given a size $\sim 130 \times 163$ Å while the substrate is scaled accordingly to the sheet which is further specified in Sec. 2.3. For an expected strain of 200%, the total system size is roughly 5.7×10^4 atoms. The specific distribution of atoms is shown in Table 2.1 along with the spatial x-y-measures in Table 2.2. An example of a strained sheet is shown in Fig. 2.3.

Table 2.1: Specification of the system size regarding the number of atoms for various system regions. These numbers correspond with the case of no cuts applied to the sheet and a substrate scaled for the expected sheet strain of 200%.

Region	**Total**	Sub-region	Subtotal	**NVE**	**NVT**	**Rigid**
Full sheet	8060	Inner sheet	6572	6572	0	0
		Pull blocks	1488	0	744	744
Substrate	49068	Upper	16356	16356	0	0
		Middle	16356	0	16356	0
		Bottom	16356	0	0	16356
All	57128			22928	17100	17100

Table 2.2: Specification of the spatial size of the system for the x-y-dimensions with a substrate scaled for an expected sheet strain of 200%. The first column denotes the size relative to the full sheet size $x_S \times y_S$, while the second column denotes the corresponding length in Å.

Region	Dim	Dim [Å]	Area [nm^2]
Full sheet	$x_S \times y_S$	130.029×163.219 Å	212.23
Inner sheet	$x_S \times 0.81\ y_S$	130.029×132.853 Å	172.74
Pull blocks	$2 \times x_S \times 0.09\ y_S$	$2 \times 130.029 \times \ 15.183$ Å	$2 \times \ 19.74$
Substrate	$1.16\ x_S \times 3.12\ y_S$	150.709×509.152 Å	767.34

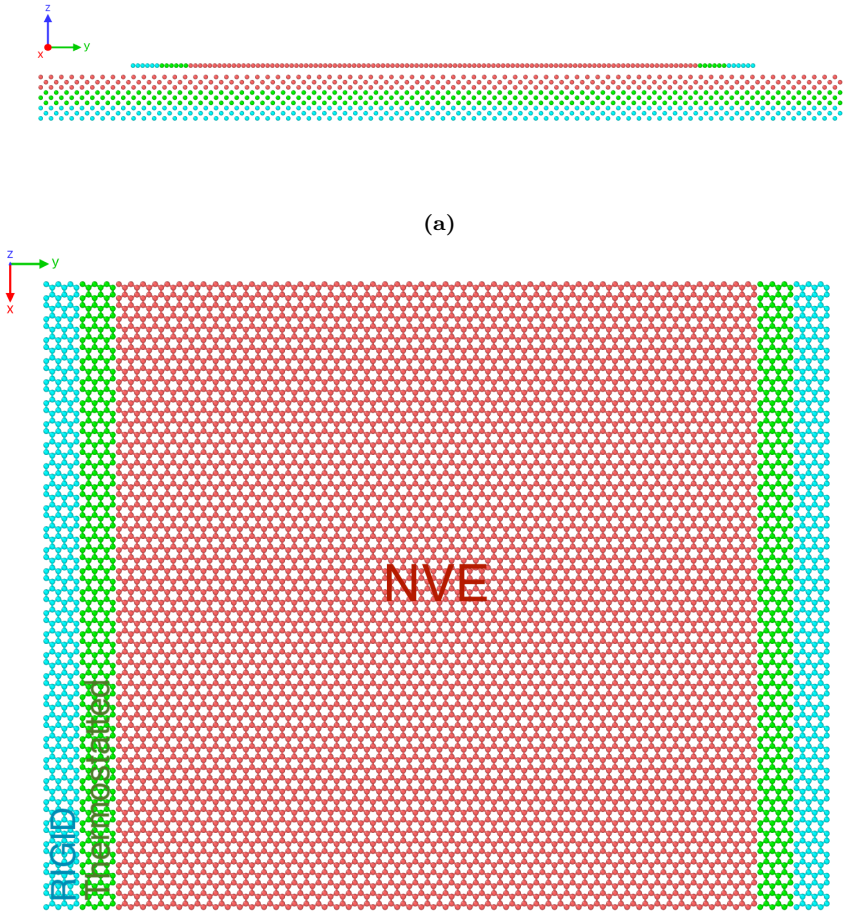


Figure 2.2: System configuration colorized to indicate NVE parts (red), thermostat parts (green) and rigid parts (blue). (a) Side view showing the sheet on top of the substrate. (b) Top view showing only the sheet.

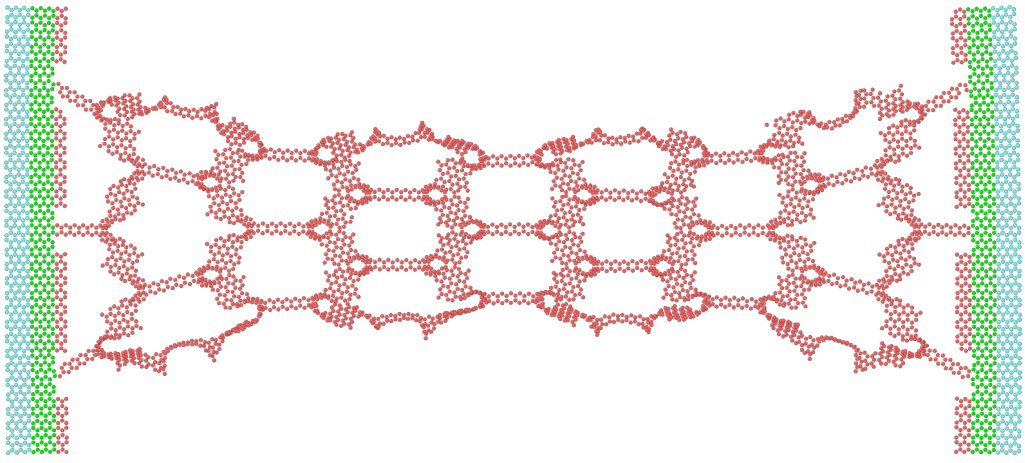


Figure 2.3: Stretched Kirigami sheet against a substrate. The substrate is excluded from the image for better visibility. The atoms are colorized to indicate NVE parts (red), thermostat parts (green) and rigid parts (blue). The pattern used is the Honeycomb $(2, 2, 1, 5)$ (see Chapter 2) at a 100% strain level.

2.2 Numerical procedure

Following the system setup as described previously, the next step involves letting the system relax and reach a stable equilibrium. During this phase, slight modifications to the sheet-substrate distance and lattice spacing take place, both of which are influenced by the thermostat's target temperature. We then stretch the sheet to the desired length, apply normal load and finally slide it along the substrate. The full numerical procedure can be arranged into the following steps. Some steps have been given a default duration denoted in parentheses in units of ps, 10^{-12} seconds.

1. **Relaxation** (15 ps): The sheet and substrate are relaxed for 15 ps after being added in their crystalline form with a separation distance of 3 Å. Given that the equilibrium separation distance will vary with temperature, this value is based on an average estimate suiting our temperature range of interest. In order to avoid any sheet drift we constrain it with the use of three hard spring forces with a spring constant of $10^5\,\mathrm{eV/Å}^2 \sim 1.6 \times 10^6\,\mathrm{N/m}$: One spring attaches the sheet center of mass (CM) to its original position, preventing CM drift, while the remaining two springs are attached to the pull blocks CM, to prevent rotation. In principle, fixing only one of the pull blocks would suffice, but we choose to fixate both to maintain symmetry. During the relaxation phase, we consider the pull blocks to be rigid with respect to the z-direction only (perpendicular to the sheet). That is, all the forces in the z-direction are summed up and applied as a uniform external force, while the pull blocks are free to expand and contract in the x-y-plane. This feature is incorporated to allow the pull blocks to readjust the lattice spacing according to the temperature of the system. For the following steps, the pull blocks are made truly rigid with respect to all directions, and the spring forces are terminated.

2. **Stretch**: To stretch the sheet, the two opposing rigid parts of the pull blocks are moved apart at a constant velocity until the desired strain level is achieved. The duration of this phase is determined by the values of the *strain speed* and *strain amount* parameters.

3. **Pause** (5 ps): The sheet is further relaxed for 5 ps to ensure that the sheet is stable and equilibrated after the applied strain deformation.

4. **Normal load** (5 ps): The pull blocks are subjected to a uniformly applied load in the negative z-direction, thereby pushing the sheet perpendicularly into the substrate. Initially a viscous damping force, $F = -\gamma\mathbf{v}$ with damping factor γ and velocity $\mathbf{v}$, is added to resist rapid acceleration of the sheet and prevent a hard impact between the sheet and substrate as the sheet-substrate distance decreases. The damping coefficient is set to $\gamma = 8 \times 10^{-4}\,\mathrm{nN/(m/s)}$ and terminated after 0.5 ps which was found to be suitable for the extreme load cases of our intended range. The remaining 4.5 ps is devoted to further relaxation in order to reach a sheet-substrate distance equilibrium.

5. **Sliding**: A virtual atom is introduced into the simulation which exclusively interacts with the rigid parts of the pull blocks through a spring force with spring constant K in the x-y-plane. The force in the z-direction is not influenced by the spring force and is instead governed by the equilibrium between the normal load and the normal force response from the sheet-substrate interaction. The virtual atom is immediately given a constant velocity, given by the *sliding speed* parameter. This results in an initial linear increase in sliding force proportional to sliding speed and spring constant $F_{slide} \propto Kvt$. An infinite spring constant can also be enforced for which the spring is omitted and the pull blocks are moved rigidly with a constant speed.

To limit the complexity of the friction behavior we want to consider systems without wear. To make sure that no wear is taking place for the sheet, we monitor the nearest neighbors for each atom throughout the simulation. At the initial timestep the three nearest neighbors, sitting at a distance 1.42 Å, of all graphene atoms are recorded. If any of these nearest neighbors exceeds a threshold distance of 4 Å, indicating a bond breakage, this is marked as a rupture and we halt the simulation. By conducting several test simulations involving high loads and high sliding speeds, we have visually confirmed that no wear occurs in the substrate, which demonstrates significantly greater wear resistance than the sheet. Therefore, we concluded that it was unnecessary to monitor the substrate for any signs of wear during the simulations.

2.3 Setting up the substrate

The substrate is created as a rectangular slab of silicon (Si). We create the initial configuration according to its crystalline structure given as a diamond cubic crystal with a lattice parameter $a_{\mathrm{Si}} = 5.43\,\text{Å}$. The default substrate thickness is chosen such that 9 layers of atoms appear (2 unit cells) corresponding to a thickness of 10.86 Å. The x-y dimensions are chosen to match the dimensions of the sheet. That is, we define a margin between the sheet edge and the substrate edge for the x- and y-direction respectively. Since we use periodic boundary conditions a too small margin would result in the sheet edges interacting with themselves through the boundary. The absolute lower limit for the margin choice is half the cut-off distance for the Tersoff potential, governing the graphene sheet interaction, at $R + D = 2.1\,\text{Å}$. However, due to fluctuations in the sheet, we cannot set the margin too close to that limit. In addition, we need to consider the buckling of the sheet as it is stretched, which might cause an expansion in the x-direction for certain configurations. We choose an x-margin of 20 Å which provides $2 \cdot 20\,\text{Å} - 2.1\,\text{Å} = 37.9\,\text{Å}$ of additional spacing with respect to the absolute lower limit. By looking over the simulation result visually we confirm that this leaves more than enough room in the cases of most buckling. For the y-direction the rigid parts of the pull-blocks moves a certain distance based on the strain value exclusively, and we define the y-margin based on the remaining distance to the edge after stretching. However, as the sheet travels through the periodic boundaries in the y-direction when sliding, we want to add some additional spacing through the y-margin in order to let the substrate surface relax before interacting with the sheet a second time. We choose a y-margin of 15 Å for which the preferred sliding speed of $20\,\text{m/s} = 0.2\,\text{Å/ps}$ gives 150 ps of relaxation time between encounters with the sheet.

2.4 Setting up sheet

The sheet consists of graphene, which is a single layer of carbon atoms arranged in a hexagonal lattice structure. The bulk version of graphene is graphite and is a stacked structure of multiple graphene layers.

We can describe the graphene 2D crystal structure in terms of its primitive lattice vectors $\mathbf{a_1}$ and $\mathbf{a_2}$ and a basis. The basis describes the atoms associated with each lattice site, and we populate the lattice by translating the basis by any linear combination of the lattice vectors

$$\mathbf{T}_{mn} = m\mathbf{a_1} + n\mathbf{a_2}, \qquad m, n \in \mathbb{N}.$$

For graphene, we have the primitive lattice vectors [24]

$$\mathbf{a_1} = a\left(\frac{\sqrt{3}}{2}, -\frac{1}{2}\right), \qquad \mathbf{a_2} = a\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right), \tag{2.1}$$

$$|\mathbf{a_1}| = |\mathbf{a_2}| = a = 2.46\ \text{Å}.$$

Notice that we deliberately excluded the third coordinate as we only consider a single graphene layer and thus we do not have to consider the stacking structure of 3D graphite. The basis consists of two carbon atoms given as

$$\left\{\Big(0,0\Big), \frac{a}{2}\Big(\frac{1}{\sqrt{3}}, 1\Big)\right\}. \tag{2.2}$$

The crystal structure is visualized in Fig. 2.4. The hexagonal lattice structure makes for equal spacing between all pairs of atoms with an interatomic distance

$$\left|\left|\frac{a}{2}\Big(\frac{1}{\sqrt{3}}, 1\Big)\right|\right| \approx 1.42\ \text{Å}.$$

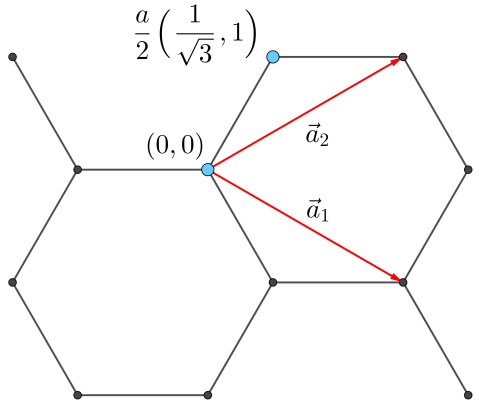


Figure 2.4: Illustration of the graphene crystal structure. The dots represent atom sites whereas blue dots denote the basis atoms (Eq. (2.2)). The red arrows denote the primitive lattice vectors (Eq. (2.1)).

2.4.1 Indexing

In order to describe the Kirigami cut patterns applied to the graphene sheet we require an indexing system that provides a unique representation of the atoms in the lattice. This allows us to represent the pattern as a binary matrix, where 0 denotes removed atoms and 1 denotes present atoms. We let the x-coordinate correspond to the so-called *armchair* direction of the sheet and the y-coordinate to the so-called *zigzag* direction. Notice that the x-coordinate will point to *zigzag* chains of atoms for which the starting point $(x, 0)$ is not evenly spaced as illustrated in Fig. 2.5. Other solutions might naturally involve the lattice vectors, but since these are used to translate between similar basis atoms it introduces an unfortunate duality as one would then need to include the basis atom of choice into the indexing system as well. Additionally, we want an indexing system that conserves the relative physical position of neighbors. That is, atom (i, j) should be in the proximity of $\{(i+1, j), (i-1, j), (i, j+1), (i, j-1)\}$. However, due to the hexagonal structure of the lattice, only three said neighbor indexes will be actual nearest neighbors in the lattice. While $(i, j \pm 1)$ is always the nearest neighbor, the index of the nearest neighbor in the x-direction oscillates for each incrementing of the x- or y-coordinate. That is, the nearest neighbors NN are decided as

$$\begin{aligned} (i+j) \text{ is even} &\rightarrow \text{NN} = \{(i-1, j), (i, j+1), (i, j-1)\}, \\ (i+j) \text{ is odd} &\rightarrow \text{NN} = \{(i+1, j), (i, j+1), (i, j-1)\}. \end{aligned} \tag{2.3}$$

We can visually verify this by consulting Fig. 2.5, which shows that the nearest neighbor indexes depend on whether the atom is oriented to the left or right side in the zigzag chain.

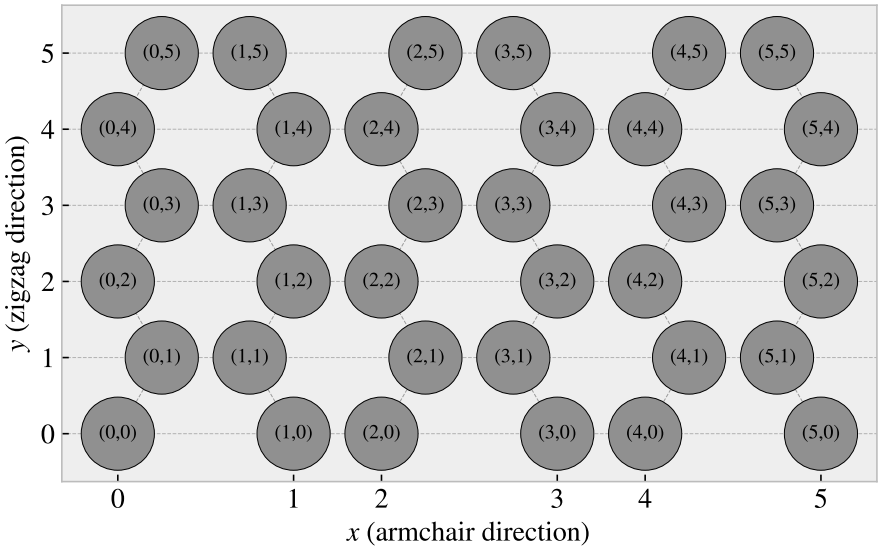


Figure 2.5: Illustration of the graphene atom site indexing. The x-coordinate increment along the armchair direction (pointing to zigzag chains) while the y-coordinate increment along the zigzag direction.

2.4.2 Removing atoms

To simplify the formulation of the cut patterns, we introduce the *center element* which is placed in each gap of the hexagonal honeycomb structure as shown in Fig. 2.6. These are not populated by any atoms but will serve as a reference for the numerical approaches to defining a cut pattern.

The nearest neighbors of the center element alternate with position, similar to the atom site indexing. However, this time it is only dependent on the x-coordinate position. Each center element has six nearest neighbors, in a clockwise direction we can denote them: "up", "upper right", "lower right", "down", "lower left", "upper left". The "up" and "down" neighbors are always accessible as $(i, j \pm 1)$. However, for even i the $(i+1, j)$ index corresponds to the "lower right" neighbour while for odd i this corresponds to the "upper right" neighbour. This shifting applies for all left- or right-oriented neighbors and the full neighbor list is illustrated in Fig. 2.7.

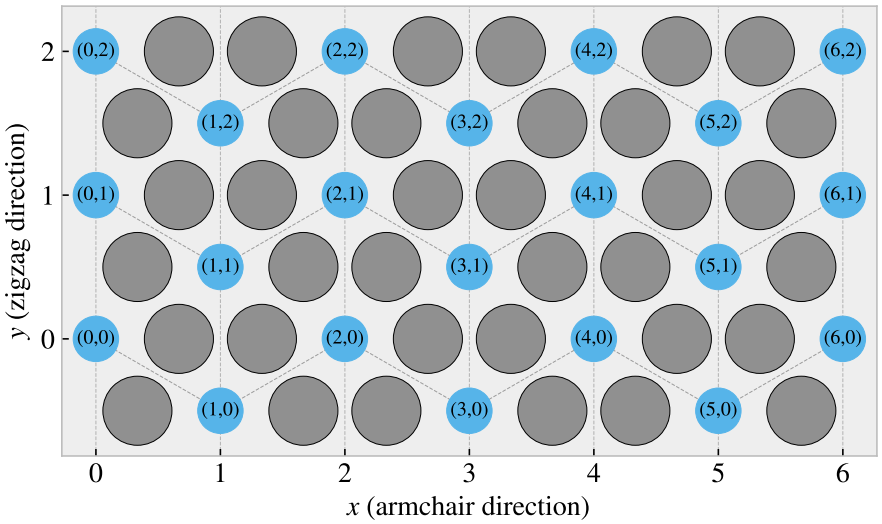


Figure 2.6: Illustration of the indexing for the introduced center elements depicted with blue circles placed in the gap of the honeycomb structure laid out by the graphene sheet atoms depicted with grey circles. The y-coordinate increment along the zigzag direction (pointing to armchair chains) while the x-coordinate increment along the armchair direction.

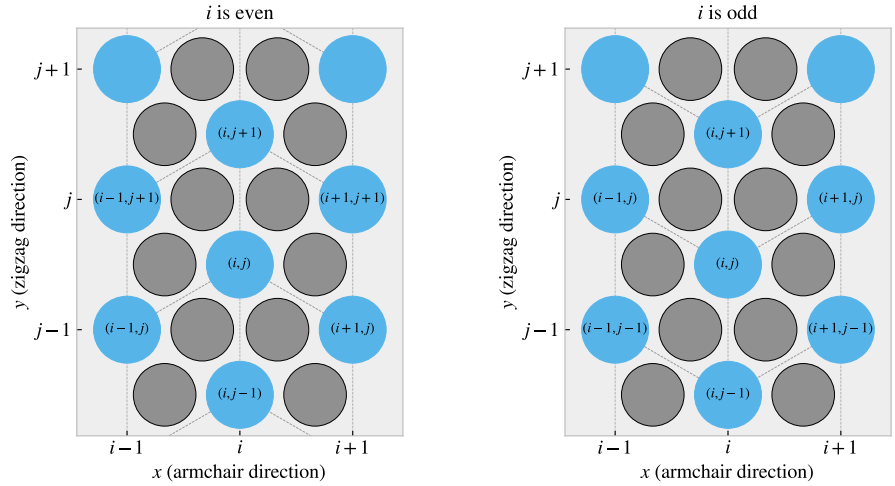


Figure 2.7: Illustration of the center element neighbor indexes for the case when the x-coordinate i is even (left) and i is odd (right).

Analog to the case of atom site indexing, we notice that the nearest neighbor indexes depend on whether the center element is oriented up or down on the armchair chain.

We define a cut pattern by a connected path of center elements. As we walk between center elements through nearest neighbor walking we can remove atoms according to one of two rules

1. Remove intersecting atoms: We remove the pair of atoms placed directly in the path we are walking. That is, when jumping to the "up" center element we remove the two upper atoms located in the local hexagon of atoms. This method is sensitive to the order of the center elements in the path.

2. Remove all surrounding atoms: We simply remove all atoms in the local hexagon surrounding each center element. This method is independent of the ordering of center elements in the path.

We notice that removing atoms using either of these rules will not guarantee an injective, one-to-one, mapping. The first rule, being path dependent, will more often result in a unique result. However, for both methods, it is

possible to construct two different paths leading to the same cut pattern as shown in the following example:

$$\text{Path 1:}\quad (i,j) \to \underbrace{(i+1,j+1)}_{\text{upper right}} \to \underbrace{(i,j+1)}_{\text{up}} \to \underbrace{(i+1,j+2)}_{\text{upper right + up}} \to \underbrace{(i+1,j+1)}_{\text{upper right}}$$

$$\text{Path 2:}\quad (i,j) \to \underbrace{(i+1,j+1)}_{\text{upper right}} \to \underbrace{(i+1,j+2)}_{\text{upper right + up}} \to \underbrace{(i,j+1)}_{\text{up}}$$

For the second rule, it is even more obvious that different paths can result in the same final pattern. For instance, if we incircle a center element completely there will be no surrounding atoms left to remove when jumping to that center element. This highlights the motivation for defining the atom-based indexing system which yields an injective mapping between the binary matrix and the graphene lattice cut pattern. However, using the center elements as a reference makes it easier to design the cut patterns since we can always go in one of the six directions defined by the center element neighbors. In contrast, the atom indexing system has alternating directions for its neighbors, which makes it more involved to define cut patterns.

2.5 Kirigami patterns

We propose a series of Kirigami-inspired cut patterns for the altering of the graphene sheet. We seek inspiration from macroscale patterns that showcase a considerable amount of out-of-plane buckling when stretched. We choose to imitate two different designs: 1) An alternating repeating series of perpendicular cuts as shown in Fig. 2.8a popularly used in studies of morphable metematerials [25]. This pattern produces surface buckling with a tetrahedron (three-sided pyramid) shape when stretched. 2) A more intricate pattern shown in Fig. 2.8b which is used commercially by Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [26] as protective wrap for items during shipping. This pattern buckles into a hexagonal honeycomb structure when stretched. In addition to the modeling of the so-called *Tetrahedron* and *Honeycomb* patterns, we also create a series of random walk patterns.

(a) **(b)**

Figure 2.8: Macroscale kirigami cut patterns used as inspiration for the nanoscale implementation. (a) Tetrahedron: Alternating perpendicular cuts producing a tetrahedron-shaped surface buckling when stretched. Reproduced from [25]. (b) Honeycomb: Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [26] producing a honeycomb-shaped surface buckling when stretched. Reproduced from [26].

2.5.1 Tetrahedron

The *Tetrahedron* pattern is defined in terms of center elements for which all atoms surrounding a given center element are removed. The pattern consists of two straight cuts, referred to as line 1 and line 2, that are arranged perpendicular to each other. The lines are positioned such that the center of one line aligns with the end of the other line, and with a given spacing in between (see Fig. 2.9). In order to achieve perpendicular cuts we cannot rely purely on the six center element directions corresponding to the center element neighbors which are spaced by 60°. We let line 1 run along the center elements in the direction of the "upper right" (and "lower left") center elements, while line 2 goes in the direction between the "down" and "lower right" ("up" and "upper left") center elements, corresponding to the direction $(1/\sqrt{3}, -1)$. We define variations of the pattern by the

number of center elements L_1 and L_2 in line 1 and 2 respectively, together with the spacing between the lines d, as the tuple (L_1, L_2, d). The pattern is constructed by translating the two lines to the whole sheet according to the spacing. Due to the alignment criteria of having one line point to the center of the other line, we can only allow an odd line length. Furthermore, in order to ensure that each center element is translated to an i-index of similar odd or evenness, we must in practice require that $|L_2 - L_1| = 2, 6, 10, \ldots$. Fig. 2.9 shows a visual representation of the pattern components for the $(7, 5, 2)$ patteren.

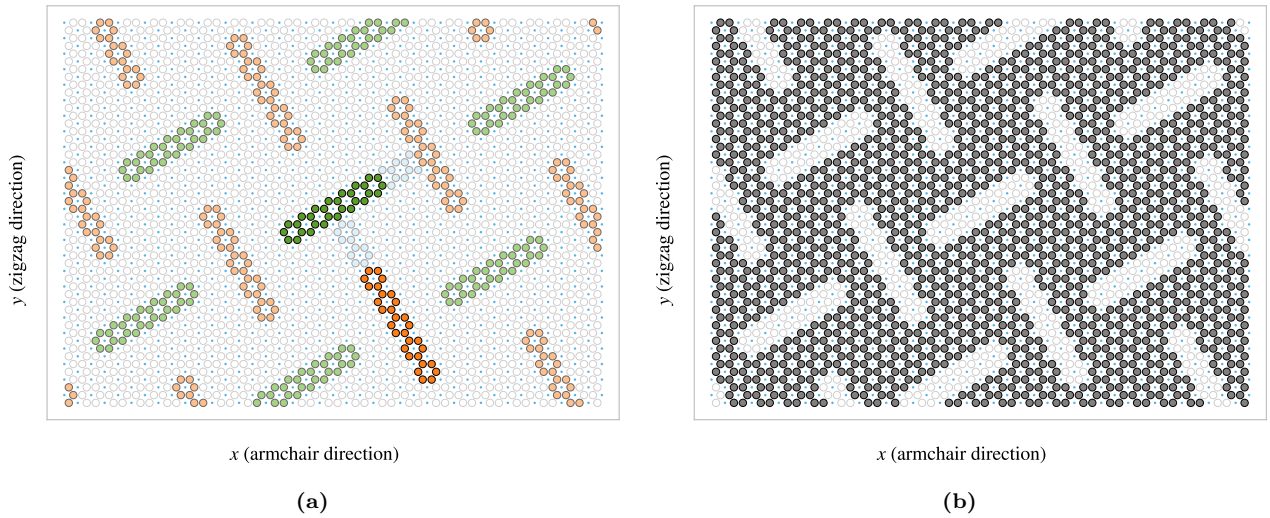


Figure 2.9: Visual representation of the Tetrahedron pattern consisting of two perpendicular lines, line 1 and line 2, of length L_1 and L_2 respectively, with spacing d. This example uses $(L_1, L_2, d) = (7, 5, 2)$ and a sheet matrix size 40×50 corresponding to 2000 atom sites and an approximate sheet size of 84×77 Å. The non-filled circles represent the possible atom site positions and the blue dots are the center elements. (a) Highlights the removed atoms in the pattern. Line 1 is shown in green and line 2 in orange, with lighter colors for the translated variations. The spacing is indicated in light blue. (b) The sheet after applying the cut pattern with grey circles denoting present atoms.

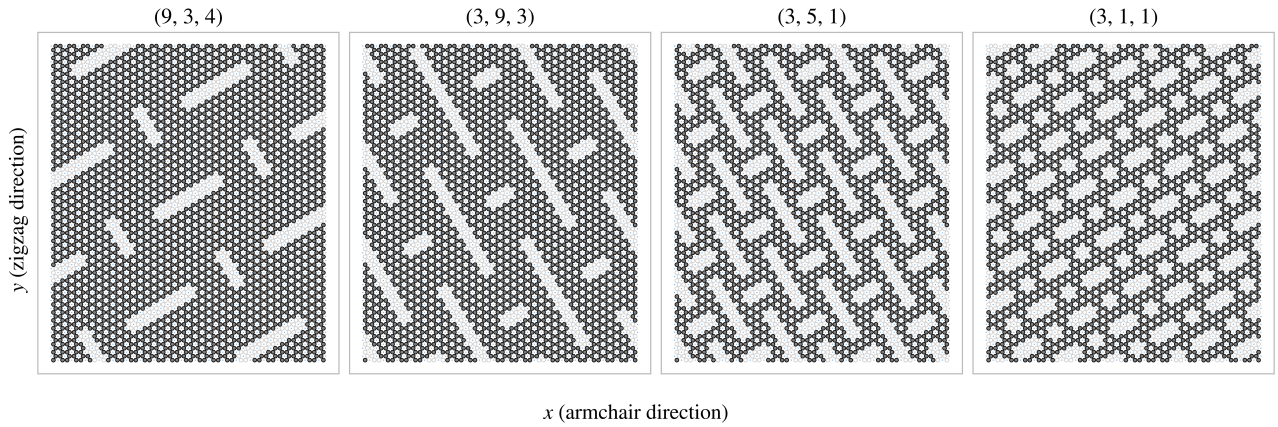


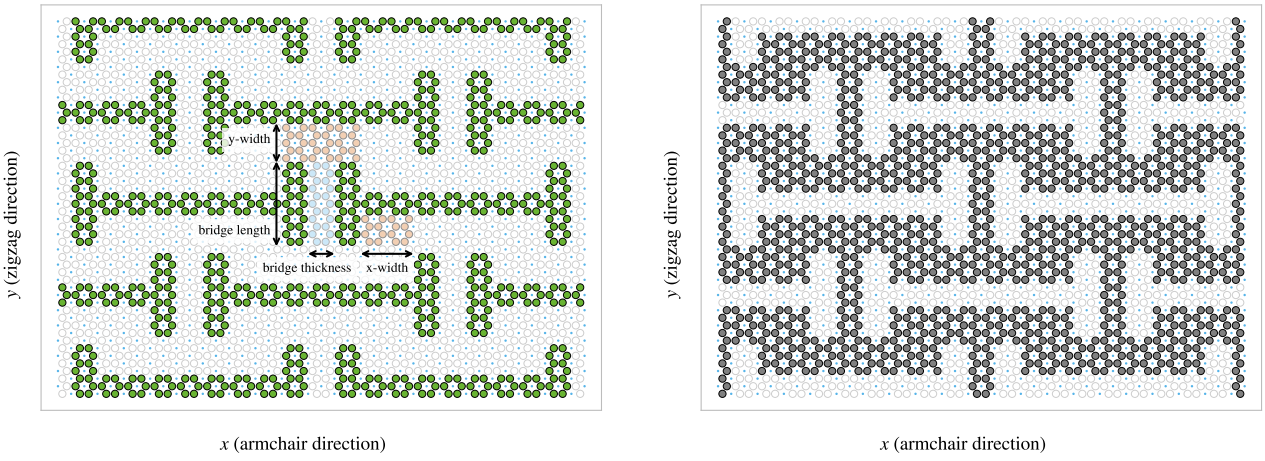
Figure 2.10: Example of different Tetrahedron cut pattern variations. The specific parameters are noted as titles and all of the patterns use the center of the sheet as a reference position. The circles in the figure represent atom sites, where grey-filled circles indicate the presence of atoms and transparent circles indicate removed atoms. The blue dots in the figure indicate the center elements. The sheet matrix size is 40×80 corresponding to 3200 atom sites and an approximate sheet size of 84×123 Å.

In addition to the three parameters L_1, L_2, d, the pattern is also anchored to a reference point that describes the position of line 1 and line 2 before being translated to span the sheet. Due to the repeating structure of the pattern, there exist a small finite number of unique reference positions. For the pattern $(7, 5, 2)$ used as an

example in Fig. 2.9, there are 140 unique reference points[2]. Some additional variation of the pattern is showcased in Fig. 2.10 each with a reference position at the center of the sheet. Note that a smaller sheet size than used in the simulations is used in both Fig. 2.9 and Fig. 2.10 for illustrative purposes.

2.5.2 Honeycomb

The *Honeycomb* pattern is defined, similarly to the Tetrahedron pattern, in terms of the center elements for which all surrounding atoms are removed. The Honeycomb pattern is built from a repeating series of cuts reminiscent of the Roman numeral one rotated by 90° (⊢). For a given spacing these are put next to each other in the x-direction, ⊢ ⊢ ⊢, to achieve a row where only a thin *bridge* in between is left to connect the sheet vertically in the y-direction. By placing multiple rows along the y-direction with alternating x-offset, we get the class of honeycomb patterns as visualized in Fig. 2.11. The pattern is described in terms of the parameters: (x-width, y-width, bridge thickness, bridge length) which is annotated in Fig. 2.11a with the parameters (2, 2, 1, 5) used as an example. Some additional variations of the pattern class are showcased in Fig. 2.12.



(a) **(b)**

Figure 2.11: Visual representation of the Honeycomb pattern defined by the (x-width, y-width, bridge thickness, bridge length) parameters as annotated in panel (a). This example uses the parameters $(2, 2, 1, 5)$ and a sheet matrix size 40×50 corresponding to 2000 atom sites and an approximate sheet size of 84×77 Å. The non-filled circles represent the possible atom site positions and the blue dots are the center elements. (a) Highlights the removed atoms in the pattern with annotations for the four defining parameters. (b) The sheet after applying the cut pattern with grey circles denoting present atoms.

[2]The general formula for calculating this number is rather complicated in comparison to its importance in this context. Therefore, we have omitted the formula and only provide the numerically backed result for the specific parameter set. The derivation of the formula was also deemed not to be rigorous enough.

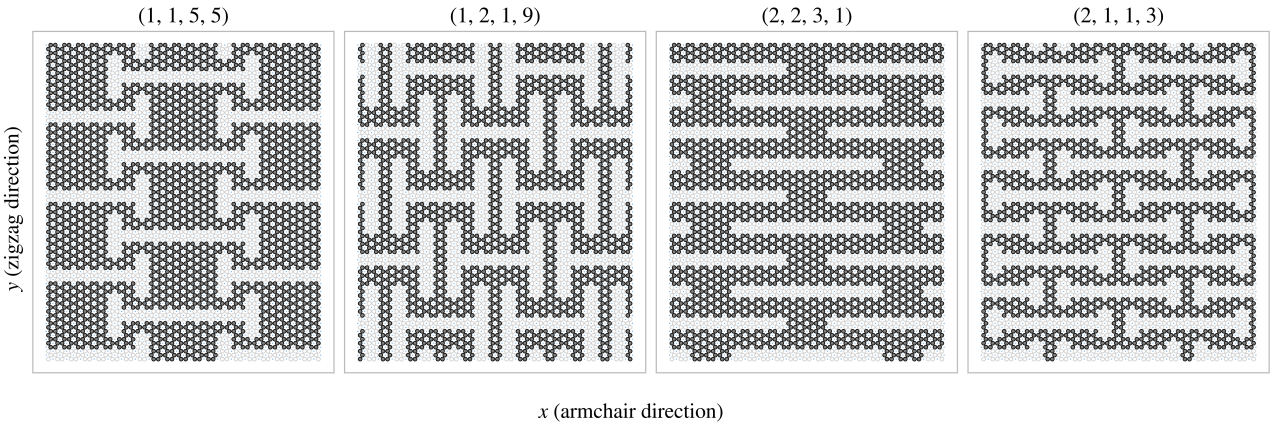


Figure 2.12: Example of different Honeycomb cut pattern variations. The specific parameters are noted as titles and all of the patterns use the center of the sheet as a reference position. The circles in the figure represent atom sites, where grey-filled circles indicate the presence of atoms and transparent circles indicate removed atoms. The blue dots in the figure indicate the center elements. The sheet matrix size is 40×80 corresponding to 3200 atom sites and an approximate sheet size of 84×123 Å.

2.5.3 Random walk

The random walk serves as a method for generating Kirigami patterns with randomized features. This approach is motivated by the aim of generating an ensemble of patterns that covers a larger region of the configuration space than the more structured patterns mentioned earlier. This is considered to be important for the quality of the dataset used for machine learning. By this argument, a straightforward way to create random configurations could be achieved simply by random noise, either uniform or Gaussian. However, this would often leave the sheet detached with numerous non-connected atom clusters. Intuitively, we do not find this promising for the generation of large-scale structures which we hypothesize to be of interest. The random walk pattern generation is characterized by the parameters summarized in Table 2.3 which will be introduced throughout the following paragraphs.

2.5.3.1 Fundamentals

For an uncut sheet, we deploy M random walkers, one at a time, and let them walk for a maximum number of S steps. We can either let the walker travel between atom sites, removing the atoms in the path as it goes, or between the center elements, removing all surrounding atoms. This is managed by setting the *connection* parameter to either *atom* or *center elements*. The method of removing only the intersecting atoms between center elements was also incorporated, but we ended up not using it due to plenty of other interesting options. Nonetheless, we will always remove a site once visited such that the walker itself, or any other walkers, cannot use this site again. This leads to a self-avoiding random walk, meaning that the walker does not cross its own path. However, it furthermore constrains the walkers to avoid the path taken by any previous walker, and thus we might denote this property as "other avoiding". By default, the walker has an equal chance of choosing any of its adjacent neighbors for the next step, i.e. we draw the next step from a discrete uniform distribution. Optionally, we can use periodic boundary conditions, by setting the parameter *periodic* to true, allowing neighboring sites to be connected through the edge in both the x and y-direction. When traveling on atom sites this ensures that we have three neighbor options for the next step while traveling on the center elements this gives six neighbor options. If the walker happens to arrive at an already visited site the walk is terminated early. Optionally, we can choose to remove any neighboring sites already visited from the neighbor list and choose uniformly between the remaining options instead. This is done by setting the parameter *avoid invalid* to true. This prolongs the walking distance, but the walker is still able to find itself in a situation where no neighboring sites are available. Note that the walker is not allowed to backtrack its own path either, and thus in such a case the walk will be terminated despite the setting of *avoid invalid*.

Table 2.3: Parameters for the random walk generator.

Parameter	Value	Description
Num. walkers (M)	Integer ≥ 1	Number of random walks to be initiated on the sheet (one at a time).
Max. steps (S)	Integer ≥ 1	The maximum steps allowed for any random walker.
Min. distance	Integer ≥ 0	The minimum distance required between any future paths and the previous paths in terms of the shortest walking distance in between.
Bias	(direction, strength ≥ 0)	Bias direction and strength defining the discrete probability for the choice of the next site.
Connection	Atoms / Center elements	Whether to walk between atom sites or center elements removing all adjacent atoms.
Avoid invalid	True/False	Whether to remove already visited sites from the neighbor list before picking the next site. This prevents jumping to already visited sites and lowers the likelihood of early termination.
Stay or break	$p = [0, 1]$	Probability that the walker will maintain its direction for the next step.
Periodic	True/False	Whether to use periodic boundary conditions on all four sides.
Avoid clustering	Integer ≥ 0	Amount of times to restart the whole random walk generation in order to arrive at a non-detached configuration. If no valid configuration is reached after this number of attempts, the non-spanning clusters are removed.
RN6	True/False	Randomly change the bias direction between the deployment of each random walker to one of the six center element directions.
Grid start	True/False	The option to have the random walkers start in an evenly spaced grid.
Centering	True/False	Relocate the path of a random walk after termination such that the path center of mass gets closer to the starting point (without violating the rules regarding already visited sites).

2.5.3.2 Spacing of walking paths

To control the spacing between the paths of the various walkers, we implement a so-called *minimum distance* parameter, taking integer values ≥ 0. This parameter describes the minimum spacing required between paths in terms of the least amount of walking steps. When a walker has ended its walk, either by early termination or hitting the maximum limits of steps, all sites within walking distance corresponding to the minimum distance parameter are marked as visited, although they are not removed from the sheet. This prevents any subsequent walkers to visit those sites in their walk according to the general behavior introduced in the previous paragraph. In practice, this is done through a recursive algorithm as described in Algorithm 1. For a given path the function *walk_distance()* is called with the input being a list of all sites in the given paths. The function gathers all the neighbors of each site, regardless of their state on the sheet. It then calls itself recursively using this neighbor list as input, while incrementing a distance counter that is also passed along as an argument. This results in an expansion along all possible outgoing paths from the initial path of interest. Once the distance limit is reached, the function returns the final neighbor lists, which are then accumulated into a final output. This output corresponds to a list of all sites within the minimum distance to the path.

Algorithm 1 Recursive algorithm implemented as a class method of the random walk generator. For a given path input it flags all sites within a distance given by the class attribute *self.min_dis*.

```
Require: self.min_dis > 0
function WALK_DISTANCE(self, input, dis = 0, pre = [ ])
    new_neigh ← [ ]                                   ▷ Initialize list for new neighbors
    for site in input do
        neigh ← get_neighboring_sites(site)           ▷ Get surrounding neighbors
        for n in neigh do
            if (n not in pre) and (n not in new_neigh) then   ▷ If not already added
                AddItem(new_neigh, n)                 ▷ then add the site
            end if
        end for
    end for
    dis += 1                                          ▷ Increment distance counter
    if dis ≥ self.min_dis then                        ▷ Max limit hit
        return input + new_neigh
    else                                              ▷ Start a new walk from each of the neighboring sites
        pre ← input
        return pre + self.walk_distance(new_neigh, dis, pre)
    end if
end function
```

2.5.3.3 Bias

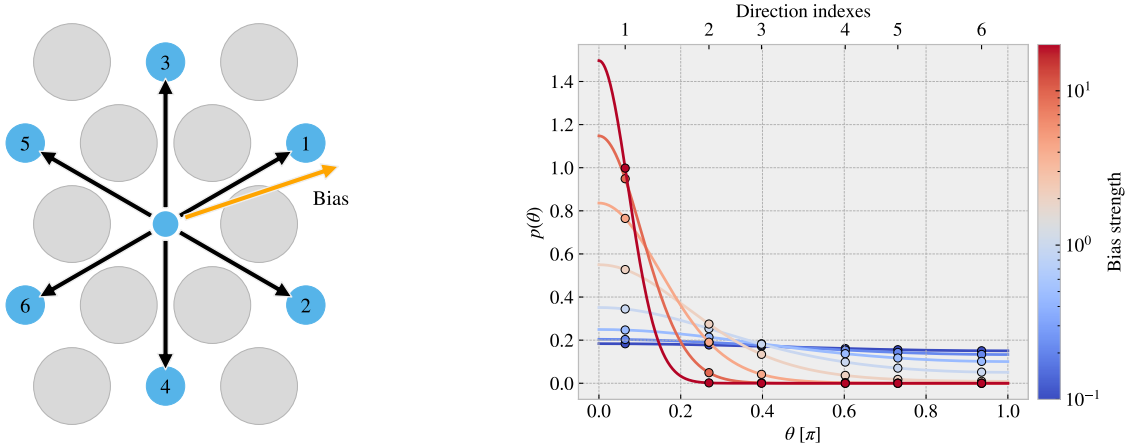
We provide the option to perform a biased random walk by specifying the bias parameter, which consists of a direction and a strength. To achieve this, we model each step of the walk analog to a system in the canonical ensemble under the influence of an external force $\mathbf{F}$ representing the bias. For such a system each microstate i, corresponding to the sites in the neighbor list, has the associated probability p_i given by the Gibbs–Boltzmann distribution

$$p_i = \frac{1}{Z} e^{-\beta E_i}, \qquad Z = \sum_i e^{-\beta E_i},$$

where Z is the canonical partition function, $\beta = 1/k_B T$ for the boltzmann constant k_B and temperature T, and E_i the energy of site i. We model the energy of each site as the work required to move there. For a step $\mathbf{s}$ the energy becomes $E_i = -\mathbf{s} \cdot \mathbf{F}$, where the sign is chosen such that the energy (difference) is negative when moving along the bias, analogous to an energy gain by moving there. Due to the symmetry of the random walk sites, both for the atom sites and the center elements, the step length to neighboring sites will always be equal. By defining the bias strength $B = \beta |\mathbf{F}||\mathbf{s}|$ we get that the probability for jumping to site i is

$$p_i = \frac{1}{Z} e^{B \hat{\mathbf{s}} \cdot \hat{\mathbf{F}}} \propto e^{B \hat{\mathbf{s}} \cdot \hat{\mathbf{F}}},$$

where the hat denotes the unit vector. The bias strength B then captures the opposing effects of the magnitude of the external force and the temperature of the system since $B \propto |\mathbf{F}|/T$. We notice that $\hat{\mathbf{s}} \cdot \hat{\mathbf{F}} = \cos(\theta)$ for the angle θ between the step and bias direction. This shows that the bias will have the biggest positive contribution to the probability when the step direction is fully aligned with the bias direction ($\theta = 0$), have no contribution for orthogonal directions ($\theta = \pm\pi/2$) and the biggest negative contribution when the directions are antiparallel ($\theta = \pi$). The partition function serves as a normalization constant. Thus, numerically we can enforce this simply by setting $Z = 1$ at first, calculating p_i, and then normalizing the result at the final stage as a division by the sum of all p_i. In the numerical implementation, we then pick the next step by the weighted discrete probability distribution p_i. In Fig. 2.13 we have illustrated how a bias of different strengths impacts the probability distribution for a random walk between center elements. We can visually confirm that the bias will favor sites that lie close to the bias direction. This preference is more distinct at high bias strengths while at low strength $B \to 0$ we get a uniform distribution that aligns with the default unbiased random walk.



(a) **(b)**

Figure 2.13: Illustration of the probability distribution for the various step directions during a biased random walk between center elements. (a) The possible step directions are represented by black arrows that point towards the neighboring center elements depicted as blue circles. The bias direction is denoted by an orange arrow, and the numbering indicates the most probable direction (1) towards the least probable direction (6). The atom sites are marked as grey circles for reference. (b) The probability distribution as a function of the angle θ between the step direction and the bias direction. The distribution is normalized according to the discrete probabilities marked with dots for which the continuous line simply highlights the shape of the distribution. The direction indexes correspond to the numbering on panel (a). The color map indicates different strengths of the bias.

2.5.3.4 Stay or break

The *stay or break* parameter defines the probability p_{stay} that the walker will keep its direction or otherwise break into a different direction by probability $1 - p_{\text{stay}}$. We implement this by altering the discrete probability used for the choice of the next step. We manually set the probability to p_{stay} for the site corresponding to a continuation in the same direction and renormalize the distribution. This allows us to perform a biased random walk in combination with a preference for keeping direction. For the center element walk it is trivial to determine which of the neighbor directions correspond to a continuation of direction based on the last visited site. However, for an atom site walk, it is not possible to follow the same direction in a straight line due to the hexagonal layout of the lattice. We recall that the nearest atom neighbor indexes alternate for each increment in the x or y index (see Eq. (2.3)) which corresponds to the alternating neighbor directions D as

$$(i+j) \text{ is even} \rightarrow D = \left\{ \frac{a}{2}\left(\frac{-2}{\sqrt{3}}, 0\right), \frac{a}{2}\left(\frac{1}{\sqrt{3}}, 1\right), \frac{a}{2}\left(\frac{1}{\sqrt{3}}, -1\right) \right\},$$
$$(i+j) \text{ is odd} \rightarrow D = \left\{ \frac{a}{2}\left(\frac{2}{\sqrt{3}}, 0\right), \frac{a}{2}\left(\frac{-1}{\sqrt{3}}, 1\right), \frac{a}{2}\left(\frac{-1}{\sqrt{3}}, -1\right) \right\}.$$

One way to mitigate this issue is to use the six directions from the center element walk as the common direction to "stay or break" from. As showcased in Fig. 2.14, for each center element direction (black arrows) there are two possible atom site directions (red and orange arrows) that are equally close to the center element direction. The red and orange arrows represent $(i+j)$ being even or odd respectively, and we notice that these appear in pairs such that we can always uniquely determine which of the atom directions is closest to the center element direction. Following this idea we can map each center direction to an atom direction depending on the even or oddness of the position. For $p_{\text{stay}} = 1$ this results in a guaranteed zigzag motion along the center element direction that it happens to start on.

The *stay or break* feature is still subject to previously defined rules. For instance, in the case where the preferred site is not available, the walker will either terminate when going there, or the preferred site is removed from the neighbor list when *avoid invalid* is set to true. In the latter case, the walker will be forced to break out of its direction and follow the new direction that it happens to choose.

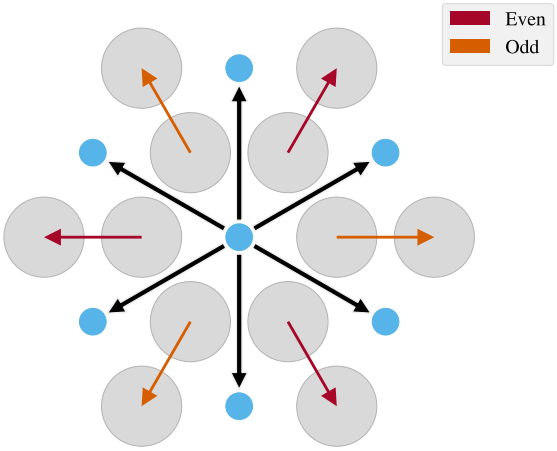


Figure 2.14: Visualization of the center element directions (black arrows) connecting the center elements depicted as blue dots, and the atom site directions (red and orange arrows) connection the atom sites depicted as grey circles. The red arrows correspond to the case where the sum of the atom site indexes (i, j) is even, while the orange arrows correspond to the case where the sum is odd. Notably, for each case, there is always one atom site direction that is uniquely closest to a given center element direction.

2.5.3.5 Deployment schemes

By default, each random walker is given a uniform random starting point among the non-visited available sites left on the sheet. This includes any modifications in relation to the minimum distance parameter. By setting the *grid start* parameter to true, the starting points are instead predefined on an evenly spaced grid. That is, the sheet is subdivided into the least amount of squares that will accommodate space for each starting point. $\{1\}$ walker leads to a 1×1 partition, $\{2, 3, 4\}$ walkers lead to a 2×2 partition, $\{5, 6, 7, 8, 9\}$ walker lead to a 3×3 partition and so on. For each partition square, the starting point is placed as centrally as possible. The lower left partition square is then chosen as a default starting place for the first walker and the remaining sites are filled according to the order that maximizes the minimum distance between a new starting point and the ones already occupied[3]. An example of the deployment is shown in Fig. 2.15. Notice, that if the planned grid start position is made invalid before deployment, it is skipped.

The *centering* parameter lets us relocate the path of the random walker such that the path center of mass gets closer to the starting point. When set to true, the path is moved in the direction defined from the center of mass toward the starting point for which the closest valid relocation on the direct translation line is chosen. This can be used in combination with the *grid start* and the *bias parameter* to make rather ordered configurations. In addition, the *RN6* parameter can be used to update the bias direction to one of the six center element directions for each new walker deployed. This lets us create more organized configurations like the one shown in Fig. 2.16b.

2.5.3.6 Validity

The simulation procedure requires the sheet to be fully attached, non ruptured, which can be summarized as the following requirements.

1. There exists only a single cluster on the sheet. We define a cluster as the set of atoms which can all be reached through nearest neighbor walking on the cluster.

2. The cluster of atoms is spanning the sheet in the y-direction. This means that there exists at least one path through nearest neighbor walking that connects the bottom and the top of the sheet. This is required because the sheet must be attached to the pull blocks in the simulation.

In order to accommodate these requirements we count the number of clusters and search for a spanning cluster after all walkers have terminated. If the requirements are not met we simply rerun the random walk from scratch.

[3] In hindsight, we realize that it would have been less biased to choose a random partition square as the starting one, but we do not consider this to be of great importance for the usage of this feature in the dataset.

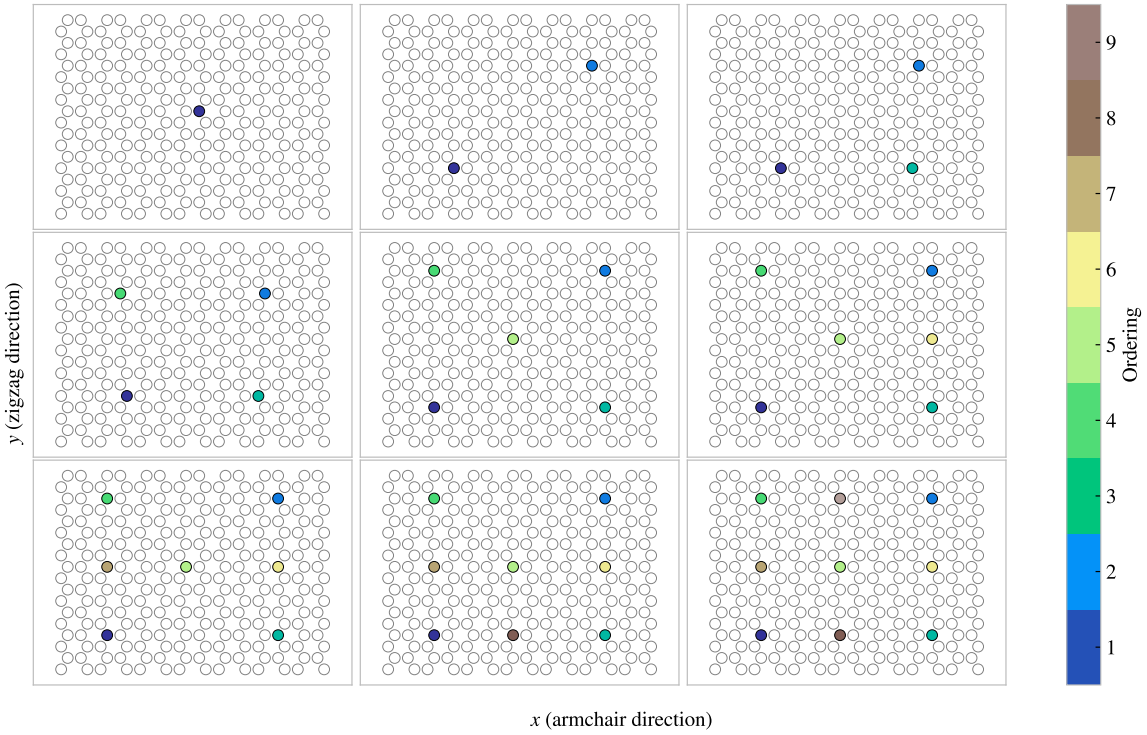


Figure 2.15: The figure illustrates the distribution of starting points when the *grid start* parameter is set to true for a 14×18 sheet, for varying numbers of deployed random walkers ranging from 1 to 9. The color map is used to indicate the order in which the walkers were deployed.

This is done according to the *avoid clustering* parameter which takes integer values corresponding to the number of times to repeat this process. If the requirements are not met during any of those reruns the non-spanning clusters are simply removed. In the case of no spanning cluster, the configuration is skipped. This crude scheme was later reinvented as a more refined repair scheme that alters the sheet with the intention of performing the least amount of changes, addition or subtraction of atoms, in order to meet the attachment requirements. This was done as a part of the accelerated search procedure and hence it was not utilized in the creation of the random walk dataset. However, we give a brief description of the algorithm here:

1. Find all clusters and rank them in descending order of size, such that the largest cluster is labeled as cluster 1.
2. Deploy walkers from the edges of the smallest cluster, allowing them to walk in all possible directions similar to what was done with Algorithm 1. Here the allowed walking distance is defined as the number of atoms within the starting cluster.
3. If a walker from the starting cluster reaches another cluster within the allowed walking distance, these two clusters are connected through the walking path. Otherwise, the starting cluster is deleted. This decision is based on the consideration that the number of atoms required to connect the clusters should not exceed the number of atoms removed if the starting cluster is deleted. However, if the starting cluster is the last cluster connected to the top or bottom edge, an unlimited number of walks are allowed to ensure the creation of a spanning cluster.
4. If there is only one cluster remaining on the sheet, the repair procedure is complete. If there are still multiple clusters, then repeat the steps from 1. Note, that if no spanning cluster was ever present, we expand the final cluster until a spanning connection can be established.

2.5.3.7 Random walk examples

Some examples of the random walk patterns are illustrated in Fig. 2.16 with corresponding parameters in Table 2.4.

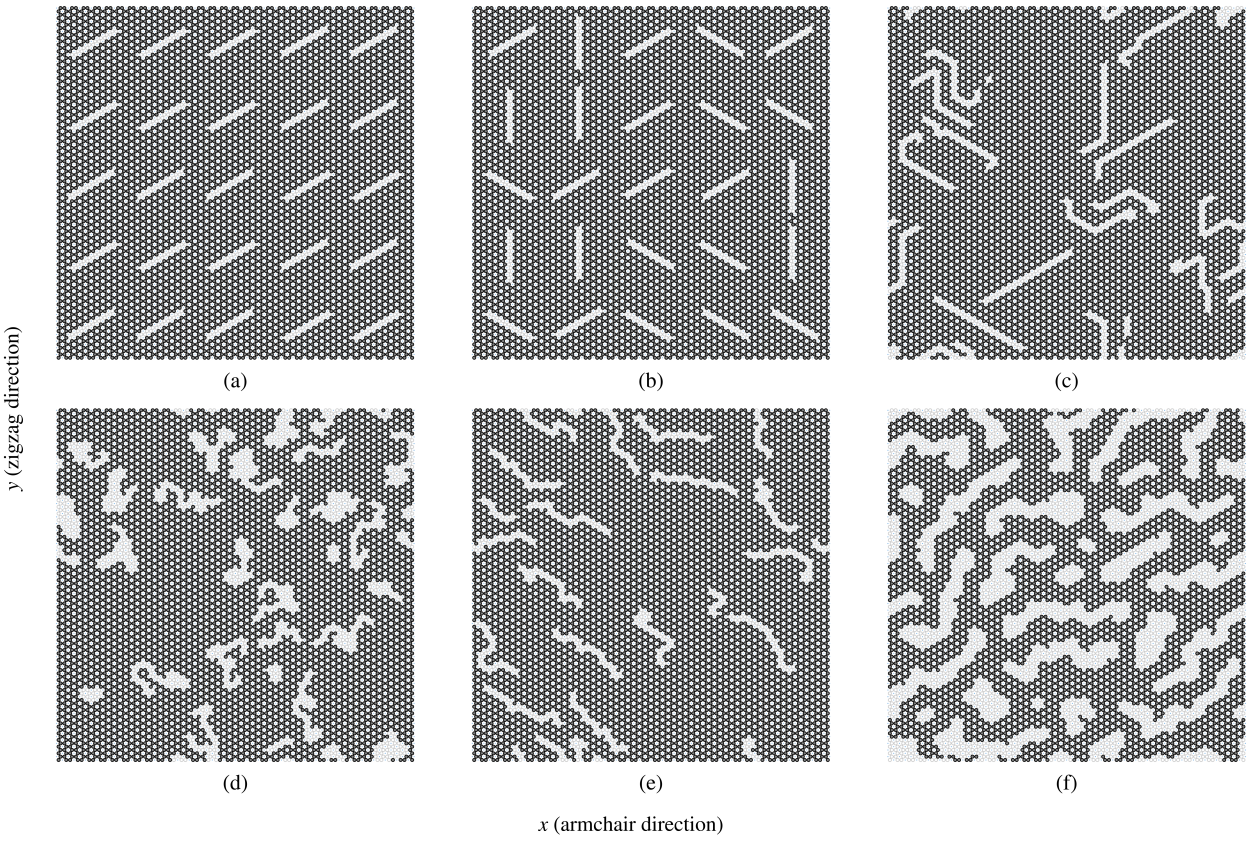


Figure 2.16: Example of different Random walk cut pattern variations. The specific parameters are given in Table 2.4. The circles in the figure represent atom sites, where grey-filled circles indicate the presence of atoms and transparent circles indicate removed atoms. The sheet matrix size is 62×106 corresponding to the full-size system used in the MD simulations with 6572 atom sites and an approximate sheet size of 130×163 Å.

Table 2.4: Parameters for the random walk patterns shown in Fig. 2.16.

Fig. num.	(a)	(b)	(c)	(d)	(e)	(f)
Num. walkers	25	25	20	30	20	32
Max. Steps	15	15	30	40	30	30
Min. distance	0	0	4	4	4	4
Bias	$\begin{pmatrix}\text{upper right}\\ 100\end{pmatrix}$	$\begin{pmatrix}\text{upper right}\\ 100\end{pmatrix}$	None	None	$\begin{pmatrix}\text{lower right}\\ 1.2\end{pmatrix}$	$\begin{pmatrix}\text{lower left}\\ 1.2\end{pmatrix}$
Connection	Atoms	Atoms	Atoms	Atoms	Atoms	Center elements
Avoid invalid	False	False	True	True	True	True
Stay or break	0	0	0.9	0	0	0
Periodic	True	True	True	True	True	True
Avoid clustering	10	10	10	10	10	10
RN6	False	True	True	False	False	False
Grid start	True	True	False	False	False	False
Centering	True	True	False	False	False	False

Appendices

Bibliography

[1] E. Gnecco and E. Meyer, *Elements of friction theory and nanotribology* (Cambridge University Press, 2015).

[2] Bhusnan, "Introduction", in *Introduction to tribology* (John Wiley & Sons, Ltd, 2013) Chap. 1.

[3] H.-J. Kim and D.-E. Kim, "Nano-scale friction: a review", International Journal of Precision Engineering and Manufacturing **10**, 141–151 (2009).

[4] K. Holmberg and A. Erdemir, "Influence of tribology on global energy consumption, costs and emissions", Friction **5**, 263–284 (2017).

[5] B. Bhushan, "Gecko feet: natural hairy attachment systems for smart adhesion – mechanism, modeling and development of bio-inspired materials", in *Nanotribology and nanomechanics: an introduction* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), pp. 1073–1134.

[6] P. Z. Hanakata, E. D. Cubuk, D. K. Campbell, and H. S. Park, "Accelerated search and design of stretchable graphene kirigami using machine learning", Phys. Rev. Lett. **121**, 255304 (2018).

[7] P. Z. Hanakata, E. D. Cubuk, D. K. Campbell, and H. S. Park, "Forward and inverse design of kirigami via supervised autoencoder", Phys. Rev. Res. **2**, 042006 (2020).

[8] L.-K. Wan, Y.-X. Xue, J.-W. Jiang, and H. S. Park, "Machine learning accelerated search of the strongest graphene/h-bn interface with designed fracture properties", Journal of Applied Physics **133**, 024302 (2023).

[9] Y. Mao, Q. He, and X. Zhao, "Designing complex architectured materials with generative adversarial networks", Science Advances **6**, eaaz4169 (2020).

[10] Z. Yang, C.-H. Yu, and M. J. Buehler, "Deep learning model to predict complex stress and strain fields in hierarchical composites", Science Advances **7**, eabd7416 (2021).

[11] A. E. Forte, P. Z. Hanakata, L. Jin, E. Zari, A. Zareei, M. C. Fernandes, L. Sumner, J. Alvarez, and K. Bertoldi, "Inverse design of inflatable soft membranes through machine learning", Advanced Functional Materials **32**, 2111610 (2022).

[12] S. Chen, J. Chen, X. Zhang, Z.-Y. Li, and J. Li, "Kirigami/origami: unfolding the new regime of advanced 3D microfabrication/nanofabrication with "folding"", Light: Science & Applications **9**, 75 (2020).

[13] OpenAI, *Dall-e2*, (2023) `https://openai.com/product/dall-e-2`.

[14] Midjourney, *Midjourney*, (2023) `https://www.midjourney.com`.

[15] Z. Deng, A. Smolyanitsky, Q. Li, X.-Q. Feng, and R. J. Cannara, "Adhesion-dependent negative friction coefficient on chemically modified graphite at the nanoscale", Nature Materials **11**, 1032–1037 (2012).

[16] B. Liu, J. Wang, S. Zhao, C. Qu, Y. Liu, L. Ma, Z. Zhang, K. Liu, Q. Zheng, and M. Ma, "Negative friction coefficient in microscale graphite/mica layered heterojunctions", Science Advances **6**, eaaz6787 (2020).

[17] D. Mandelli, W. Ouyang, O. Hod, and M. Urbakh, "Negative friction coefficients in superlubric graphite–hexagonal boron nitride heterojunctions", Phys. Rev. Lett. **122**, 076102 (2019).

[18] R. W. Liefferink, B. Weber, C. Coulais, and D. Bonn, "Geometric control of sliding friction", Extreme Mechanics Letters **49**, 101475 (2021).

[19] M. Metzsch, *Github repository*, `https://github.com/mikkelme/MastersThesis`.

[20] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales", Comp. Phys. Comm. **271**, 108171 (2022).

[21] E. M. Nordhagen, *LAMMPS simulator*, https://github.com/evenmn/lammps-simulator.

[22] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a python library for working with atoms", Journal of Physics: Condensed Matter **29**, 273002 (2017).

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: an imperative style, high-performance deep learning library", in *Advances in neural information processing systems 32* (Curran Associates, Inc., 2019), pp. 8024–8035.

[24] D. Gray, A. McCaughan, and B. Mookerji, "Crystal structure of graphite, graphene and silicon", Physics for Solid State Applications **6**, 730 (2009).

[25] L. Burrows, *New pop-up strategy inspired by cuts, not folds*, (Feb. 24, 2017) https://seas.harvard.edu/news/2017/02/new-pop-strategy-inspired-cuts-not-folds.

[26] *Scotch cushion lock protective wrap*, https://www.scotchbrand.com/3M/en_US/scotch-brand/products/catalog/~/Scotch-Cushion-Lock-Protective-Wrap/?N=4335+3288092498+3294529207&rt=rud.