

# 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





© 2023 Mikkel Metzsch Jensen

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

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

Printed: Reprocentralen, 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

**GAN** Generative Adversarial Networks. 2

**MD** Molecular Dynamics. 1, 2, 3, 4, 9, 27, 28, 29

**ML** Machine Learning. 2, 29

**NN** Nearest neighbours. 14

**SFA** Surface force apparatus. 9



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Goals . . . . .	3
1.3	Contributions . . . . .	3
1.4	Thesis structure . . . . .	4
<b>I</b>	<b>Background Theory</b>	<b>5</b>
<b>II</b>	<b>Simulations</b>	<b>7</b>
<b>2</b>	<b>Creating a graphene kirigami system</b>	<b>9</b>
2.1	Region definitions . . . . .	9
2.2	Numerical procedure . . . . .	12
2.3	Setting up the substrate . . . . .	13
2.4	Setting up sheet . . . . .	13
2.4.1	Indexing . . . . .	14
2.4.2	Removing atoms . . . . .	15
2.5	Kirigami patterns . . . . .	16
2.5.1	Tetrahedron . . . . .	16
2.5.2	Honeycomb . . . . .	18
2.5.3	Random walk . . . . .	19
2.5.3.1	Fundamentals . . . . .	20
2.5.3.2	Spacing of walking paths . . . . .	21
2.5.3.3	Bias . . . . .	21
2.5.3.4	Stay or break . . . . .	22
2.5.3.5	Deployment schemes . . . . .	23
2.5.3.6	Validity . . . . .	24
2.5.3.7	Random walk examples . . . . .	25
<b>3</b>	<b>Summary</b>	<b>27</b>
3.1	Summary and conclusions . . . . .	27
3.1.1	Designing an MD simulation . . . . .	27
3.1.2	Genereting Kirigami patterns . . . . .	28
3.1.3	Control friction using Kirigami . . . . .	28
3.1.4	Capture trends with machine learning . . . . .	28
3.1.5	Accelerated search . . . . .	29
3.1.6	Negative friction coefficient . . . . .	30
3.2	Outlook . . . . .	30
<b>Appendices</b>		<b>31</b>
<b>A</b>	<b>Appendix A</b>	<b>33</b>

<b>A Appendix B</b>	<b>35</b>
<b>B Appendix C</b>	<b>37</b>

# Chapter 1

## Introduction

### 1.1 Motivation

Friction is the force that prevents the relative motion of objects in contact. From our everyday life, 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 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 “toggable” 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 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 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<sup>1</sup> 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. 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 *reverse design*, where the model predicts a design from a set of 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. attributes 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 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 rare property only found a few cases. These are mainly for the unloading phase of adhesive surfaces [15] or 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.

---

<sup>1</sup>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 solutions, which is often beyond the capabilities of computational resources.

## 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) 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 framework 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 load and stretch for different Kirigami designs.
4. Develop and train a machine learning model to predict the MD simulation result and perform an accelerated search of new designs with the scope 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, but the straining of certain Kirigami sheets allows for both a relative increase and decrease in friction [Henrik](#). 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 search.

During our investigations, I have built three numerical tools, beyond the regular scripts associated with data analysis, which can be found 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 concerning the choice of sheet configuration, system size, simulation parameters and MD potentials, which makes it applicable for further studies within this topic. I have also built an automated 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. 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 allow 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 used for this thesis have been originally developed for the purpose with the exception of the usage of libraries as mentioned above and commonly known 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 will formulate our research questions in the light of the friction theory.

In Part II, we begin by presenting our definition and setting up of the system in Chapter 2. This includes the MD simulations and the generation of Kirigami designs. This is followed by a pilot study in ?? where we evaluate the simulation results for various physical conditions and compare a non-cut sheet to two different Kirigami designs. Further explorations of the Kirigami configurations are carried out in ?? which includes the generation of a dataset and the employment of machine learning and an accelerated search. 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 in ?. Finally, we summarize our results and provide an outlook for further studies in Chapter 3. Additional figures are shown in ??, ?? and ??.

# 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.

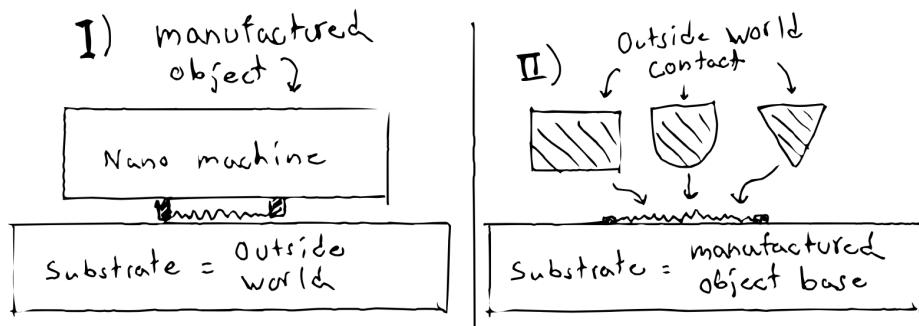


Figure 2.1: TMP System variations

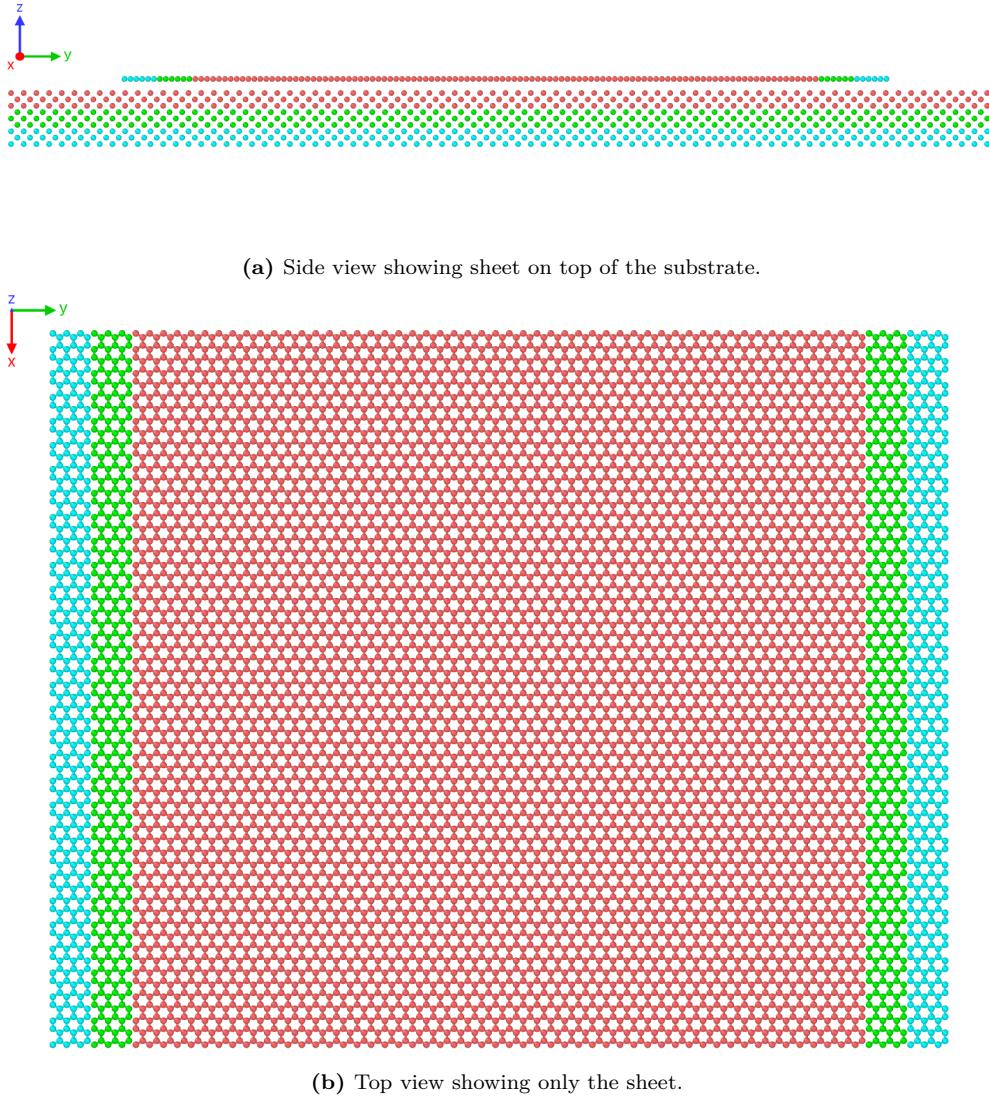
### 2.1 Region definitions

We subdivide the two main parts of the simulation, 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 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 relaxation 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 separated. 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*) 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 in to rigid objects. For the substrate this refers to an additionally fixation as well.

The full sheet is given a size  $\sim 130 \times 163 \text{ \AA}$  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.



**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 <sup>2</sup> ]
Full sheet	$x_S \times y_S$	$130.029 \times 163.219 \text{ \AA}$	212.23
Inner sheet	$x_S \times 0.81 y_S$	$130.029 \times 132.853 \text{ \AA}$	172.74
Pull blocks	$2 \times x_S \times 0.09 y_S$	$2 \times 130.029 \times 15.183 \text{ \AA}$	$2 \times 19.74$
Substrate	$1.16 x_S \times 3.12 y_S$	$150.709 \times 509.152 \text{ \AA}$	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  $10^5 \text{ eV}/\text{\AA}^2 \sim 1.6 \times 10^6 \text{ N/m}$ : 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 separating 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 equilibrated 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,  $F = -\gamma v$ , is added to the sheet to resist the rapid acceleration of the sheet and prevent a hard impact between the sheet and substrate. The damping coefficient is set to  $\gamma = 8 \times 10^{-4} \text{ nN}/(\text{m/s})$  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,  $F_{slide} \propto Kv_{slide}t$ , 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  $1.42 \text{ \AA}$ , of all graphene atoms are recorded. If any of these nearest neighbors exceeds a threshold distance of  $4 \text{ \AA}$ , 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  $a_{\text{Si}} = 5.43 \text{ \AA}$ . The default substrate thickness is chosen such that 9 layers of atoms appear (2 unit cells) corresponding to a thickness of  $10.86 \text{ \AA}$ . 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  $R + D = 2.1 \text{ \AA}$ . 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  $20 \text{ \AA}$  which provides  $2 \cdot 20 \text{ \AA} - 2.1 \text{ \AA} = 37.9 \text{ \AA}$  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  $15 \text{ \AA}$  for which the preferred sliding speed of  $20 \text{ m/s} = 2 \text{ \AA/ps}$  gives  $15 \text{ ps}$  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

$$\mathbf{T}_{mn} = m\mathbf{a}_1 + n\mathbf{a}_2, \quad m, n \in \mathbb{N}.$$

For graphene, we have the primitive lattice vectors

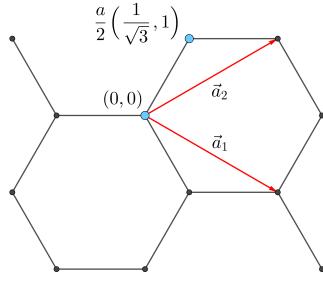
$$\mathbf{a}_1 = a \left( \frac{\sqrt{3}}{2}, -\frac{1}{2} \right), \quad \mathbf{a}_2 = a \left( \frac{\sqrt{3}}{2}, \frac{1}{2} \right), \quad |\mathbf{a}_1| = |\mathbf{a}_2| = 2.46 \text{ \AA}.$$

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

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

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

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



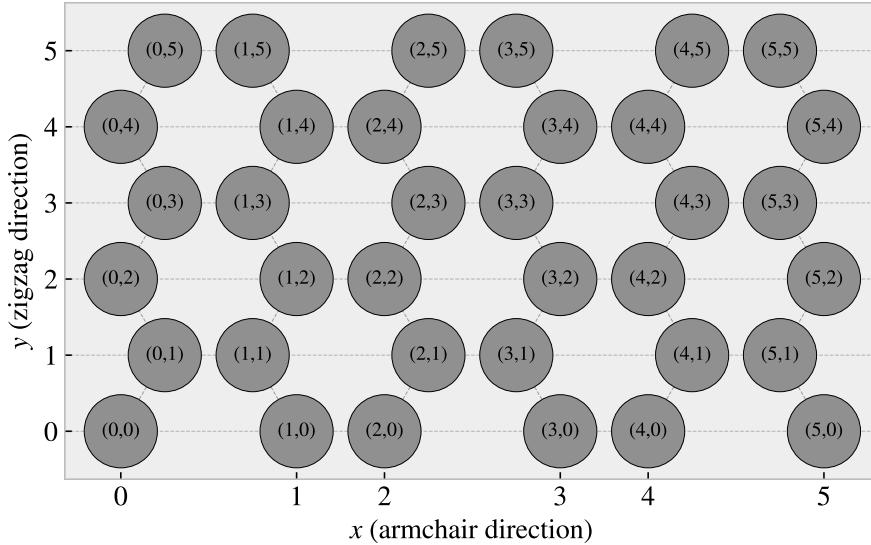
**Figure 2.3:** Graphene crystal structure with basis.

### 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 align 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 will 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

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

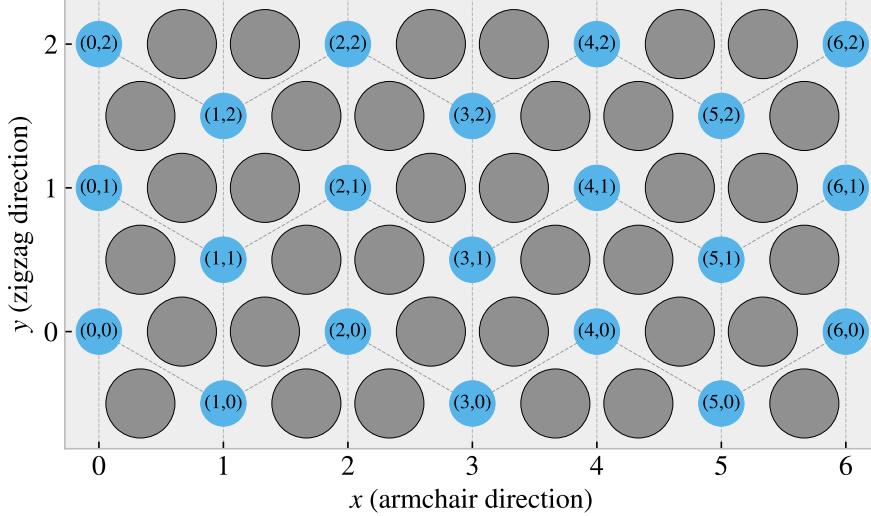
By consulting Fig. 2.4 we can verify this visually, and it basically comes down to the fact whether the atom is oriented to the right or the left side in the zigzag chain.



**Figure 2.4:** Graphene atom indexing

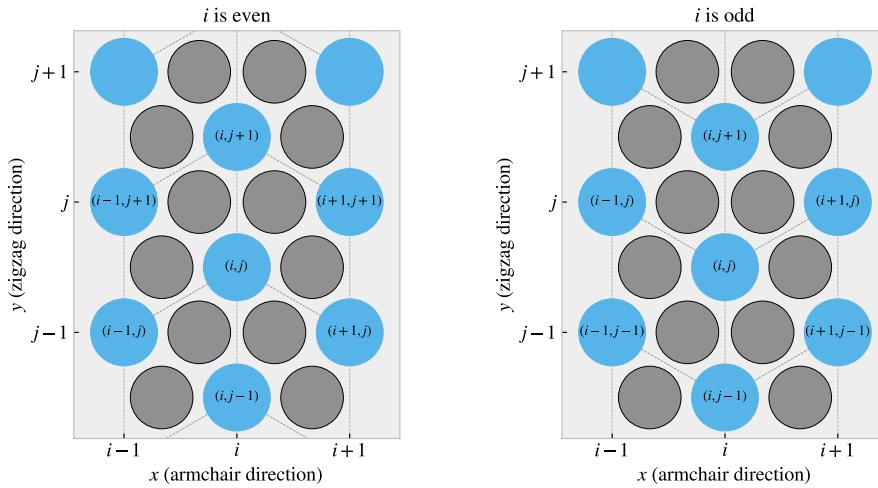
### 2.4.2 Removing atoms

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 accessed 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 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:

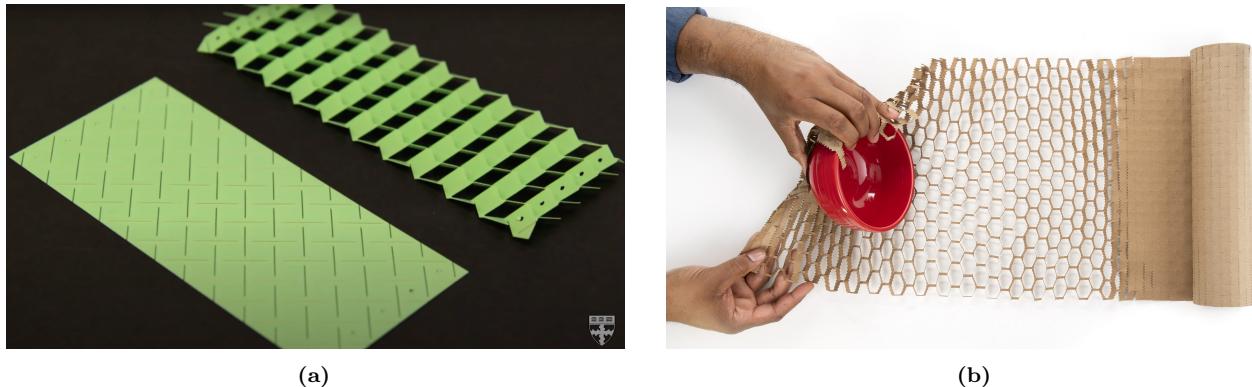
$$\begin{aligned} \text{Path 1: } (i, j) &\rightarrow \underbrace{(i+1, j+1)}_{\text{upper right}} \rightarrow \underbrace{(i, j+1)}_{\text{up}} \rightarrow \underbrace{(i+1, j+2)}_{\text{upper right + up}} \rightarrow \underbrace{(i+1, j+1)}_{\text{upper right}} \\ \text{Path 2: } (i, j) &\rightarrow \underbrace{(i+1, j+1)}_{\text{upper right}} \rightarrow \underbrace{(i+1, j+2)}_{\text{upper right + up}} \rightarrow \underbrace{(i, j+1)}_{\text{up}} \end{aligned}$$

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

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 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 metamaterials [24]. This pattern 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™ Cushion Lock™ [25] 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 cut patterns.

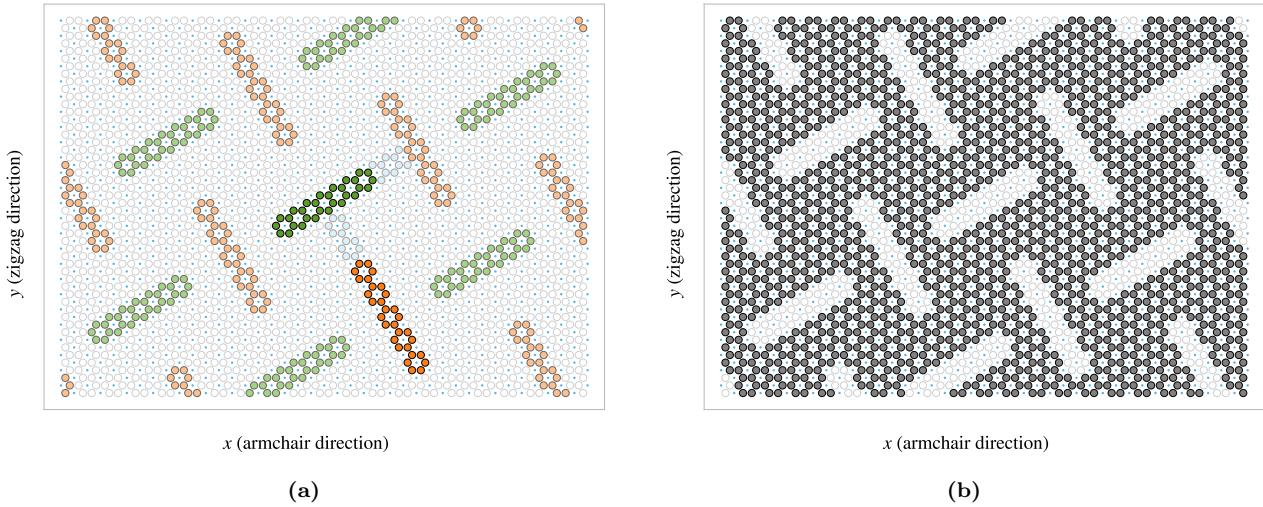


**Figure 2.7:** Macroscale kirigami cut patterns used as inspiration for the nanoscale implementation. (a) Tetrahedron: Alternating perpendicular cuts producing a tetrahedron shaped surface buckling when stretched [24]. (b) Honeycomb: Scotch™ Cushion Lock™ [25] producing a honeycomb shaped surface buckling when stretched.

### 2.5.1 Tetrahedron

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

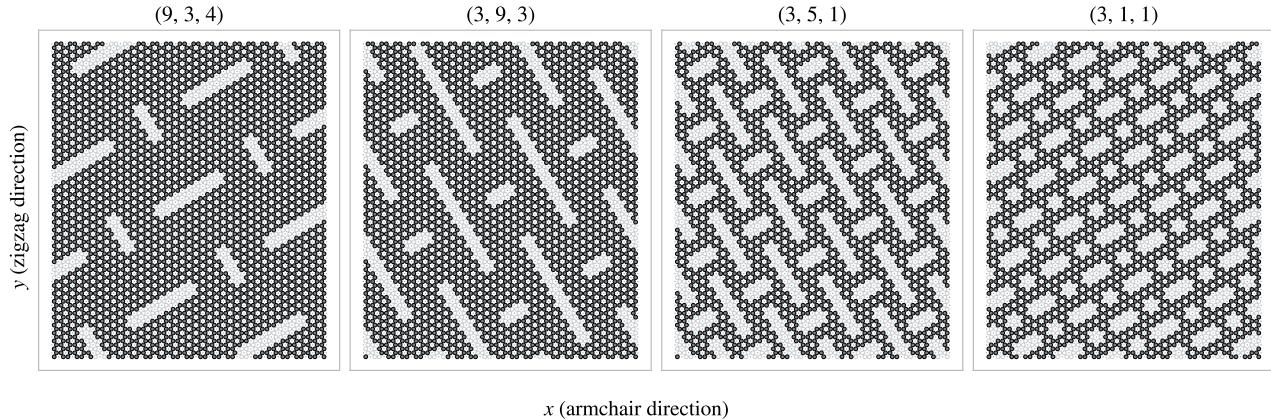
perpendicular to each other. This is done in such a way such that one line aligns with the center of the other line and with a given spacing in between. This is illustrated in Fig. 2.8. In order to achieve perpendicular cuts we cannot rely purely on the six principal directions corresponding to the center element neighbours which is spaced by  $60^\circ$ . 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 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  $|L_2 - L_1| = 2, 6, 10, \dots$ . Fig. 2.8 shows a visual representation of the pattern components for the  $(7, 5, 2)$  pattern.



**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  $140^2$  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.

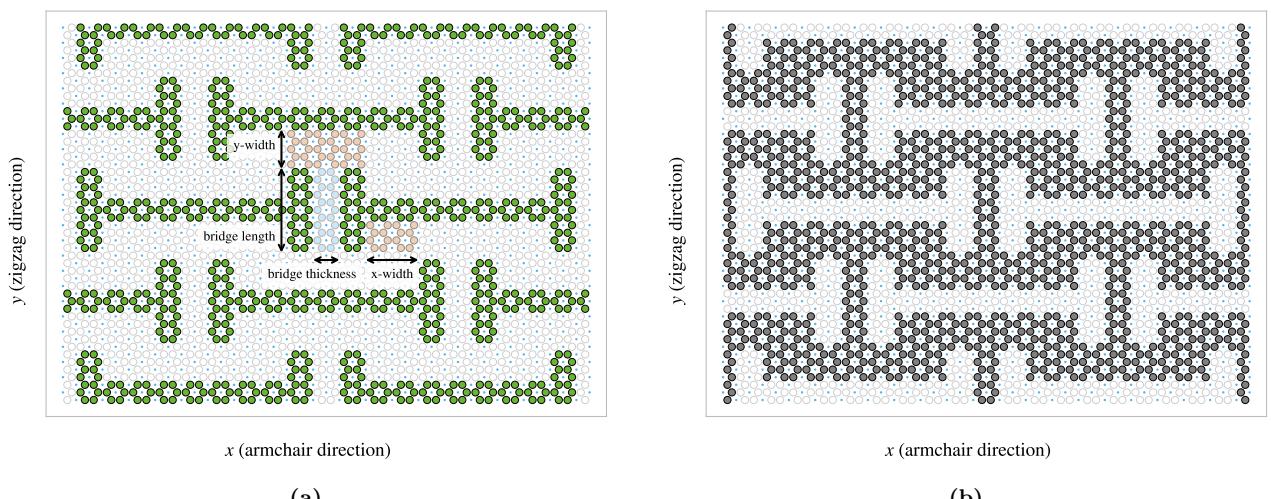
<sup>2</sup>The general formula for this number is rather complicated in comparison to the importance in this context. Thus, we exclude the formula for this calculation as the derivation is rather handwavy executed, but the number stated here is numerically backed for this specific parameter set.



**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 surrounding atoms are removed. The Honeycomb pattern is build from a repeating series of cuts reminiscent of the roman numeral one rotated by  $90^\circ$  ( $\text{I}$ ). For a given spacing these are put next to each other in the x-direction,  $\text{I} - \text{I} - \text{I}$ , 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-offsets 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.



**Figure 2.10**

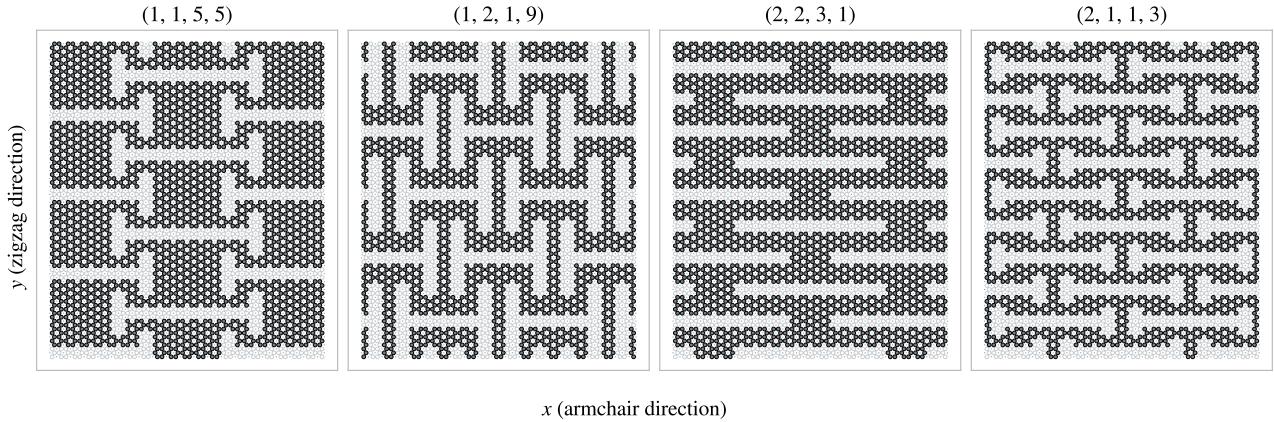


Figure 2.11

### 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 $\geq 1$	Number of random walks to be initiated on the sheet (one at a time).
Max. steps ( $S$ )	Integer $\geq 1$	The maximum steps allowed for any random walker.
Min. distance	Integer $\geq 0$	The minimum distance required between any future paths and the previous paths in terms of the shortest walking distance in between.
Bias	(Direction, strength $\geq 0$ )	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 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 $\geq 0$	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 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 corresponds with the property of a self-avoiding random walk, but it furthermore constraint the walkers not to visit any path previously visited by other walkers which we might denote “other avoiding” then. By default, the walker has an equal chance of choosing any of its adjacent neighbours 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 or false, 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 it cannot backtrack its own path either, and in such a case the walk will be terminated despite the setting of *avoid invalid*.

### 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* parameter, taking integer values  $\geq 0$ , 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 walking distance of the minimum distance is 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 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 neighbors (regardless of their state on the sheet) and calls itself using this neighbor 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 neighbor 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 a 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
1: function WALK_DISTANCE(self, input, dis = 0, pre = [ ])      ▷ Initialize list for new neighbours
2:   new_neigh ← [ ]
3:   for site in input do
4:     neigh ← get_neighbouring_sites(site)                      ▷ Get surrounding neighbours
5:     for n in neigh do
6:       if (n not in pre) and (n not in new_neigh) then          ▷ If not already added
7:         AddItem(new_neigh, n)
8:       end if
9:     end for
10:    end for
11:    dis += 1                                              ▷ Increment distance counter
12:    if dis ≥ self.min_dis then                                ▷ Max limit hit
13:      return input + new_neigh
14:    else                                                 ▷ Start a new walk from each of the neighboring sites
15:      pre ← input
16:      return pre + self.walk_distance(new_neigh, dis, pre)
17:    end if
18: end function

```

---

Do we need a figure supporting this?

### 2.5.3.3 Bias

We include the option to perform a biased random walk through the *Bias* parameter which takes a direction and a strength input. We implement this by modeling each walking step analog to 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

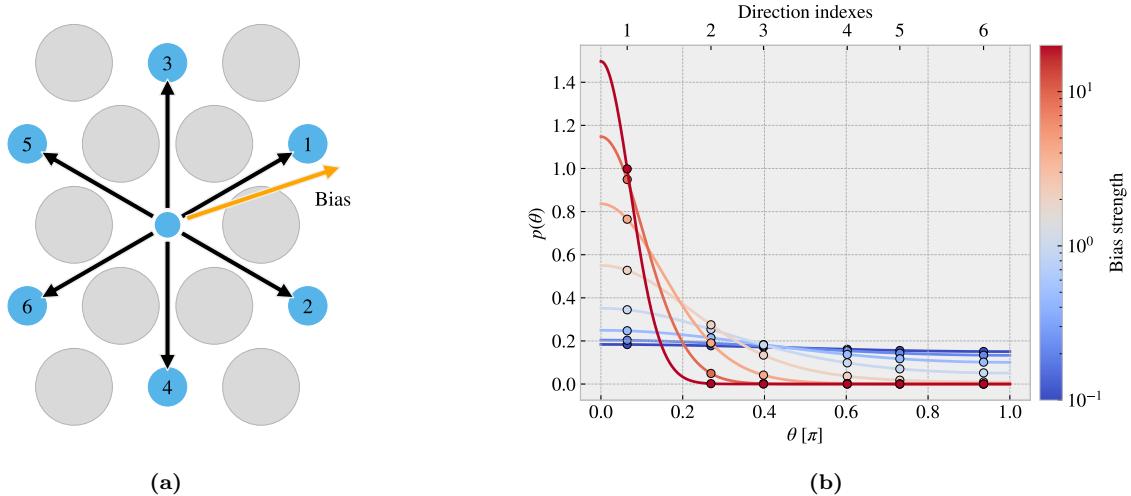
$$p_i = \frac{1}{Z} e^{-\beta E_i}, \quad Z = \sum_i e^{-\beta E_i},$$

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  $s$  the energy becomes  $E_i = -s \cdot \mathbf{F}$ , 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 both the atom sites and the center elements sites the step length to neighboring 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 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  $\theta$  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 simply 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 step destination weighted by the discrete probability distribution  $p_i$ . In Fig. 2.12 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 favorise the directions that lie closer to the bias direction. This factorization is more distinct at high bias strengths while at low strength  $B \rightarrow 0$  we get a uniform distribution that aligns with the default unbiased random walk.



**Figure 2.12:** Illustration of the probability distribution for the various step direction during a bias random walk 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 distribution is normalized according to the discrete probabilities marked with dots for which the continuous line simply highlights the curve of the distribution. The direction indexes corresponds to the numbering on figure (a). The color map indicates different strengths of the bias.

#### 2.5.3.4 Stay or break

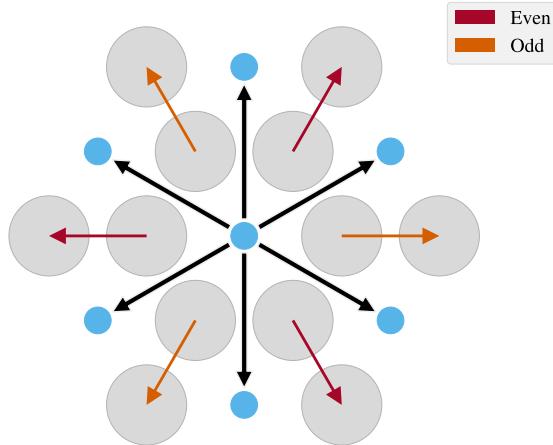
The *Stay or break* parameter defines the probability  $p_{\text{stay}}$  that the walker will keep its direction or otherwise break into a different direction by probability  $1 - p_{\text{stay}}$ . That is, we manually substitute  $p_{\text{stay}}$  in the discrete probability for the next step corresponding to a continuation in the same direction. We then shift the remaining probabilities such that the distribution 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 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. 2.13, 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 guaranteed zigzag motion along the center element direction that it happens to start on.



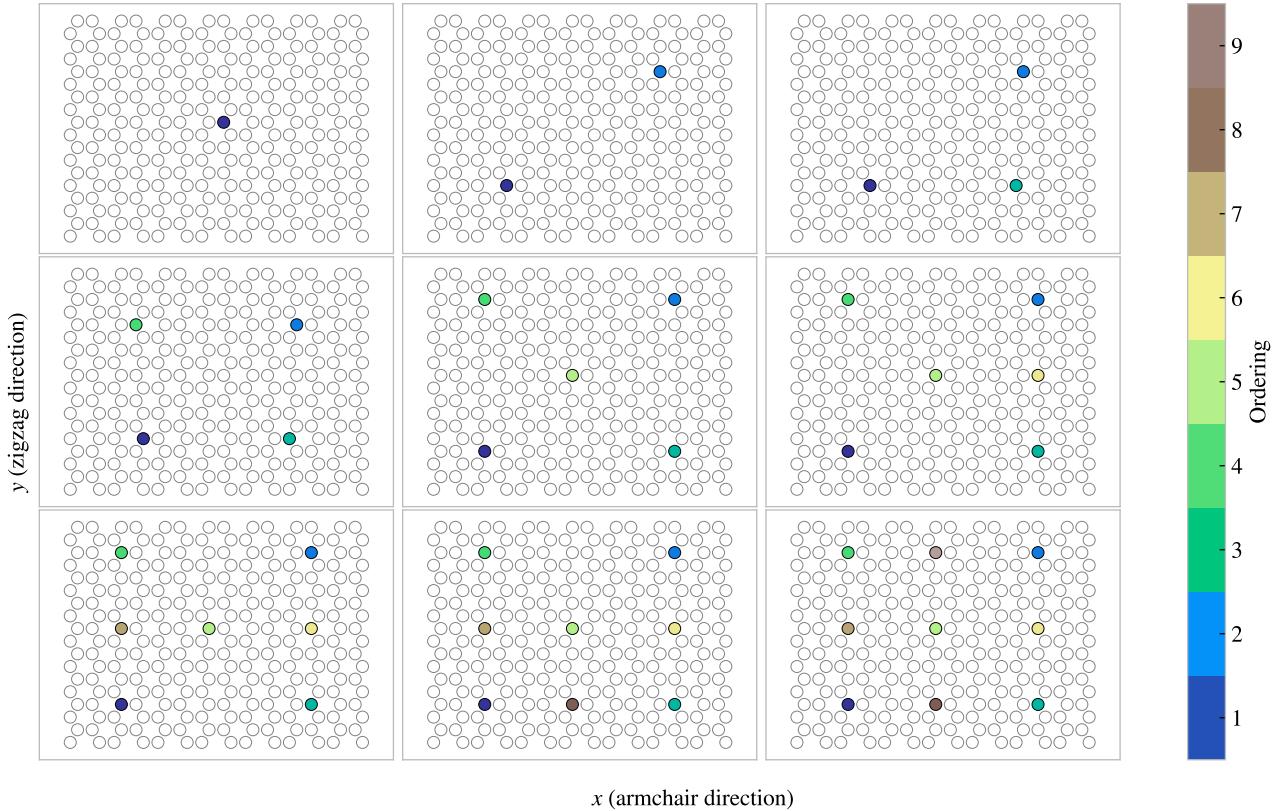
**Figure 2.13:** [Caption]

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 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 go in.

#### 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 a space for each starting point. 1 walker leads to a  $1 \times 1$  partition, {2, 3, 4} walkers lead to a  $2 \times 2$  partition, {5, 6, 7, 8, 9} walker lead to a  $3 \times 3$  partition and so on. For each partition square the starting point is placed as central as possible. The lower left partition square is then 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 used<sup>3</sup>. 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.

<sup>3</sup>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:** Population of starting points with the centering parameter toggled on in a  $14 \times 18$  sheet. This is shown for 1-9 number of random walkers with the color map conveying the order of the population.

The *centering* parameter lets us relocate the path of the random walker such that the path center of mass aligns 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 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 directions of the center element walk for each new walker deployed. This lets us create configurations like the one shown in Fig. 2.15b.

### 2.5.3.6 Validity

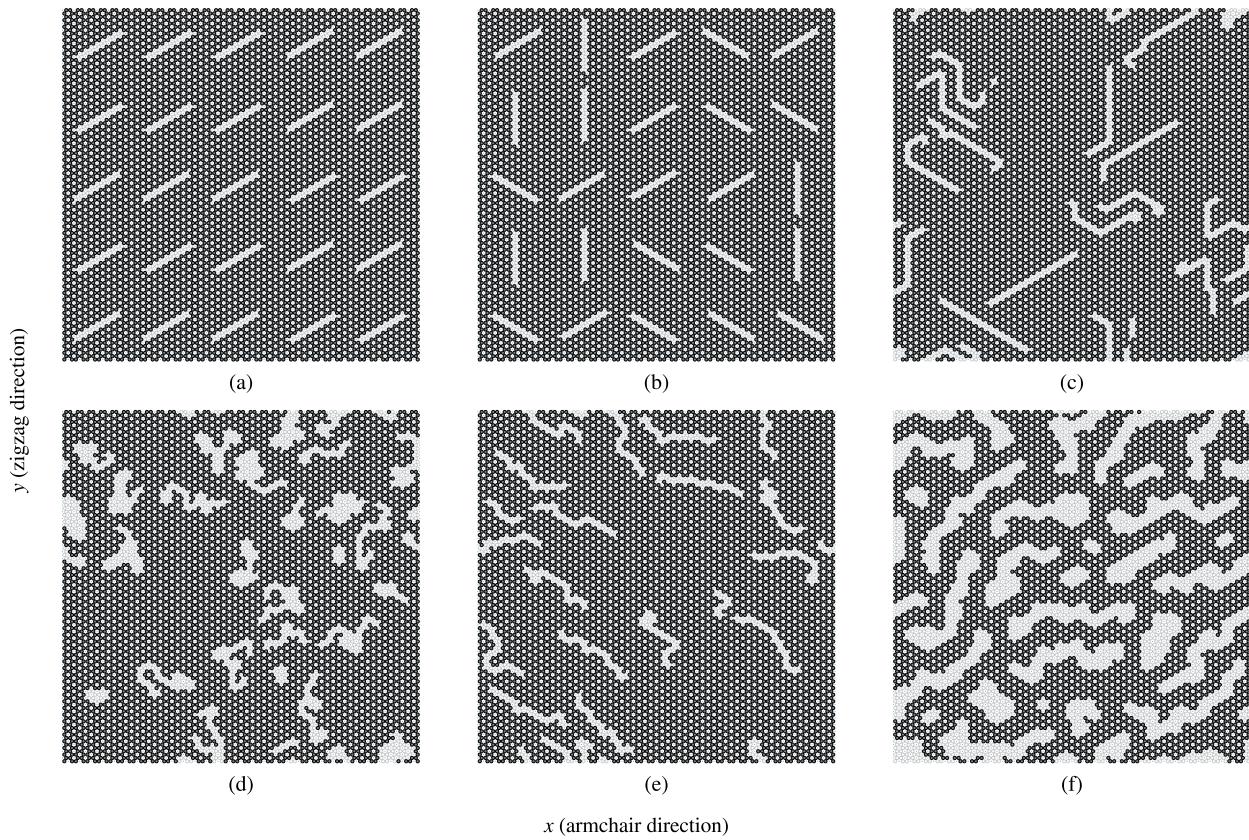
The simulation procedure requires the sheet to be fully attached, non ruptured, which can be summarized as the following requirements.

1. There 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 attached 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 according to *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 which alters the sheet by 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.

### 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



# Chapter 3

## Summary

In this thesis we have studied nanoscale friction of a Kirigami graphene sheet under the influence of load and strain using MD simulations. We have developed a numerical framework for generating various Kirigami designs which was used to create a dataset of the frictional behavior depending on the Kirigami pattern, strain and loading. Our findings suggest that the frictional behavior of a Kirigami sheet is highly dependent on the geometry of the pattern and the strain conditions. We observed that the out-of-plane buckling can be associated with a non-linear friction-strain curve which can be utilized to demonstrate a negative friction coefficient in a system with coupled load and strain. Moreover, we have investigated the possibility to use machine learning on this dataset and attempted an accelerated search. Our result suggest that machine learning can be feasible for this approach, but more data is needed to provide a more reliable foundation for a search of new Kirigami patterns. In this chapter we will summarize the findings in more detail and draw some conclusions. At the end we will provide some topics for further research.

### 3.1 Summary and conclusions

#### 3.1.1 Designing an MD simulation

We have designed an MD simulation for the examination of friction for a graphene sheet sliding on a silicon substrate. The key system features were the introduction of the pull blocks, defined as the end regions of the sheet with respect to the sliding direction, which was utilized for applying normal load and sliding the sheet. The pull blocks were made partly rigid and used to employ a thermostat as well. By an analysis of the friction forces retrieved from sliding simulations we defined a standardized metric for kinetic friction. We measured the force acting from the substrate on the full sheet (including the pull blocks) with respect to the sliding direction and defined kinetic friction by the mean value of the last half of the simulation. The uncertainties were estimated on the basis of the fluctuations in the running mean. We found that the assessment of static friction was ambiguous for our simulation and did not pursue this further. From the analysis of the force traces, friction force vs. time, we identify the friction behavior in our simulation domain as being in the smooth sliding regime mainly due to the choice of sliding speed (20 m/s) and infinitely stiff springs. This was further supported by a demonstration of a transition to stick-slip behavior with softer springs and a lowering of sliding speed. By conducting a more systematic investigation of the effects of temperature, sliding speed, spring constant and timestep, we settled on the default values based on numerical stability and computational cost. We found that friction increased with temperature which we attribute to being in the ballistic sliding regime. We used the room temperature 300 K as a standard choice. Furthermore, we found friction to increase with velocity as expected, with some signs of phonon resonance at certain sliding speeds as well. We chose a rather high velocity of 20 m/s mainly for the consideration of computational costs. For the spring constant, we found decreasing friction with increasing stiffness of the springs which is associated with the transition from a stick-slip-influenced regime toward smooth sliding. The choice of an infinitely stiff spring was made from a stability assessment. Finally, we confirmed that a timestep of 1 fs provides reasonable numerical stability. However, based on fluctuations with timestep we find that the uncertainty in the simulations might be higher than first estimated.

### 3.1.2 Generetig Kirigami patterns

In order to invstigate the effects of Kirigami design we have created a numerical framework for generating various patterns. By defining an indexing system for the hexagonal lattice structure we were able to define the Kirigami designs as a 2D binary matrix for numerical implementation. We digitalized two different macroscale designs, which we named the *Tetrahedron* and *Honeycomb* pattern, that successfully produced out-of-plane buckling when stretched. Through our numerical framework we were able to create an ensemble of perturbed unique variations which yielded approximately 135k and 2025k for the Tetrahedron and Honeycomb patterns respectively. When considering the possibility to translate the patterns we find the ability to increase the number by roughly a factor 100. In addition we created a framework for generating random walk based Kirigami patterns. This was regulated by introducing features such as bias, avoidance of existing cuts, preference to keeping a direction and procedures to repairing the sheet for simulation purposes. In general, the capabilities of the numerical framework for generating Kirigami designs exceeded our computational resources with regard to performing MD simulation under different load and strain for each of the designs. Thus our MD-based dataset only utilized a subset of configurations with 9660 data points based on 216 Kirigami configurations (Tetrahedron: 68, Honeycomb: 45, Random walk: 100, Pilot study: 3). Thus our Kirigami generative framework can be valuable for further studies on an extended dataset.

### 3.1.3 Control friction using Kirigami

We have investigated the frictional behavior of the Tetrahedron and Honeycomb patterns in comparison to a non-cut sheet under various strains and loads. Initially, we observed that straining the Kirigami sheets in vacuum resulted in an out-of-plane buckling. When adding the substrate to the simulation this translated into a decreasing contact area with strain. We found the Honeycomb sheet to exhibit the most significant buckling with a corresponding reduction of relative contact area to approximately 43%. The non-cut sheet did not produce any significant buckling in comparison. We found that friction generally increased with strain which contradicts the asperity theory hypothesis of decreasing friction with decreasing contact area. Moreover, the friction-strain curve exhibited highly non-linear trends with strong negative slopes (see ??), while the non-cut sheet did not show any significant dependency on the strain. We also found that the non-stretched Kirigami patterns did affect friction to some degree, but this was one order of magnitude lower than the effects associated with the strain in combination. This led us to the conclusion that the changing contact area cannot be regarded as a dominant mechanism for friction in the Kirigami sheet system nor the independent consideration of sheet configuration or tension in the sheet. When considering the dependency with load we generally found a weak dependency which can be associated with a friction coefficient on the order of  $10^{-4}$ – $10^{-5}$  even though we could not confirm any clear relationship. This is best attributed to a superlubric state of the graphene sheet as seen in other studies as well. The slope of the friction-load curves was not considerably affected by the straining of the Kirigami sheet which led us to the conclusion that the strain-induced effects are dominant in comparison to any load-related effects. By proposing a linear coupling between load and strain with ratio  $R$  we find that these results suggest the possibility to find negative friction coefficients in certain load ranges following  $-R12.75\text{ nN}$  for the Tetrahedon and  $-R \cdot 2.72\text{ nN}$  for the Honeycomb pattern.

### 3.1.4 Capture trends with machine learning

The dataset reveals some general correlations with mean friction, such as a positive correlation to strain (0.77) and porosity (0.60), and a negative correlation to contact area (-0.67). These results align with the finding in the pilot study, suggesting that the change in friction is associated with cuts in the sheet (porosity) and a changing contact area.

By defining the friction property metrics:  $\min F_{\text{fric}}$ ,  $\max F_{\text{fric}}$ ,  $\max \Delta F_{\text{fric}}$  and  $\max \text{drop}$  (maximum decrease in friction with strain), we investigated the top candidates within our dataset. From these results, we found no incentive for the possibility to reduce friction with the Kirigami approach since the non-cut sheet provided the lowest overall friction. Regarding the maximum properties, we found an improvement from the original pilot study values and with the Honeycomb pattern producing the highest scores. This suggests that the data contains some relevant information for optimization with respect to these properties. Among the top candidates, we found that a flat friction-strain profile is mainly associated with little decrease in the contact area and vice versa which again aligns with the pilot study findings.

For the machine learning investigation, we have implemented a VGGNet-16-inspired convolutional neural network with a deep “stairlike” architecture: C32-C64-C128-C256-C512-C1024-D1024-D512-D256-D128-D64-D32, for convolutional layers  $C$  with the number denoting channels and fully connected (dense) layers  $D$  with the number denoting nodes. The final model contains  $1.3 \times 10^7$  parameters and was trained using the ADAM optimizer for a cyclic learning rate and momentum scheme. We trained the network for a 1000 epochs while saving the best model during training based on the validation score. The model validation performance gives a mean friction  $R^2$  score of  $\sim 98\%$  and a rupture accuracy of  $\sim 96\%$ . However, we got lower scores for a selected subset of the Tetrahedon ( $R^2 \sim 88.7\%$ ) and Honeycomb ( $R^2 \sim 96.6\%$ ) pattern based on the top 10 max drop scores respectively. These scores were lower despite the fact that the selected set was partly included in the training data as well in addition to the fact that the hyperparameter selection favored the performance on this selected set as well. Thus we conclude that these selected configurations, associated with a highly non-linear friction-strain curve, represent a bigger challenge for the machine learning prediction. One interpretation is that these involve the most complex dynamics and perhaps that this is not readily distinguished from the behavior of the other configurations which constitute the majority of the data set. By evaluating the ability of the model to rank the dataset according to the property scores we found in general a good representation of the top 3 scores for the maximum categories, while the minimum friction property ranking was lacking. We attribute this latter observation to a higher need for precision which the model did not possess.

In order to provide a more true evaluation of the model performance we created a test set based on MD simulations for an extended Random walk search. This test revealed a significantly worse performance than seen for the validation set with a two-order of magnitude higher loss and a negative friction mean  $R^2$  score which corresponds to the prediction being worse than simply guessing on the true data mean. However, by reconsidering the choice of architecture complexity hypertuning with respect to the validation loss, we found similar poor results on the test set. This indicates, that the test score is not simply a product of a biased hypertuning process but instead points to the fact that our original dataset did not cover a wide enough configuration distribution to accurately capture the full physical complexity of the Kirigami friction behavior. Based on the validation scores we conclude that the use of machine learning is feasible, but that we need to improve the dataset in order to reach a reliable model.

### 3.1.5 Accelerated search

Using the ML model we performed two types of accelerated search. One by evaluating the property scores of an extended dataset and another with the use of the genetic algorithm approach. For the extended dataset search, we used the developed pattern generators to generate  $135\text{ k} \times 10$  Tetrahedon,  $2025\text{ k} \times 10$  Honeycomb and  $10\text{ k}$  Random walk patterns. For the minimum friction property, the search suggests a favoring of a low cut density (low porosity) which aligns with the overall idea that the dataset does not provide an incentive for further friction reduction with the non-cut sheet resulting in the lowest friction. The maximum properties resulted in some minor score increases but the suggested candidates were mainly overlapping with the original dataset. By investigating the model prediction sensitivity to the translation of the Tetrahedron and Honeycomb patterns we found that the model predictions varied drastically with small translations. This can be attributed to a physical dependency since the edge of the sheet is affected by this translation. However, due to the poor model performance on the test set, we find it more likely to be a model insufficiency related to the lacking dataset.

For the genetic algorithm approach, we investigated the optimization for the max drop property using starting population based on the result from the extended dataset accelerated search, and some random noise initializations with different porosity values. This approach did not provide any noteworthy incentive for new design structures worth more investigation. In general, the initialization of the population itself proved to be a more promising strategy than the genetic algroithm. However, this is highly affected by the uncertainty of the model predictions, and thus we did not pursue this any further. By considering the Grad-CAM explanation method we found that the model predictions occasionally payed considerable attention to the top and bottom edge of the configurations. This is surprising since these are not true edges but are connected to the pull blocks in the simulation. Despite the uncertainties in the predictions, we argue that this might be attributed to thermostat effects from the pull blocks and thus we note this as a feature worth investigating in the simulations.

### 3.1.6 Negative friction coefficient

By enforcing a coupling between load and sheet tension, mimicking a nanomachine attached to the sheet, we investigated the load curves arising from the loading of the Tetrahedron (7, 5, 1) and Honeycomb (2, 2, 1, 5) pattern from the pilot study. The non-linear trend observed for increasing strain carried over to the coupled system as well producing a highly non-linear friction-load curve. The Honeycomb pattern exhibited additionally a non-linear strain-tension curve which resulted in an almost discontinuous increase in friction for the initial increase in load. We attribute this feature to an unfolding process visually confirmed from the simulation frames. For the coupled system with a load-to-tension ratio of 6 we found regions in the friction-load curve with significant negative slopes. By considering the maximum and minimum points for such regions we estimated the average friction coefficient to be  $-0.31$  in the range  $F_N = [4.65, 6.55]$  nN for the Tetrahedron pattern and  $-0.38$  in the range  $F_N = [0.71, 4.31]$  nN for the Honeycomb pattern.

Our findings suggest that the combined use of Kirigami cuts and strain has considerable potential for friction control, especially under the influence of a coupled system of load and strain.

## 3.2 Outlook

In this thesis, we have successfully shown that certain Kirigami designs exhibit non-linear behavior with strain. This discovery was done through a focus on the exploration of the effects of different designs. This invites further investigation of the underlying mechanisms behind this phenomenon. By considering one or two design, based on Tetrahedron and Honeycomb patterns, it would be valuable to investigate the effects on the friction-strain curve under various physical parameters.

First of all, we suggest an investigation of how the friction-strain curve depends on temperature, sliding speed, spring constant, and loads for an increased range  $F_N > 100nN$ . This is especially interesting in the context of physical conditions leading to a stick-slip behavior since our study takes a basis in smooth sliding friction. Moreover, it would be valuable to verify that the choices for relaxation time, pauses, interatomic potentials and substrate material are not critical for the qualitative observations. Especially the Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) potential for the modeling of the graphene sheet might be of interest. In this context, it might also be useful to investigate the effects of excluding adhesion in the simulations. In order to gain further insight into the role of commensurability one could vary the scan angle as well. Since we suspect that our simulation corresponds to a incommensurable superlubric state, certain scan angles are hypothesized to yield higher friction. If the friction-strain curve is based on a commensurability effect this might yield qualitative different results and perhaps also allowing for a lowering of friction with strain. We might also consider investigating the friction-strain relationship under a uniform load to get insight into how the loading distribution effects the out-of-plane buckling and associated frictional effects.

Another topic worth studying is the relation to scale and edge effects. This includes an investigation of scaling effects, considering the ratio of the sheet to the edge, but also a translation of the sheet patterns to study the presence of any Kirigami-induced edge effects. The latter is motivated by the findings from the machine learning predictions. With this regard, we would also suggest a more detailed study of the effect of the thermostat in the pull blocks which is suggested to have a possibly importance when judging from the Grad-CAM analysis.

For the machine learning approach, our findings suggest that the dataset is extended considerably. This can be done with respect to the aim of exploring Kirigami configurations as done in this thesis, but it can also be done for a single configuration under variation of some of the simulation parameters to support some of the above mentioned investigations. In that context, one could consider more advanced model architectures and machine learning techniques. If successful this would invite further studies of inverse design methods such as GAN or diffusion models.

# Appendices



## Appendix A

## Appendix A



## **Appendix A**

## **Appendix B**



## **Appendix B**

## **Appendix C**



# Bibliography

- <sup>1</sup>E. Gnecco and E. Meyer, *Elements of friction theory and nanotribology* (Cambridge University Press, 2015).
- <sup>2</sup>Bhusnur, “Introduction”, in *Introduction to tribology* (John Wiley & Sons, Ltd, 2013) Chap. 1, 1–?
- <sup>3</sup>H.-J. Kim and D.-E. Kim, “Nano-scale friction: a review”, *International Journal of Precision Engineering and Manufacturing* **10**, 141–151 (2009).
- <sup>4</sup>K. Holmberg and A. Erdemir, “Influence of tribology on global energy consumption, costs and emissions”, *Friction* **5**, 263–284 (2017).
- <sup>5</sup>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.
- <sup>6</sup>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).
- <sup>7</sup>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).
- <sup>8</sup>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).
- <sup>9</sup>Y. Mao, Q. He, and X. Zhao, “Designing complex architectured materials with generative adversarial networks”, *Science Advances* **6**, eaaz4169 (2020).
- <sup>10</sup>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).
- <sup>11</sup>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).
- <sup>12</sup>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).
- <sup>13</sup>OpenAI, *Dall-e2*, 2023.
- <sup>14</sup>Midjourney, *Midjourney*, 2023.
- <sup>15</sup>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).
- <sup>16</sup>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).
- <sup>17</sup>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).
- <sup>18</sup>R. W. Liefferink, B. Weber, C. Coulais, and D. Bonn, “Geometric control of sliding friction”, *Extreme Mechanics Letters* **49**, 101475 (2021).
- <sup>19</sup>M. Metzsch, *Tuning Frictional Properties of Kirigami Altered Graphene Sheets using Molecular Dynamics and Machine Learning*.

- <sup>20</sup>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).
- <sup>21</sup>E. M. Nordhagen, *LAMMPS simulator*.
- <sup>22</sup>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. Kolsbjergr, 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).
- <sup>23</sup>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.
- <sup>24</sup>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>.
- <sup>25</sup>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](https://www.scotchbrand.com/3M/en_US/scotch-brand/products/catalog/~/Scotch-Cushion-Lock-Protective-Wrap/?N=4335+3288092498+3294529207&rt=rud).