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

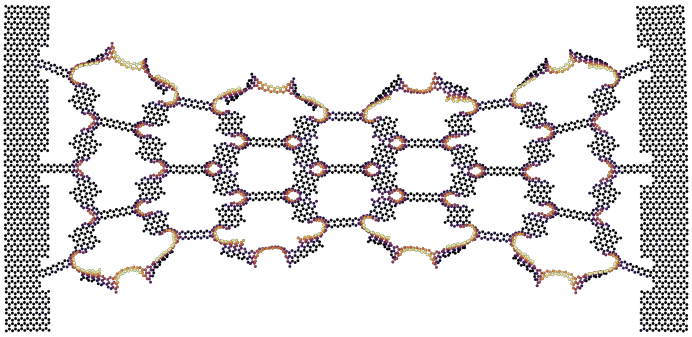
UNIVERSITY OF OSLO

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

Abstract.

Acknowledgments

Acknowledgments.

List of Symbols

F_N Normal force (normal load)

Acronyms

CM Center of Mass. 12

FFM Friction Force Microscopes. 9

MD Molecular Dynamics. 1, 2, 3, 9

ML Machine Learning. 2, 3

NN Nearest neighbours. 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. Even though the everyday person might not be familiar with the term *friction* we recognize it as the inherent resistance to sliding motion. Some surfaces appear slippery and some 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 [p. 5] [1]. 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" and 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, reducing friction is of great interest as this has tremendous advantages for energy efficiency. It has been reported that tribological problems have a significant potential for 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 are 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 geckos can produce large adhesive forces, they retain the ability to remove their 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 origin 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 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 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 make for seemingly intractable problems ruling out analytic solutions.

Earlier architecture 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 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 accordingly to the most promising candidates, for instance, as done with the *genetic algorithm* based on mutation and crossing of the best candidates so far. Another more sophisticated approach is through generative methods such as *Generative Adversarial Networks* (GAN) or diffusion models with the latter being used in state-of-the-art AI systems such as OpenAI's DALL·E2 or Midjourney SOURCE?. By working with a so-called *encoder-decoder* network structure, one can build a model that reverses the prediction process. That is, the model predicts a design from a set of physical target properties. In the papers by Hanakata et al. 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 variety in yield 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 behavior 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 stretch 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 theory also take on negative values. The latter would constitute a rarely found property which is mainly found for the unloading phase of adhesive surfaces [13] or for the loading phase of particular heterojunction materials [14, 15]

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. [16] 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 regime would involve a different set of underlying mechanisms and thus contribute to new insight in this field.

1.2 Goals

In this thesis, we investigate the possibility to alter 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) simulations, we evaluate the frictional properties of various Kirigami designs under different physical conditions. With the use of machine learning (ML), we perform an accelerated search of designs to explore new designs. The main goals of this thesis can be summarized as follows.

1. Design a robust MD simulation procedure to evaluate the frictional properties of a Kirigami graphene sheet under specified physical conditions.
2. Develop a numerical framework for creating various Kirigami designs, both by seeking inspiration from macroscale designs and by the use of stochastically based algorithms.
3. Investigate the friction behavior under varying load and stretch for different Kirigami designs.
4. Develop and train a ML model to predict the MD simulation result and perform an accelerated search of new designs for the scope of optimizing certain frictional properties.

1.3 Contributions

What did I actually achieve Include Githib link

1.4 Thesis structure

In Part I: Background Theory, we cover the theoretical background related to Friction (**??**), Molecular Dynamics (**??**) and Machine Learning (**??**).

In **??**: Friction, we introduce the most relevant theoretical concepts of friction through a division by scale: Macroscale (**??**), Microscale (**??**) and nanoscale (**??**). We emphasize the nanoscale since this is of the most importance for our study. This is followed by a summary of relevant experimental and numerical results **??** and a more formal specification of our research questions (**??**).

In **??**: Molecular Dynamics, we introduce the main concepts related to the simulations used in this thesis. The main parts involve a description of the potentials used (**??**), the numerical solutions (**??**) and the modeling of temperature (**??**)

In **??**: Machine Learning, we introduce the basics of machine learning through a general presentation of the neural network **??** followed by the convolutional network (**??**) which we will use in our study. Additionally, we discuss a strategy for choosing model hypertuning (**??**) and a simple approach for model prediction explanations (). Finally, we introduce a version of the genetic algorithm applicable for accelerated search based on a machine learning model (**??**).

In Part II: Simulations, we define our numerical procedure and present and discuss the main findings of this thesis.

In Chapter 2: System, we ...
In **??**: Pilot study, we ...
In **??**: XXX ...
In **??**: XXX, ...
The thesis is summarized in **??**

Additional figures are shown in **??**, **??** and **??**. get appendix with only letter A., B. and C.

Part I

Background Theory

Part II

Simulations

Chapter 2

Creating a graphene kirigami system

The system definition 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. Thus we aim to design the simplest possible system that allows such a measurement for different variations on Kirigami design, stretch 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. This setup allows for a variety of tip designs connected to the moving body, and alternatively, the tip can be substituted with a flat surface making the setup resemble a SFA experiment instead. For this setup, we would have to attach a pre-stretched sheet to the substrate and require the edges to be fixated in order to ... the stretch sustain the stretch. We would essentially regard the sheet as a part of the substrate, and the possible applications would regard to certain effects being associated constant stretch. Another approach is to have the sheet ends fixated on the moving body instead. This allows for the connection to a nanomachine design which couples the load and the stretch of the sheet. Thus, the possible applications allow for a dynamic effect with changing stretch 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 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 stretch to motivate our study. 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.

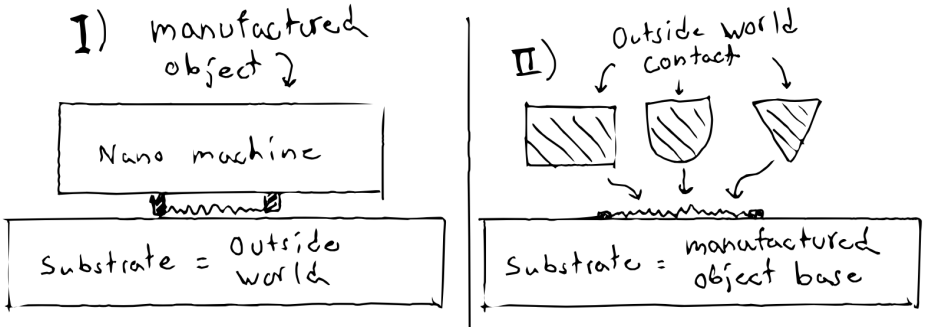


Figure 2.1: TMP System variations

2.1 Region definitions

We subdivide the two main parts of the simmulation, the sheet and the substrate, into specific regions according their functionality in the MD simmulations. 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 and dragging of the sheet, and for applying the thermostat. The remaining *inner sheet* is left for the kirigami cuts

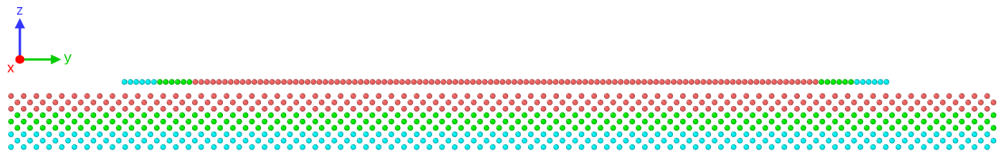
and are simulated as an NVE ensemble. The pull blocks are equally split between a thermostat part and a rigid part. The rigid part is however thermolized during the initial relaxtion period but made rigid for the final duration of the simulation. Note that the rigid parts on both sides of the sheet is then considered as a single rigid object even though they are physically seperated. This means that all force interactions concerning the rigid parts will be applied as a common average making them move in total synchronization. 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. In Fig. 2.2 the system is displayed with colors matching the three distinct roles:

1. Red: NVE parts which is governing the frictional behaviour of interest.

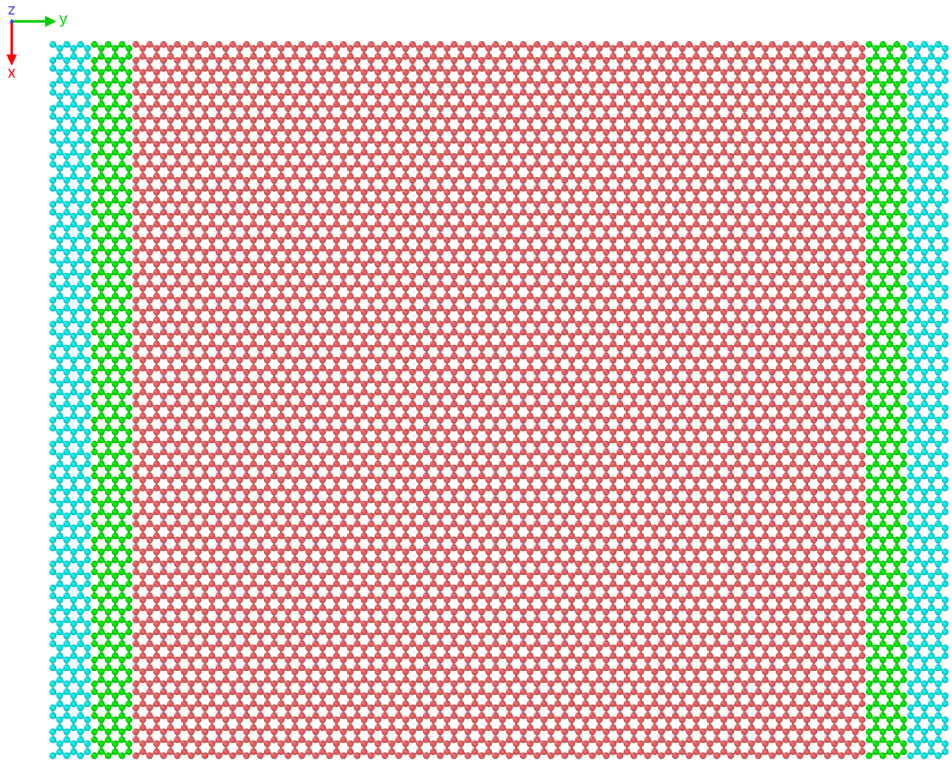
2. Green: Thermostats (NVT) sourrounding the NVE parts in order to modify the temperature without making disturbing changes to the interaction of the sheet and substrate.

3. Blue: Parts that are initially or eventually turned in to rigid objects. For the substrate this refers to an addtionally 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 stretch of 200% the total system size is roughly 55k atoms. The specific distribution is shown in Table 2.2 along with the spatial x-y-measures in Table 2.1.



(a) Side view showing sheet on top of the substrate.



(b) Top view showing only the sheet.

Figure 2.2: System configuration colorized to indicate NVE parts (red), thermostat parts (green) and rigid parts (blue).

Table 2.1: Specification of the spatial size of the system for the x-y-dimensions with a substrate scaled for an expected stretch 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 × 130.029 × 15.183 Å	2 × 19.74
Substrate	$1.16\ x_S \times 3.12\ y_S$	150.709 × 509.152 Å	767.34

Table 2.2: Specification of the system size in number of atoms for various system regions. These numbers corresponds with the case of no cuts applied to the sheet and a substrate scaled for the expected stretch of 200 %.

Region	**Total**	Sub region	Sub total	**NVE**	**NVT**	**Rigid**
Full sheet	7800	Inner sheet	6360	6360	0	0
		Pull blocks	1440	0	720	720
Substrate	47376	Upper	15792	15792	0	0
		Middle	15792	0	15792	0
		Bottom	15792	0	0	15792
All	55176			22152	16512	16512

2.2 Numerical procedure

After implementing the initial configuration to the system we first need to let the system relax and reach a stable equilibrium state. This involves an adjustment to the system temperature and the sheet-substrate distance. We then stretch the sheet to the desired length before applying normal load the pull blocks. 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 $10^5\,\mathrm{eV/Å}^2 \sim 1.6 \times 10^6\,\mathrm{N/m}$: 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, it would be sufficient to fixate only one of the pull blocks, but we fixate both for the sake of 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 introduced to allow the sheet pull blocks to readjust the lattice spacing according to the temperature of the system. For the following phases, the pull blocks are made truly rigid with respect to all directions, and the spring forces are terminated.

2. **Stretch**: The sheet is stretched by seperating the two opposing rigid parts of the pullblock at constant velocity until the desired stretch amount is met. The duration of this phase is thus governed by the *stretch speed* and *stretch amount* parameters.

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

4. **Normal load** (5 ps): A normal load is applied to the rigid parts of the pull blocks. Initially a viscous damping force, $F = -\gamma\mathbf{v}$, is added to the sheet to resist the rapid acceleraction of the sheet and prevent a hard impact between the sheet and substrate. 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 simply devoted for further relaxation.

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 variable spring constant K in the x-y-plane. The z-direction is not affected by the spring force and is purely governed by the balancing forces of the normal load and the normal response from the sheet-substrate interaction. The virtual atom is immediately given a constant velocity, in accordance with the *sliding speed* parameter, which makes the sliding force increase linearly, $F_{slide} \propto K v_{\mathrm{slide}} t$, with sliding speed. 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 according to the sliding speed.

To limit the complexity of our prediction task, 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. The substrate was proven to be way more resistant to wear than the sheet, and by running a few test simulations of high load and sliding speed we confirmed visually that no wear is occurring.

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 thus half the cut-off distance for the Tersoff potential, governing the graphene sheet interaction, at $R + D = 2.1$ Å. However, due to fluctuations in the sheet, we cannot set the margin too close to that limit. Additionally, we must take into account the buckling of the sheet as it is stretched, which might induce an expansion in the x-direction for certain configurations. We choose a 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 extreme buckling. For the y-direction the rigid parts of the pull-blocks moves a certain distance based on the stretch 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 $20\,\text{m/s} = 2\,\text{Å/ps}$ gives 15 ps of relaxation time between encounters with the sheet, similar to the initial relaxation time.

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 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

$$\mathbf{a_1} = a\left(\frac{\sqrt{3}}{2}, -\frac{1}{2}\right), \qquad \mathbf{a_2} = a\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right), \qquad |\mathbf{a_1}| = |\mathbf{a_2}| = 2.46\ \text{Å}.$$

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 consist of two carbon atoms given as

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

The crystal structure is visualized in Fig. 2.3. It turns out that the spacing between atoms is equal for all pairs of atoms with an interatomic distance

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

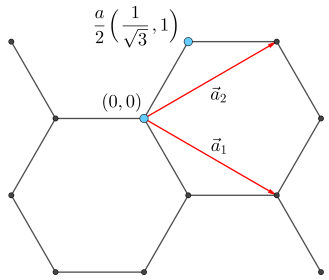


Figure 2.3: Graphene crystal structure with basis.

2.4.1 Indexing

In order to define the kirigami cut patterns applied to the graphene sheet we need to define an indexing system. We must ensure that this gives a unique description of the atoms as we eventually want to pass a binary matrix, containing 0 for removed atoms and 1 for present atoms, that uniquely describes the sheet. We do this by letting the x-coordinate allign with the so-called *armchair* direction of the sheet and making the y-coordinate increment along the so-called *zigzag* direction. Notice that the x-coordinate will point to *zigzag* chains of atoms for which the starting point, at $y = 0$ is not evenly spaced as illustrated in figure Fig. 2.4. 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 willl be actual nearest neighbors in the lattice. While $(i, j \pm 1)$ is always a 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 are decided as

$$\begin{aligned} (i+j) \text{ is even} &\rightarrow \text{NN} = \{(i-1, j), (i, j+1), (i, j-1)\}, \\ (i+j) \text{ is odd} &\rightarrow \text{NN} = \{(i+1, j), (i, j+1), (i, j-1)\}. \end{aligned} \tag{2.1}$$

By consulting Fig. 2.4 we can verify this visually, and it basically comes down to the fact whether the atom is oriented to the right or the left side in the zigzag chain.

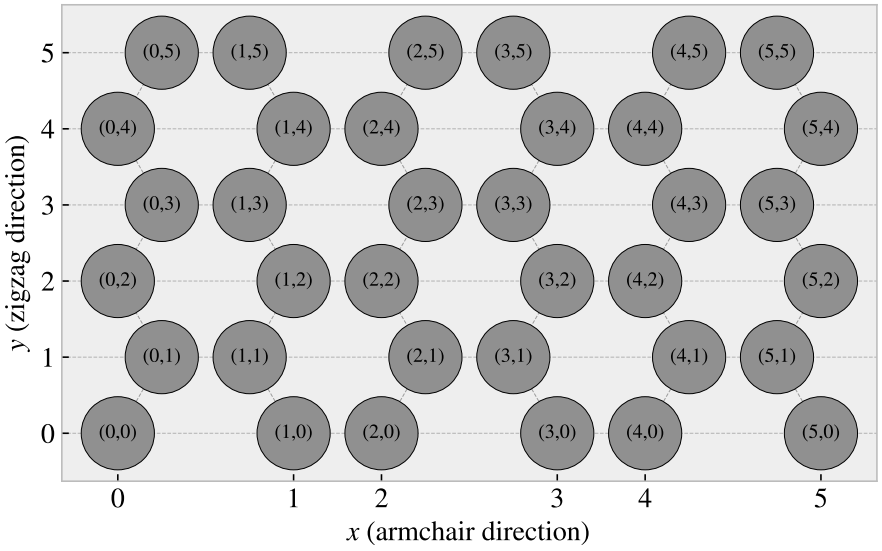


Figure 2.4: Graphene atom indexing

2.4.2 Removing atoms

As a means to ease the formulation of the cut patterns we introduce the *center element* placed in each gap of the hexagonal honeycomb structure as shown in figure Fig. 2.5. These are not populated by any atoms but will serve as a temporary reference for the algorithmic approaches of defining a cut pattern.

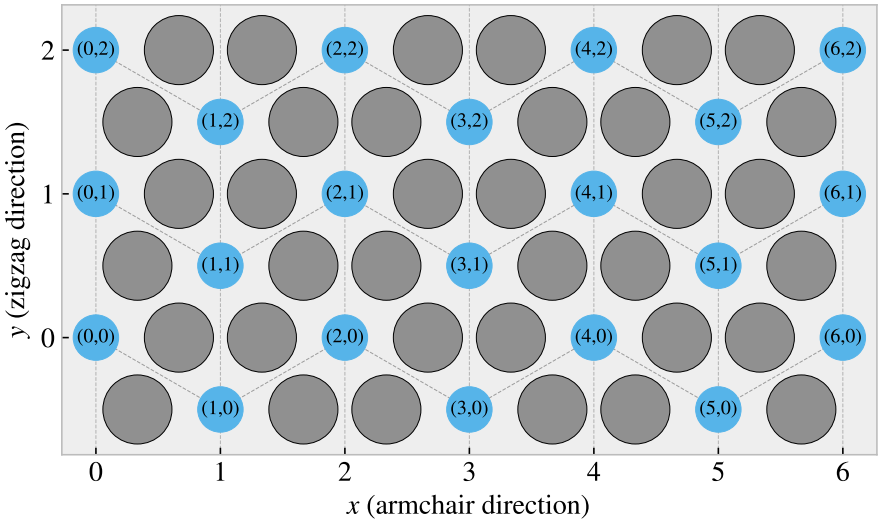


Figure 2.5: Graphene center indexing

Similar to the case of the atom indexing, the nearest neighbours center elements alternate with position, this time only along the x-coordinate index. Each center element has six nearest neighbours, in clockwise direction we can denote them: “up”, “upper right”, “lower right”, “down”, “lower left”, “upper left”. The “up” and “down” is always accesed as $(i, j \pm 1)$, but 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 neighbours and the full neighbour list is illustrated in Fig. 2.6.

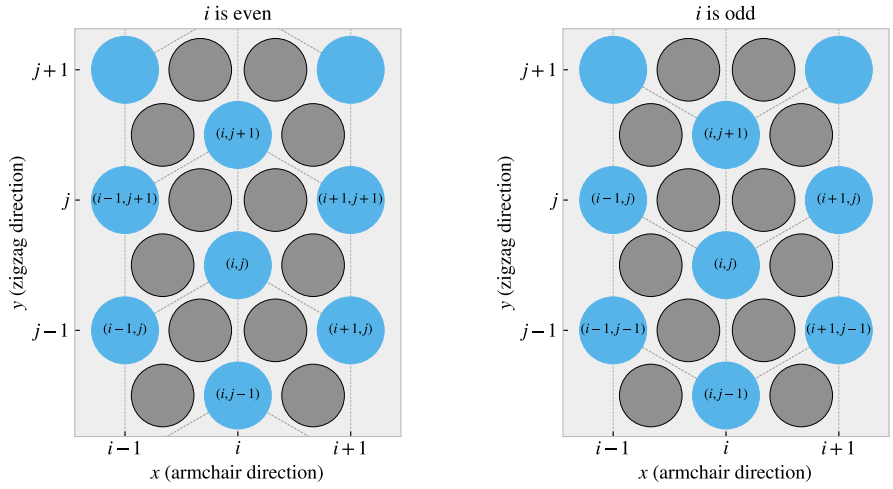


Figure 2.6: Graphene center elements directions

We define a cut pattern by connecting center elements into connected paths. As we walk from center element to center element we remove atoms according to one of two rules

1. Remove intersection 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 indepdent of the ordering of center elements in the path.

We notice that removing atoms using either of these rules will not garuantee 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:

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

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

Illustrate the example path, because I think it is otherwise impossible to follow.

For the second rule it is even more abovious 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 delete when jumping to that center element. This highlights the importance of defining the atom based indexing system will yield an injective mapping for the binary cut matrix. However, using the center elements for reference makes the following definitions of the cut patterns a lot easier to design as we always can always go in the same six directions as opposed to the atom indexing system which have alternating directions for its neighbours.

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 showcases 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. 2.7a popularly used in studies of morphable metematerials [17]. This patteren produce surface buckling with a tetrahedron (three sided pyramid) shape when stretched. 2) A more intricate pattern shown in Fig. 2.7b which is used commercially by Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [18] as protective wrap for items during shipping. This pattern buckles into a hexagoal 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 cut patterns.

(a) **(b)**

Figure 2.7: Macroscale kirigami cut patterns used as inspiraton for the nanoscale implementation. (a) Tetrahedron: Alternating perpendicular cuts producing a tetrahedron shaped surface buckling when stretched [17]. (b) Honeycomb: Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [18] producing a honeycomb shaped surface buckling when stretched.

2.5.1 Tetrahedron

The *Tetrahedron* pattern is defined in terms of center elements for which all atoms sourrounding a given center element are removed. The pattern is characterized by two straight cuts, denoted line 1 and line 2, arranged

perpendicular to each other. This is done in such a way such that one line aligns with the center of the other line and with a given spacing in between. This is illustarted in Fig. 2.8. In order to achieve perpendicular cuts we cannot rely purely on the six principal directions corresponding to the center element neighbours which is spaced by 60°. We let line 1 run along the center elemnts 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 criterias of having one line point to the center of the other line we can only have 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. 2.8 shows a visual representation of the pattern components for the $(7, 5, 2)$ patteren.

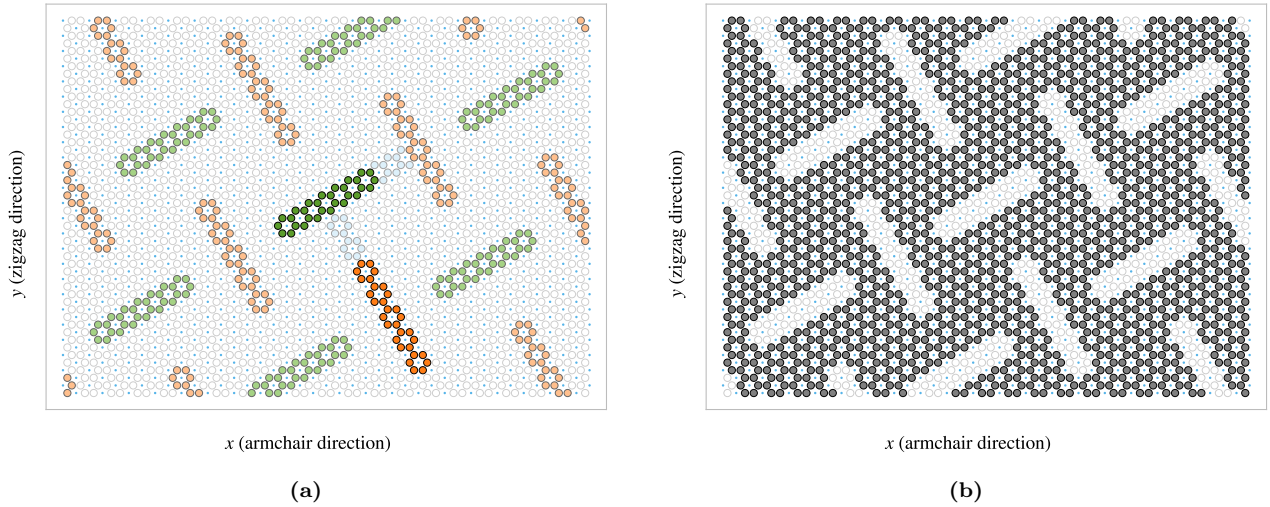


Figure 2.8: Visual representation of the tetrahedron pattern consisting of two perpendicular lines, line 1 and line 2, of length L_1 and L_2 respectively with spacing d. This example used $(L_1, L_2, d) = (7, 5, 2)$. (a) Highlight of the atoms removed. Line 1 is shown in green and line 2 in orange, with lighter colors for the translated variations, and the spacing is shown in light blue. (b) The sheet after applying the cut pattern where the grey circles denote atoms and the transparent white denotes removed atoms. The small blue circles show the center elements for reference. Sheet size used in example

In addition to the three parameters L_1, L_2, d, the pattern is also anchored to a reference point which describes the position of line 1 and 2 before translating to the whole 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. 2.8, there are 140 [1] unique reference points. Some additional variation of the pattern is showcased in Fig. 2.9 all with a reference position in the center of the sheet. Note that a smaller sheet size is used in both Fig. 2.8 and Fig. 2.9 for illustrative purposes.

[1] The general formula for this number is rather complicated in comparision to the importance in this context. Thus, we exclude the formula for this calculation as the derivation is rather handwavy executed, but the number stated here is numerically backed for this specific parameter set.

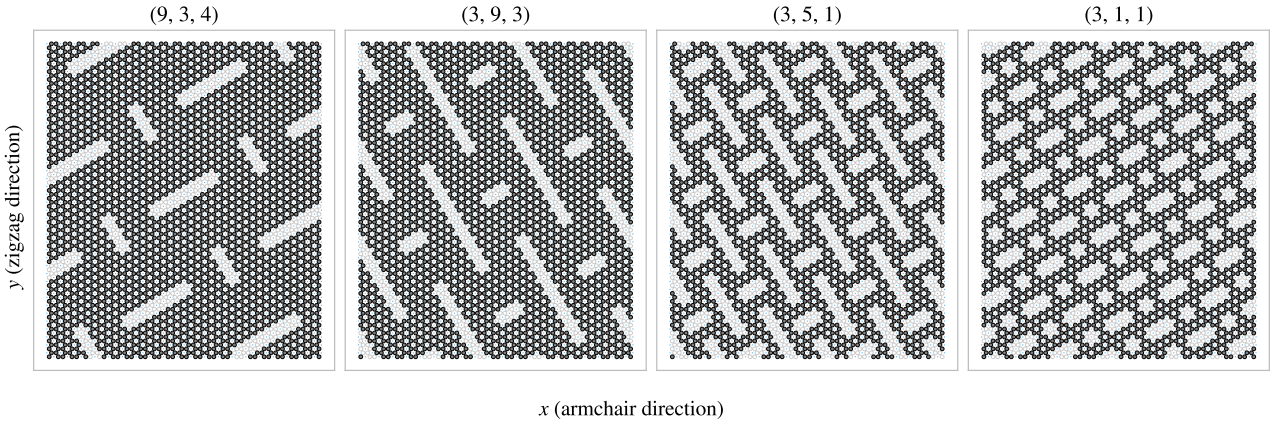
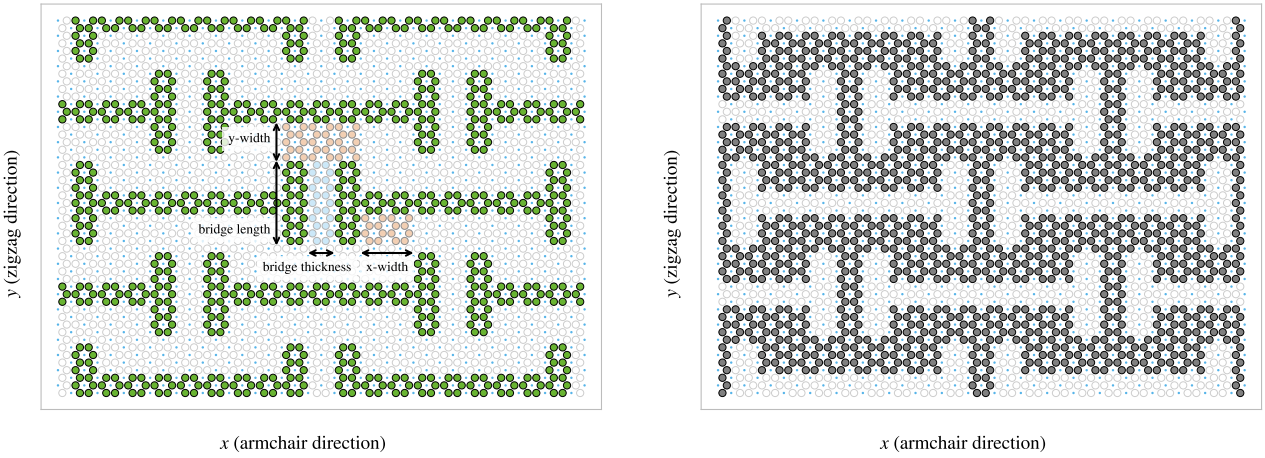


Figure 2.9: Sheet size used in example

2.5.2 Honeycomb

The *Honeycomb* pattern is defined, similarly to the Tetrahedron pattern, in terms of the center elements for which all sourrounding atoms are removed. The Honeycomb pattern is build from a repeating series of cuts remniscient 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-offsett we get the class of honeycomb patterns as visualized in Fig. 2.10. The pattern is described in terms of the parameters: (x-width, y-width, bridge thickness, bridge length) which is annotated in Fig. 2.10a with the parameters (2, 2, 1, 5) used as an example. Some additional variations of the pattern class is showcased in Fig. 2.11.



(a) **(b)**

Figure 2.10

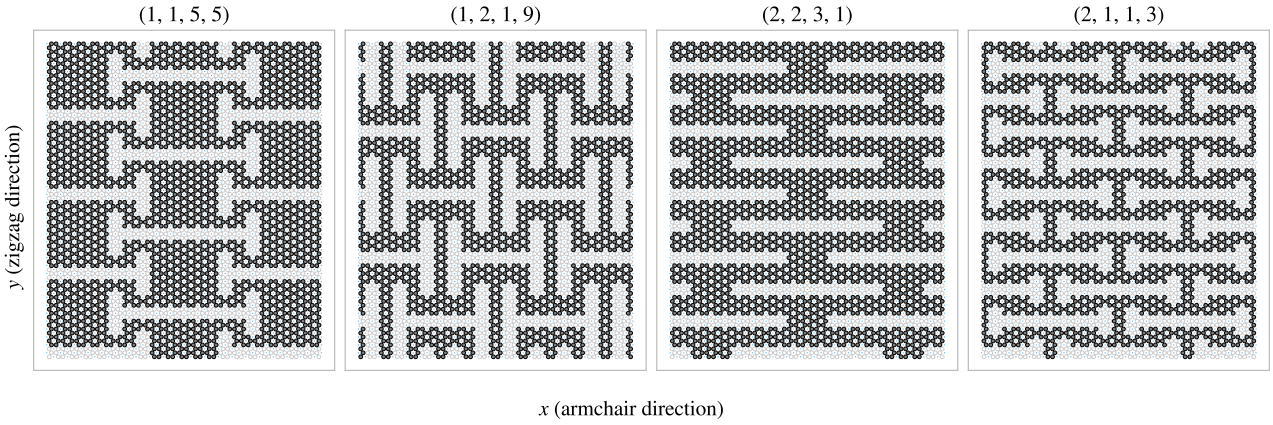


Figure 2.11

2.5.3 Random walk

The random walk serves as a method for introducing random patterns into the dataset with the scope of populating the configuration space more broadly than achieved purely with the more systematic patterns described above. 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 lots of non-connected atom clusters. Intuitively, we do not find this promising for the generation of large scale organized 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.

Table 2.3: Parameters for the random walk generator.

Parameters	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 chocie of the next site.
Connection	Atoms / Center elements	Whether to walk between atom sites or center elements (removing all adjecent atoms).
Avoid unvalid	True/False	Whether to remove already visited sites from the neighbour 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 amount of tries, 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).

2.5.3.1 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 center elements, removing all sourrounding atoms — *Connection: Atom/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 corresponds with the property of a self avoiding random walk, but it futhermore constraint the walkers not to visit any path previously visited by others walkers which we might denote "other avoiding" then. By default, the walker has an equal chance of chosing any of its adjecent neighbours for the next step, i.e. we draw the next step from a discrete uniform distribution. Optionally we can use periodic bounary conditions, *Periodic: True/False*, allowing neighbouring 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 neighbour options for the next step while traveling on the center elements this gives six neighbour options. If the walker happens to arrive at an already visited site the walk is terminated early. Optionally, we can choose to remove any neighbouring sites already visited from the neighbour list, *Avoid unvalid: True/False*, and choose uniformly between the remaining options instead. 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 it cannot backtrack its own path either, and in such a case the walk will be terminated despite the setting of *Avoid unvalid*.

2.5.3.2 Spacing of walking paths

In order to control the spacing between the paths of the various walkers we implement a so-called, *minimum distance: 0, 1, ...*, parameter describing the spacing required between paths in terms of the least amount of steps. When a walker has ended its walk, either by early termination or hitting the maximum limits of steps, all sites within a walking distance of the minimum distance 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 behaviour introduced in the previous paragraph. In practice this is done through a recursive algorithm as described in algorithm 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. For each site, the function then gathers all site neighbours (regardless of their state on the sheet) and call itself using this neighbour list as input while incrementing a distance counter passed along. This will result in an expansion along all possible outgoing paths from the initial path of interest which is terminated when the distance counter hits the distance limit. The function will then return the final neighbour lists which are accumulated into a final output corresponding to a list of all sites within the minimum distance to the path.

Algorithm 1 Recursive algorithm implemented as class method to mark sites within a distance of the class attribute self.min_dis.

```
Require: self.min_dis > 0                          ▷ This pseudocode does not handle other cases
function WALK_DISTANCE(self, input, dis = 0, pre = [ ])
    new_neigh ← [ ]                                ▷ Initialize list for new neighbours
    for site in input do
        neigh ← get_neighbouring_sites(site)       ▷ Get sourrounding neighbours
        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)
            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 neighbouring sites
        pre ← input
        return pre + self.walk_distance(new_neigh, dis, pre)
    end if
end function
```

Do we need a figure supporting this?

2.5.3.3 Bias

We include the option to perform biased random walk through the Bias: (direction, strength) parameter option. We implement this by modelling each walking step analog to the canonical ensemble under the influence of an external force **F** representing the bias. For such a system each microstate i, corresponding to the sites in the neighbour list, has the associated probability p_i given by the Gibbs–Boltzmann distribution

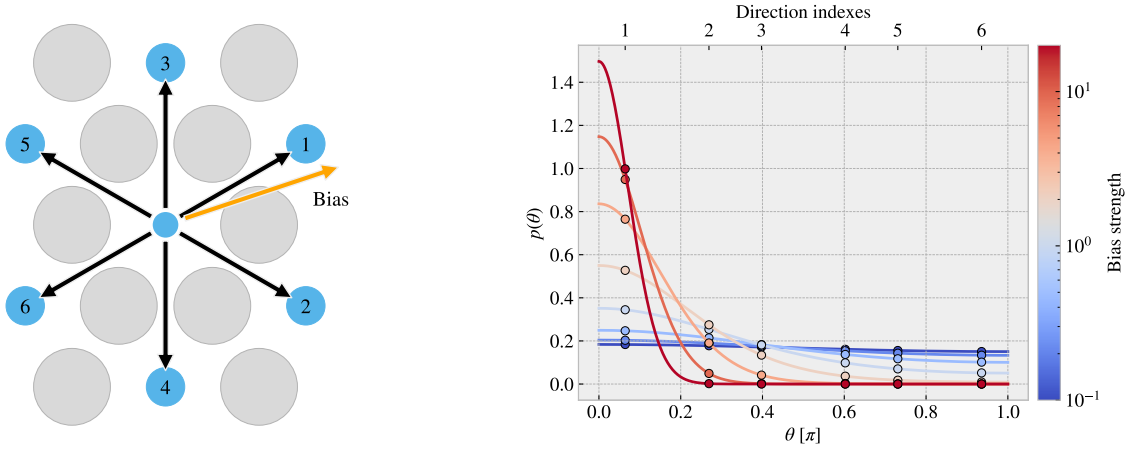
$$p_i = \frac{1}{Z} e^{-\beta E_i}, \qquad Z = \sum_i e^{-\beta E_i},$$

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

for aling the bias, analogous to an energy gain by moving there. Due to the symmetry of both the atom sites and the center elements sites the step length to neighbouring sites will always be equal. By defining the bias magnitude $B = \beta|\mathbf{F}||\mathbf{s}|$ we the probability for jumping to site i as

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

where the hat denotes the unit direction of the vector. The bias magnitude B then captures the opposing effects of the magnitude of the external force and the temperature of the system as $B \propto |\mathbf{F}|/T$. 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 allinged 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 simply as a normalization constant. Thus, numerically we can enforce this simply by setting $Z = 1$ at first, calculate p_i, and then normalize the result at the final stage as a divition by the sum of all p_i. In the numerical implementation we then pick the step destination weighted by the discrete probability distribtion p_i. In Fig. 2.12 we have illustrated how a bias of different strength impacts the probability disitrbution for a random walk between center elements. We can visually confirm that the bias will favorise the directions that lies closes to the bias direction. This favorization is more distinct at high bias strengths while at low strength $B \to 0$ we get a uniform distribution which alligns with the default unbiased random walk.



(a) **(b)**

Figure 2.12: Illustration of the probability distribution for the various step direction during a bias random walked between center elements. (*a*) Shows the possible step directions as black arrows pointing towards the neighbouring center elements shown as blue circles. The bias direction is denoted as an orange arrow and the numbering indicates the most likely direction to take (1) towards the least likely (6). The atoms sites are marked as grey circles for reference. (*b*) The probability distribution as a function of angle between the direction of choice and the bias direction. The disitrbution is normalized according to the discrete probabilities marked with dots for which the continous line simple highlights the curve of the distribution. The direction indexes corresponds to the numbering on figure (*a*). The color map indicates different strenghs of the bias.

2.5.3.4 Stay or break

The *Stay or break: True/False* parameter defines the probability p_{stay} that the walker will keep its direction or otherwise break into a different direction by probability $1 - p_{\text{stay}}$. That is, we manually substitute p_{stay} in the discrete probability for the next step corresponding to a continuation in the same direction. We then shift the remaining probabilities such that the distirbution remains normalized. In this way we can still perform bias random walk in combination. For the center element walk it is trivial to determine which of the neighbour 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 neighbour indexes alternates for each increment in x or y index (see Eq. (2.1)) which corresponds to the alternating neighbour directions D

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

One way to mittigate 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. 2.13, for each center element direction (black arrows) there are two possible atom 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 determine which of the atom directions that are 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 garuanteed zigzag motion along the center element direction that it happens to start on.

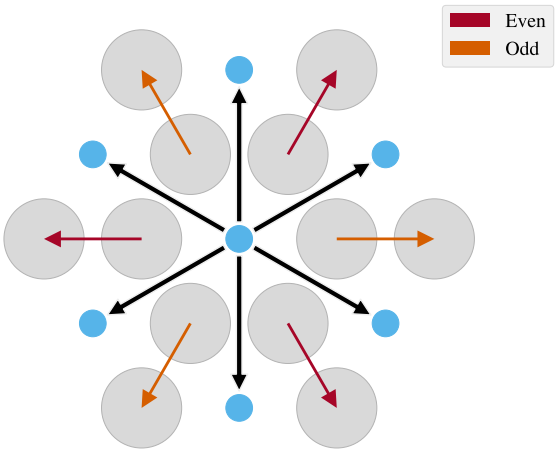


Figure 2.13: Caption

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 unvalid: 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 go in.

2.5.3.5 Deployment schemes

By default, each random walker is given a uniform random starting point among the non-visited availble sites left on the sheet. This includes any modifications in relation to the minimum distance parameter. By toggling the *Grid start: True/False* parameter on, 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 accomodate a 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 central as possible. The lower left partition square is then choosen 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 used[2]. The population of the grid is visualized in Fig. 2.14 for 1-9 walkers in total with color coding for the order of deployment.

[2]In hindsight, it would have been less biased to choose a random partition square, but we do not consider this to be of great importance for the usage of this feature in final dataset

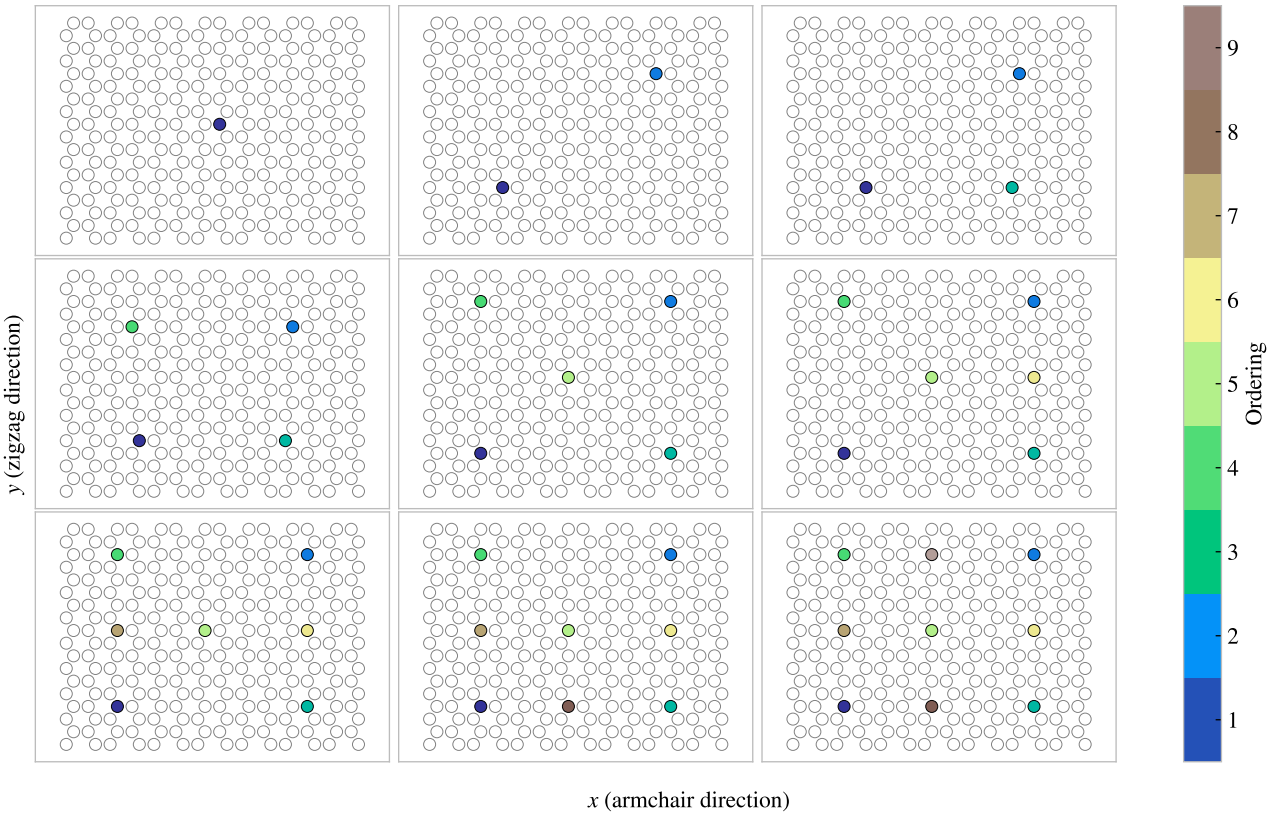


Figure 2.14: Populaiton of starting points with the centering parameter toggled on in a 14×18 sheet. This is shown for 1-9 number of random walkers with the color map conveying the order of the population.

The *Centering: True/False* parameter let us relocate the path of the random walker such that the path center of mass alligns better with the starting point. When toggled on, the path is moved in the direction defined by the center of mass and the starting point for which the closest valid relocation on the direct translation line is choosen. This can be used in combination with the grid start and the bias parameter to make rather ordered configurations. In addition, the *RN6:True/False* parameter can be used update the bias direction to on of the six directions of the center element walk for each new walker deplyoed. This lets us create confgiratiosn like the one shown in Fig. 2.15b.

2.5.3.6 Validity

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

1. There exist only a single cluster on the sheet. We define a cluster as the set of atoms which can all be reached through nearest neighbour walking on the cluster.
2. The cluster of atoms is spanning the sheet in the y-direction. This means that there exist at least one path through nearest neighbour walks that connect the bottom and the top of the sheet. This is due to the reason that the sheet must be attatched to the pull blocks.

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. This is done, *Avoid clustering: 0, 1, ...*, amount of times. 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 which alters the sheet by the intention of performing the least amount of changes (addition or subtraction of atoms) in order to meet the attatchment 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.

2.5.3.7 Random walk examples

Some examples of the random walk patterns are illustrated in Fig. 2.15.

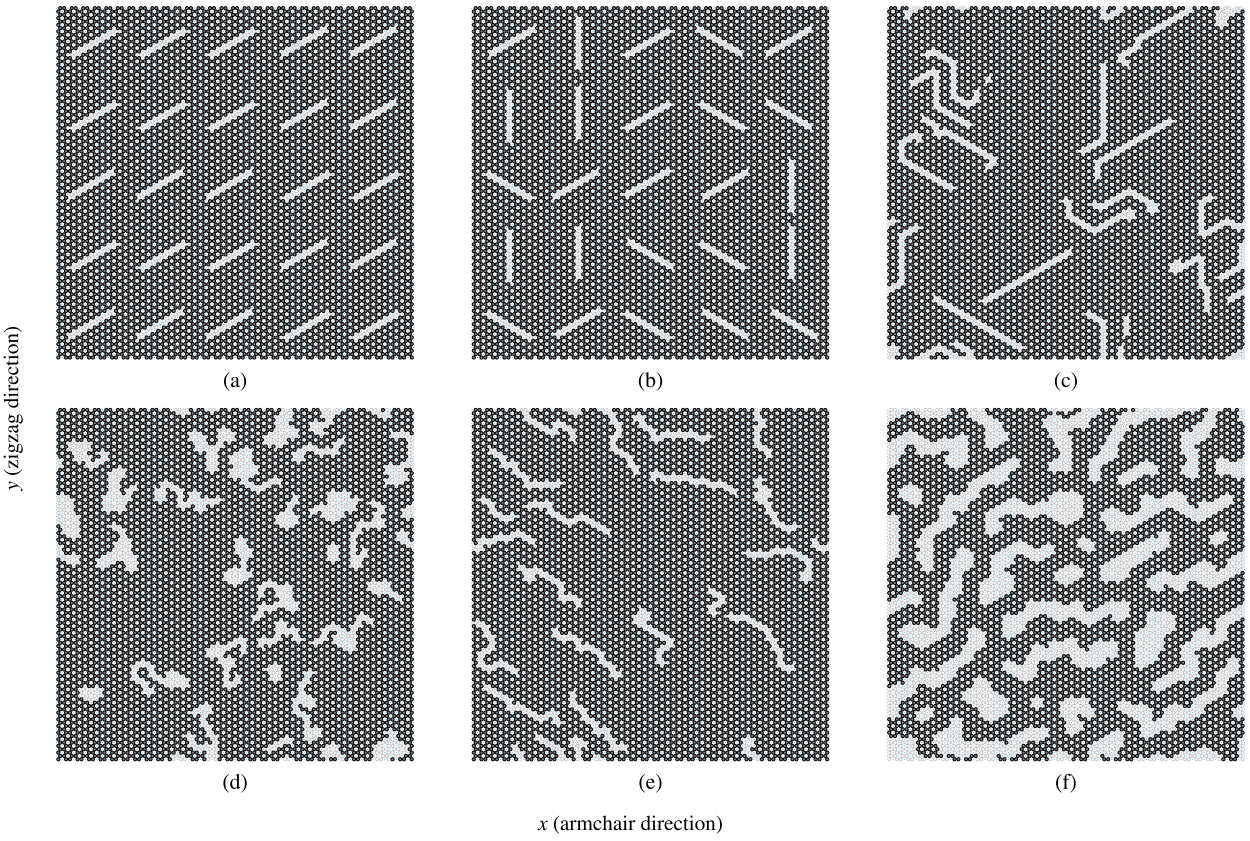


Figure 2.15: Some example uses of the random walking class. Give information of parameters? Henrik says Yes :D

Appendices

Appendix A

Appendix B

Appendix C

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, 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]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).

[14]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).

[15]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).

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

[17]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`.

[18]*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`.