Tuning Frictional Properties of Kirigami Altered Graphene Sheets using Molecular Dynamics and Machine Learning

Designing 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

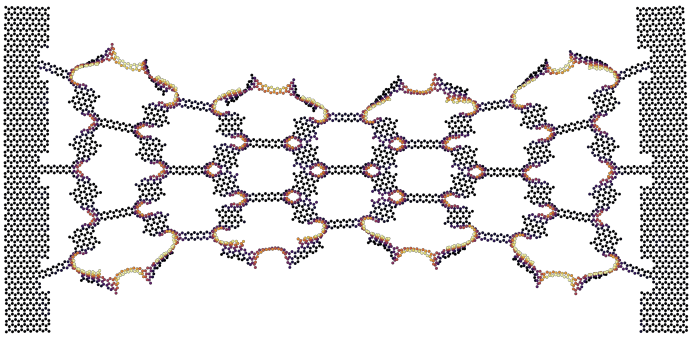
UNIVERSITY OF OSLO

Spring 2023

Tuning Frictional Properties of Kirigami Altered Graphene Sheets using Molecular Dynamics and Machine Learning

Designing a Negative Friction Coefficient

Mikkel Metzsch Jensen



© 2023 Mikkel Metzsch Jensen

Tuning Frictional Properties of Kirigami Altered Graphene Sheets using Molecular Dynamics and Machine Learning

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. 11

CNN Convolutional Neural Network. 31, 32

FC Fully Connected. 32

FFM Friction Force Microscopes. 9

MD Molecular Dynamics. 2, 3, 9

ML Machine Learning. 2, 3, 33, 40, 41, 42, 43

MSE Mean Squared Error. 33

NN Nearest neighbours. 13

SFA Surface force apparatus. 9

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Goals . . . 3
- 1.3 Contributions . . . 3
- 1.4 Thesis structure . . . 3

I Background Theory **5**

II Simulations **7**

2 Defining the system **9**
- 2.1 Region definitions . . . 9
- 2.2 Numerical procedure . . . 11
- 2.3 Creating the substrate . . . 12
- 2.4 Creating the sheet . . . 12
  - 2.4.1 Indexing . . . 13
  - 2.4.2 Removing atoms . . . 14
- 2.5 Kirigami patterns . . . 15
  - 2.5.1 Tetrahedron . . . 16
  - 2.5.2 Honeycomb . . . 17
  - 2.5.3 Random walk . . . 18
    - 2.5.3.1 Fundamentals . . . 19
    - 2.5.3.2 Spacing of walking paths . . . 20
    - 2.5.3.3 Bias . . . 20
    - 2.5.3.4 Stay or break . . . 21
    - 2.5.3.5 Deployment schemes . . . 22
    - 2.5.3.6 Validity . . . 23
    - 2.5.3.7 Random walk examples . . . 24

3 Main study **25**
- 3.1 Generating data . . . 25
- 3.2 Data analysis . . . 26
- 3.3 Properties of interest . . . 27
- 3.4 Machine learning . . . 31
  - 3.4.1 Architecture . . . 31
  - 3.4.2 Data handling . . . 32
    - 3.4.2.1 Input . . . 32
    - 3.4.2.2 Output . . . 32
    - 3.4.2.3 Data augmentation . . . 32
  - 3.4.3 Loss . . . 32
  - 3.4.4 Hypertuning . . . 33
  - 3.4.5 Final model . . . 39

3.5 Accelerated Search 43
3.5.1 Generation search 43
3.5.2 Genetic algorithm search 44

Appendices **45**

Appendix A **47**

Appendix B **49**

Appendix C **51**

Chapter 1

Introduction

Structure of Motivation section:

1. Introduce and motivate friction broadly.
2. Motives for friction control using a grasping robot as example.
3. Analog to gecko feet where adhesive properties are turned on and off.
4. Interest in origin of friction through nanoscale studies which further motivates the use of MD.
5. Intro to metamaterials and the use of kirigami designs,
6. How to optimize kirigami designs with reference to Hanakata and motivating the use of ML.
7. Out-of-plane buckling motivates the use of kirigami for frictional properties.

Does some of the latter paragraphs belong to the approach section?

1.1 Motivation

Friction is a fundamental force that takes part in most of all interactions with physical matter. 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 in our everyday life to such an extent that most people take it for granted. However, the efforts to control friction dates 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 effeciency. 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 a finetuned object handling is required. While achieving a certain "constant" friction response is readily obtained through appropiate material choices during manufacturing, we are yet to unlock the capabilities to alter friction dynamically on the go. One example from nature inspiring us to think along theese lines are the gecko feet. More precisely, the Tokay gecko has recieved a lot of attention in scientific studies aiming to unravel the underlying

mechanism of its "togglable" adhesion properties. Although geckos are able to 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 it for the next step remains reletively 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.

In the 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 preperation 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 dictates machining charascteristics [3]. With concurrent progress in computational power and devolopment of Molecular Dynamics (MD), numerical investigations serve as an extremely useful 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][7][8][9][10][11]. This is often achieved either by intertwining different material types or removing certain regions completely. In recent papers by Hanakata et al. [6](2018) [7](2020) numerical studies have showcased that mechanical properties of a graphene sheet, in this case 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 orginal human designs have contributed to specific scientiffic 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 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 prioir knowdelegde 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 simulation time on modern day hardware. However, this computational time constraint can be relaxed by the use of machine learning (ML) which have 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 all together resulting in an *accelerated search* of designs. This can be further improved by guiding the search accordingly to the most promising candidates, as for instance done with the *genetic algorithm* which suggest new designs based on mutation and crossing of the best candidates so far. Another, even more sophisticated approach, is through generative methods such as *Generative Adversarial Networks* (GAN). 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. attributes the variety in yield properties to the non-linear effects arrising 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 the kirigami cut and stretch procedure can also be exploited for the design of frictional metamaterials. For certain designs we might hope to find a relationship between stretching of the sheet and frictional properties. If signifciant, this could give rise to a variability of the friction reponse beyond manufacturing material choice. For instance, the grasping robot might apply such a material as artifical skin for which stretching or relaxing of the surface could result in a changeable friction strength; Slippery and smooth when in contact with people and rough and firmly gripping when moving heavy objects. In addition, a possible coupling between stretch and the normal load through a nanomachine design would allow for an altered friction coefficient. This invites the idea of non-linear friction coeffcients which might in theory also take on negative values given the right response from stretching. The latter would constitute an extremely rare property. This has (only?) been reported indirectly for bulk graphite by Deng et al. [13] where the friction kept increasing during the unloading phase. Check for other cases and what I can really say here.

To the best of our knowledge, kirigami has not yet been implemented to alter the frictional properties of a nanoscale system. In a recent paper by Liefferink et al. [14](2021) it is reported that macroscale kirigami can be used to dynamically control the macroscale roughness of a surface through stretching which was used to change the frictional coeffcient by more than one order of magnitude. This support the idea that kirigami designs can in fact be used to alter friction, but we believe that taking this concept to the nanoscale regime would envolve 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 and control the frictional properties of a graphene sheet through application of kirigami inspired cuts and stretching of the sheet. With the use of MD simulations we evaluate the friction properties under different physical conditions in order to get insight into the prospects of this field. By evaluating variations of two kirigami inspired patterns and a series of random walk generated patterns we create a dataset containing information of the frictional properties associated with each design under different load and stretch conditions. We apply ML to the dataset and use an accelerated search approach to optimize for different properties of interest. The subtask of the thesis are presented more comprehensively in the following.

1. Define a sheet indexing that allows for an unqiue mapping of patterns between a hexagonal graphene lattice representation to a matrix representation suited for numerical analysis.
2. Design a MD simulation procedure to evaluate the frictional properties of a given graphene sheet under specified physical conditions such as load, stretch, temperature etc.
3. Find and implement suitable kirigami patterns which exhibit out-of-plane buckling under tensile load. This includes the creation of a framework for creating variations within each pattern class. Additionally create a procedure for generating different styles of random walk patterns.
4. Perform a pilot study of a representative subset of patterns in order to determine appropriate simulation parameters to use for the further study along with an analysis of the frictional properties shown in the subset.
5. Create a dataset consisting of the chosen kirigami variations and random walk patterns and analyse data trends.
6. Train a neural network to map from the design space to physical properties such as mean friction, maximum friction, contact area etc. and evaluate the performance.
7. Perform an accelerated search optimizing for interesting frictional properties using the ML model. This should be done both through the pattern generation procedures and by following a genetic algorithm approach.
8. Use the most promising candidtes from the accelerated search to investigate the prospects of creating a nanomachine setup which exhibits a negative friction coefficient.
9. Study certain designs of interest with the scope of revealing underling mechanism. This includes simple correlation analysis but also a visualization of feature and gradient maps of the ML network.

Is the list of subtask to specific? Some of the details here might be better suited for the thesis structure section.

1.3 Contributions

What did I actually achieve

1.4 Thesis structure

How is the thesis structured.

Part I

Background Theory

Part II

Simulations

Chapter 2

Defining the system

The definition of the system plays an essential role as it sets the scene for the whole study. With the general goal of investigating the frictional behvaiour of a graphene sheet, as we alter it through kirigami cuts and stretch, two different approaches was considered as sketched in Fig. 2.1. One approach is simply to mimic a FFM type experiment as done in most other numerical friction studies. In this case, we probe the graphene sheet, resting on a substrate, with an indenting tip connected to a moving body. Friction is then measured by making the tip scan across the graphene surface. This setup allows for a variety of tip shapes and sizes, and alternatively the tip can be substituted with a flat surface making the setup resemble a SFA experiment. It is not obvious how one would achieve the stretching of the sheet, but a simple solution is to pre-stretch the sheet to a given amount and then fixating it, by the ends, on the substrate. We are then able to investigate the frictional behaviour at different amounts of stretching between simulations. If any interesting behvaiour is found, at a certain stretch amount, a possible application would call for the attatchment of a pre-stretched sheet as a surface coating. Another option is to attatch the graphene sheet to the moving body instead and introduce some sort of nanomachine in the moving body coupling the normal load with a stretching motion. This gives more design room to utilize any stretch-related friciton effects that require a dynamical changing of the stretch amount throughout the loading. While both methods serve as novel approaches with prospects of providing valuable insight into a sparsly covered field, we choose the latter option due the increased application design freedom. Hence, our system of choice consists of two seperate parts: A 2D graphene sheet and a 3D Silicon "bulk" substrate. Note, that we do attempt to model the nanomachine explicitly in this study, but we will consider the prospects of adding this in **??**.

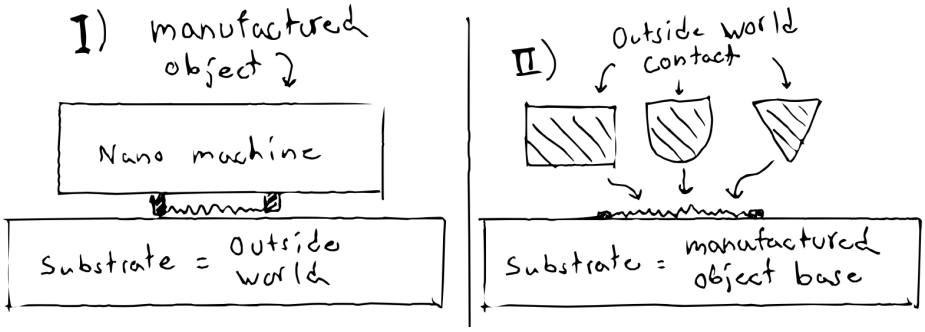


Figure 2.1: TMP System variations

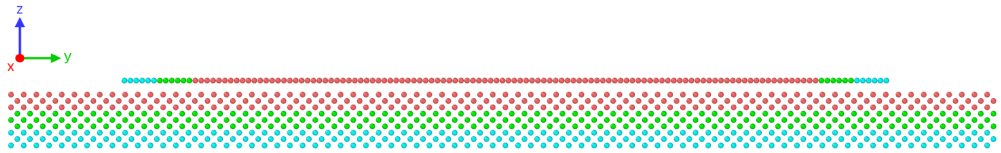
2.1 Region definitions

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

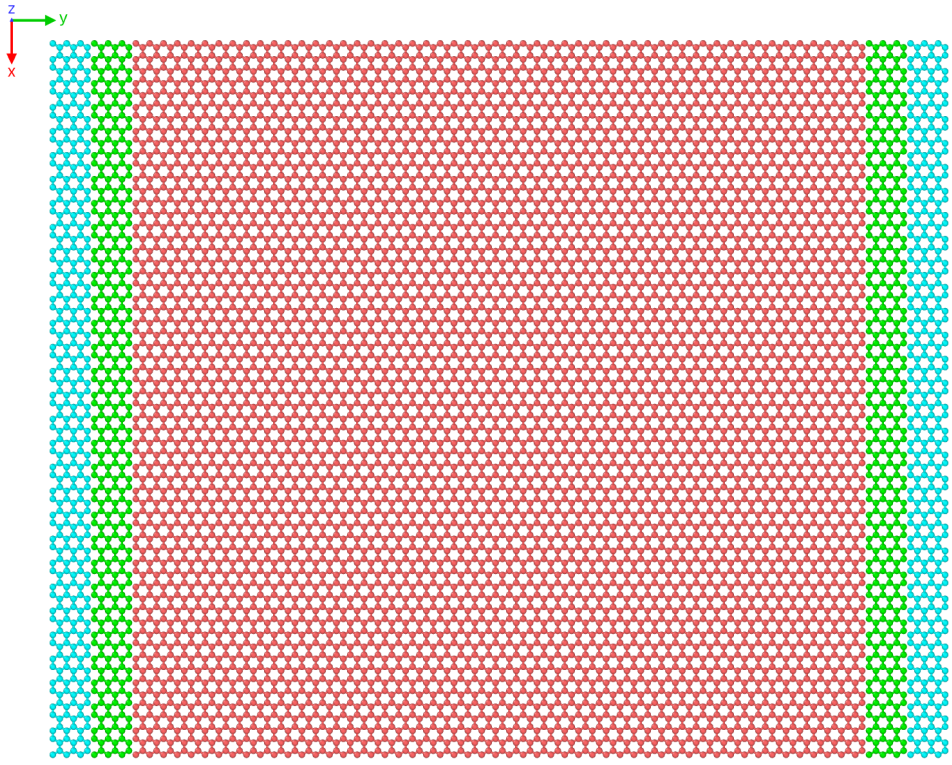
rigid part. The rigid part is however thermolized during the initial relaxtion period but made rigid for the final duration of the simulation. Note that the rigid parts on both sides of the sheet is then considered as a single rigid object even though they are physically seperated. This means that all 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 [Å^2]
Full sheet	$x_S \times y_S$	130.029×163.219 Å	$21,223.203$
Inner sheet	$x_S \times 0.81\ y_S$	130.029×132.853 Å	$17,274.743$
Pull blocks	$2 \times x_S \times 0.09\ y_S$	$2 \times 130.029 \times \ 15.183$ Å	$2 \times \ 1,974.230$
Substrate	$1.16\ x_S \times 3.12\ y_S$	150.709×509.152 Å	$76,733.789$

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

The numerical procedure related to the measurement of friction 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 crystaline form with a seperation distance of 3 Å. The equilibrium seperation distance varies slightly with temperature, but we found this number to be a reasonble middle ground for the temperature range of interest. The sheet is constrained under three hard spring forces, all with spring constant 10^5 eV/Å^2 $\sim 1.6 \times 10^6$ N/m: One spring attaches the sheet center of mass (CM) to its orginal postion, preventing any drift. The remaining two springs are attatched to the pull block ends, to their initial CM position respectively, to prevent rotation. In principle, it would be sufficient to fixate just one of the ends in order to stop rotation, but we fixate both ends for the sake of symmetry. During the relaxtion phase the pull blocks are made rigid with respect to the z-direction only (perpendicular to the sheet). That is, all the forces in the z-direction is summed up and distributed on the pull blocks as a single external force, while it is free to expand and contract in the x-y-plane. This is mainly to ensure that it achieves the correct lattice spacing according to the temperature of the system. For the following phases the rigid parts of the pull block is in fact rigid with respect to all directions. The spring forces are terminated after the relaxtion phase.

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): The 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 $\gamma = 8 \times 10^{-4}$ 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 virtuel 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 to the *sliding speed* parameter, which make the sliding force increase quadratically, $F_{slide} \propto K(v_{\text{slide}}t)^2$, with sliding speed. An infinite sprig 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.

In order to prevent rupturing, or detachment, of the sheet, we monitor the nearest neighbours for each atom throughout the simularion. At the initial timestep the three nearest neighbours, sitting at a distance 1.42 Å, of all graphene atoms is recorded. If any of these nearest neighbours exceeds a threshold distance of 4 Å, indicating a bond breakage, this is marked as a rupture and we halt the simulation early. Thus, we ensure that no wear is taking place for the sheet. The substrate was proven to be way more resistant to wear, and by running a few test simulations of high load and sliding speed we confirmed visually that no wear is accouring.

2.3 Creating the substrate

The substrate is created as a rectangular slab of Silicon (Si). We create the initial configuration according to its crystaline structure given as a diamond cubic crystal with a lattice parameter $a_{\text{Si}} = 5.43$ Å. The default substrate thickness is choosen such that 9 layers of atoms appear (2 unit cells) corresponing to a thickness of 10.86 Å. The x-y dimensions is choosen to match the dimensions. 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 to 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 relaxtion time between encounters with the sheet, similar to the inital relaxtion time.

2.4 Creating the sheet

The sheet is created by the 2D material known as graphene which consist of a single layer of carbon atom arramged in hexagonal lattice structure. The bulk version of this material is called graphite and simple referes to the merged 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 .. part of the crystal and we populate each lattice site by the given basis and translate it to fill the whole plane 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 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

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

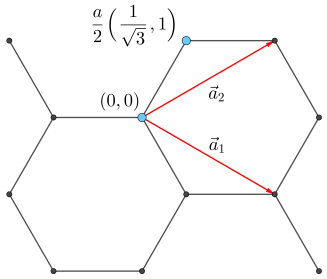


Figure 2.3: Graphene crystal structure with basis.

2.4.1 Indexing

In order to define the cut patterns applied to the graphene sheet we need to define an indexing system. We must ensure that this gives an 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 point to *zigzag* chains of atoms for which the starting point, at $y = 0$ is not evenly spacedd as illustrated in figure Fig. 2.4. Other solutions might naturally invole 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 which conserves the relative physical position of neighbours. 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 neighbour indexes will be actual nearest neighbours in the lattice. While $(i, j \pm 1)$ is always a nearest neighbour, the index of the nearest neighbour in the x-direction oscillate for each incrementing of x- or y-coordinate. That is, the nearest neighbours (NN) is 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 theleft side in the zigzag chain.

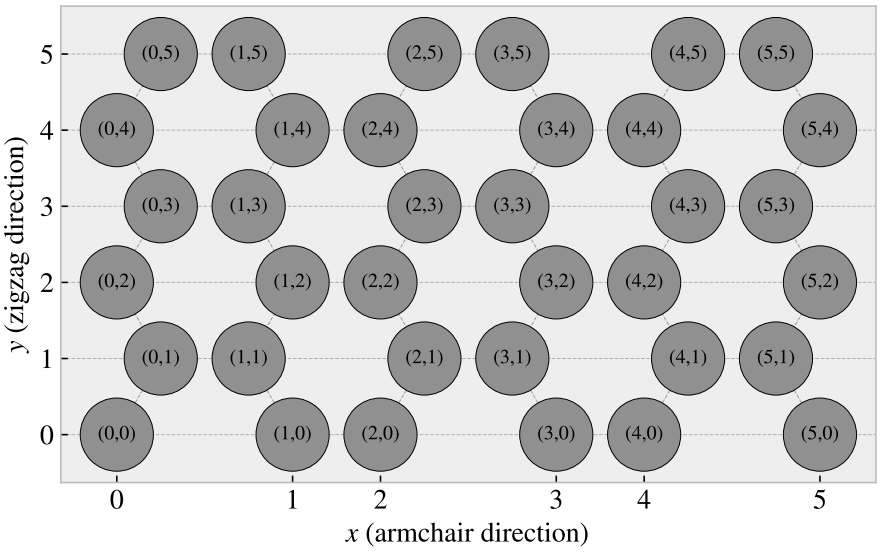


Figure 2.4: Graphene atom indexing

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.

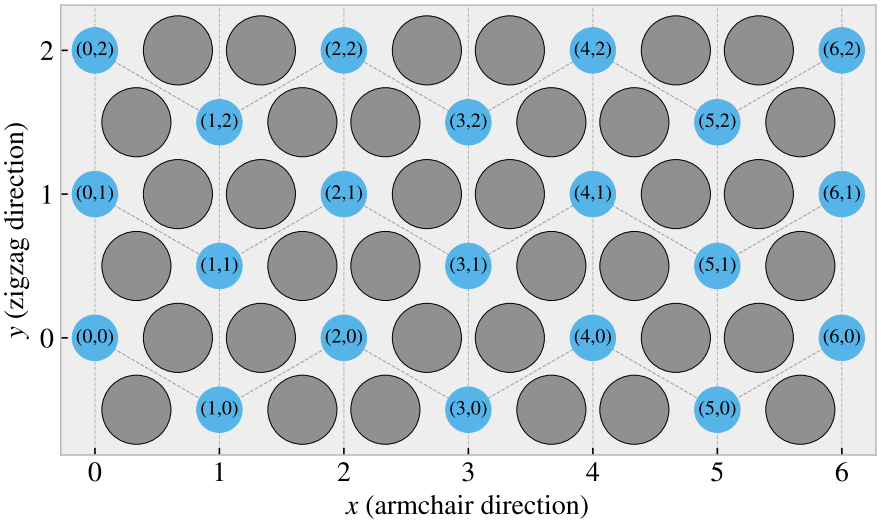


Figure 2.5: Graphene center indexing

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.

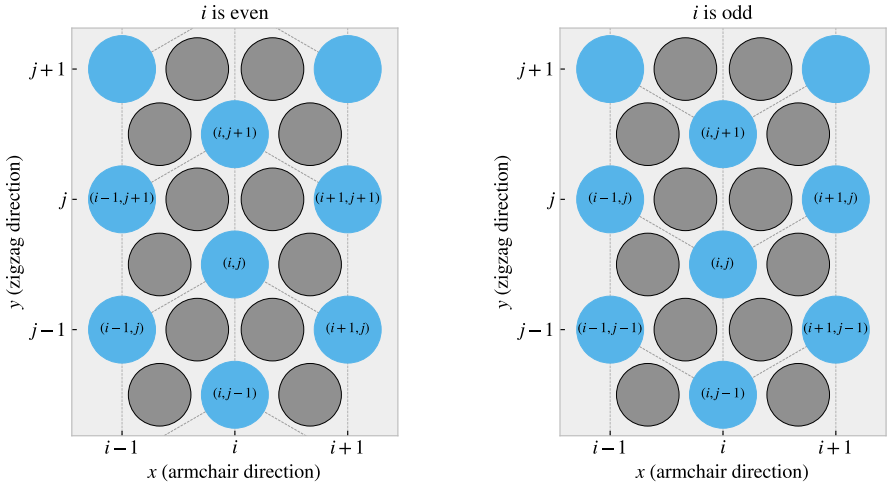


Figure 2.6: Graphene center elements directions

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 [15]. 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$^{\text{TM}}$ [16] 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 [15]. (b) Honeycomb: Scotch^{TM} Cushion Lock^{TM} [16] 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 eveness, we must in practice require that $|L_2 - L_1| = 2, 6, 10, \ldots$. In Fig. 2.8 we see a visual representation of the pattern components for the $(7, 5, 2)$ patteren.

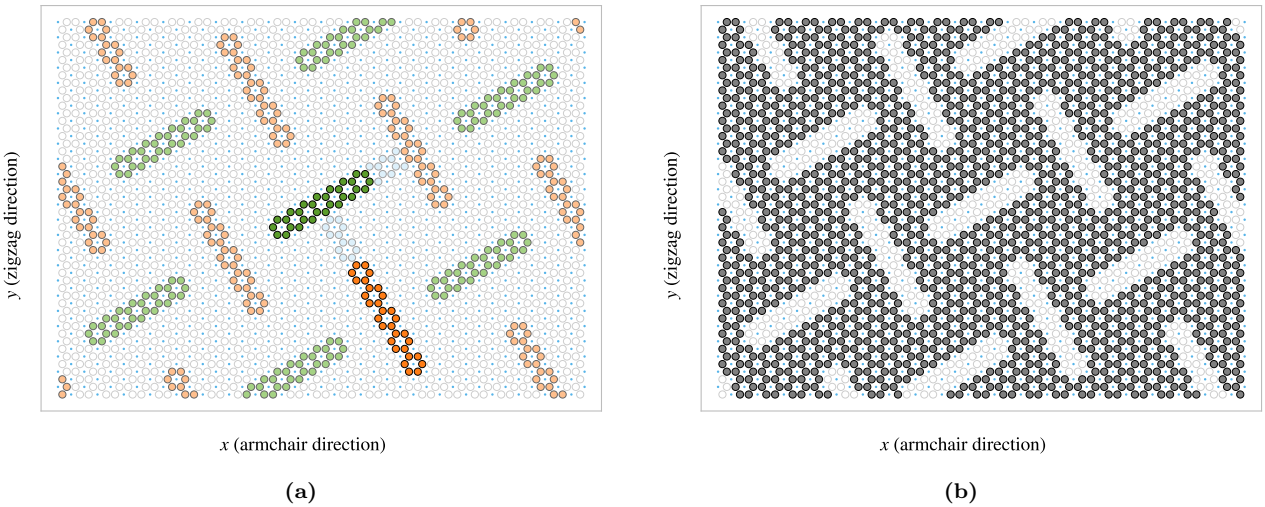


Figure 2.8: Visual representation of the tetrahedron pattern consisting of two perpendicular lines, line 1 and line 2, of length L_1 and L_2 respectively with spacing d. This example used $(L_1, L_2, d) = (7, 5, 2)$. (a) Highlight of the atoms removed. Line 1 is shown in green and line 2 in orange, with 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.

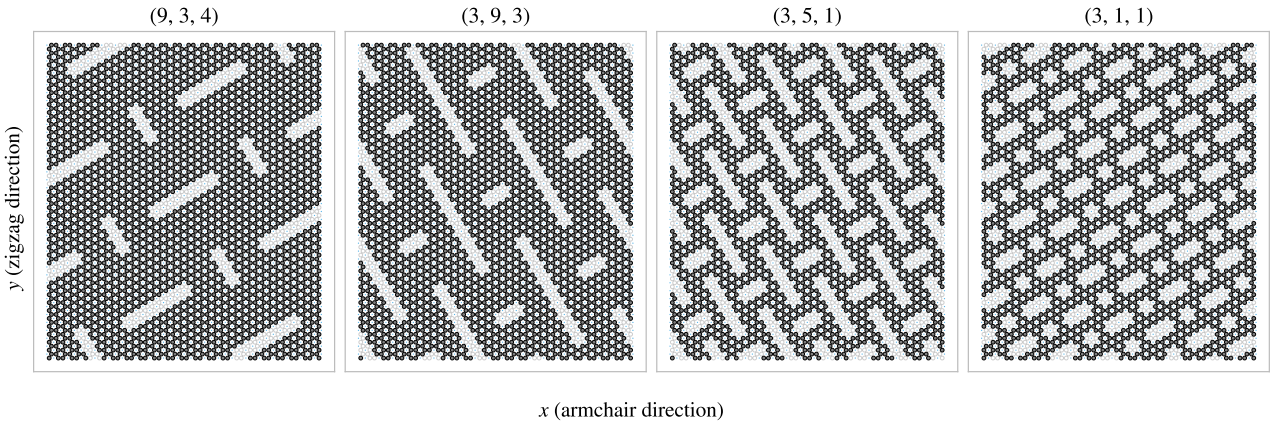


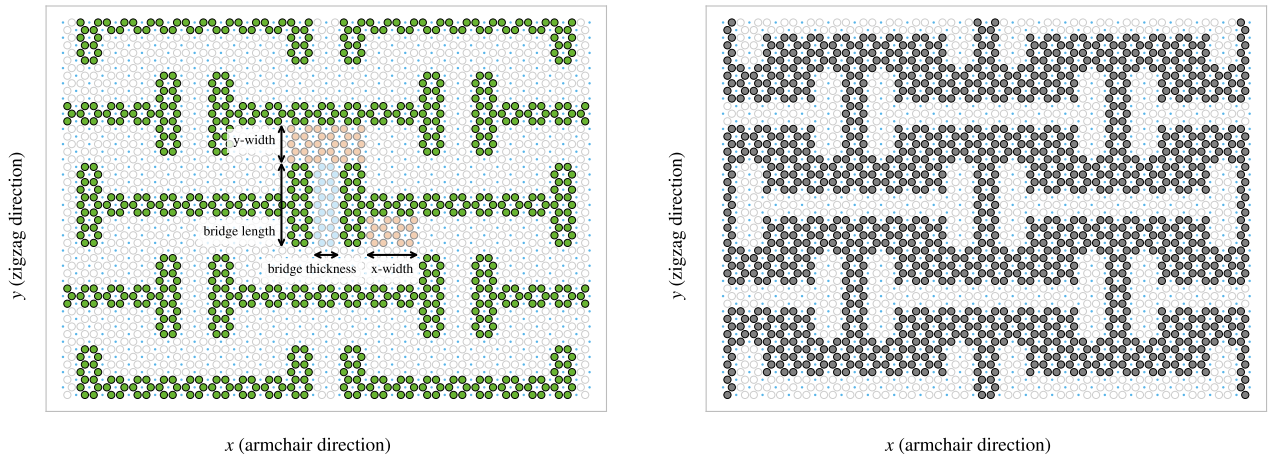
Figure 2.9: Sheet size used in example

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

[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.

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

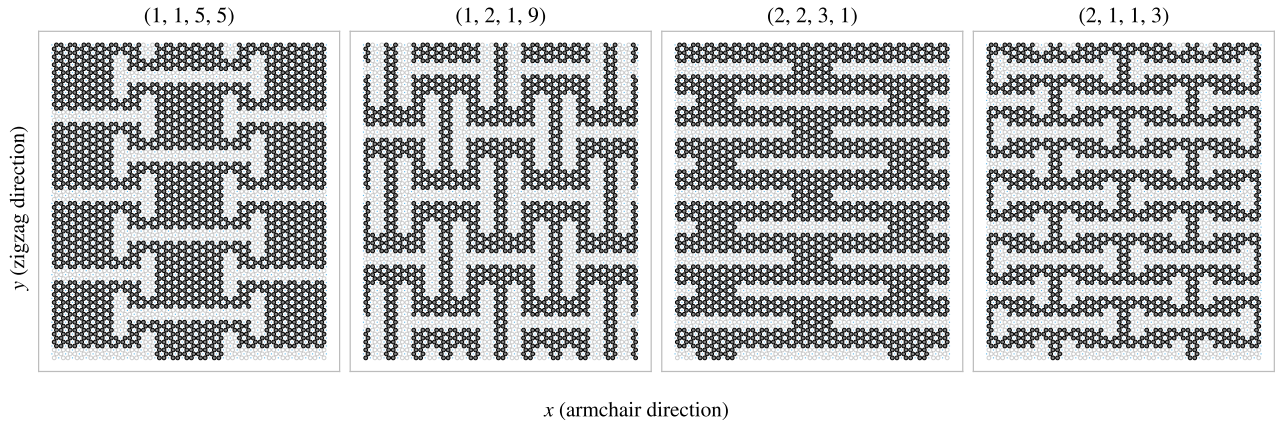


Figure 2.11

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 neighbouring sites are availble, note that it cannot backtrack its own path either, and in such a case the walk will be terminated dispite 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 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 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 is cummulated 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

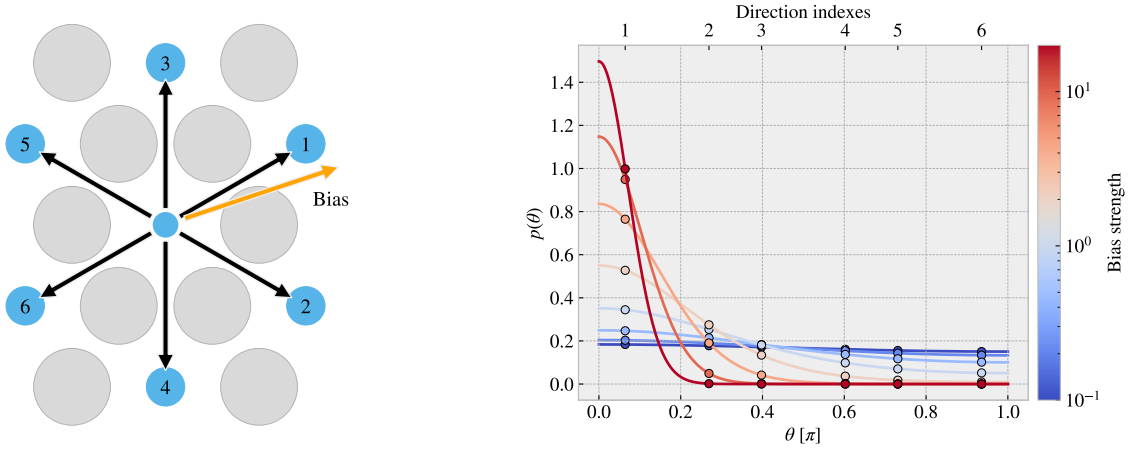
$$p_i = \frac{1}{Z} e^{-\beta E_i}, \qquad Z = \sum_i e^{-\beta E_i},$$

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.

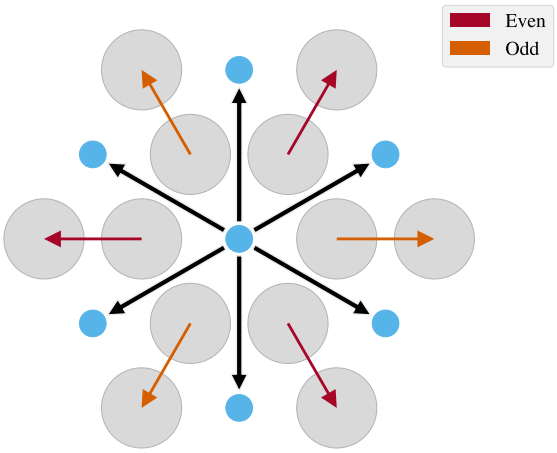


Figure 2.13: Caption

The *Stay or break* is still subject to previsouly defined rules, such that in the case that the site preferred site is not availble, it will either terminate when going there, or it is removed from the neighbour list when *avoid unvalid: True*. For the latter case the walker has broken its direction and will follow the new direction with probability p_{stay}

2.5.3.5 Deployment schemes

By default, each random walker is given an 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

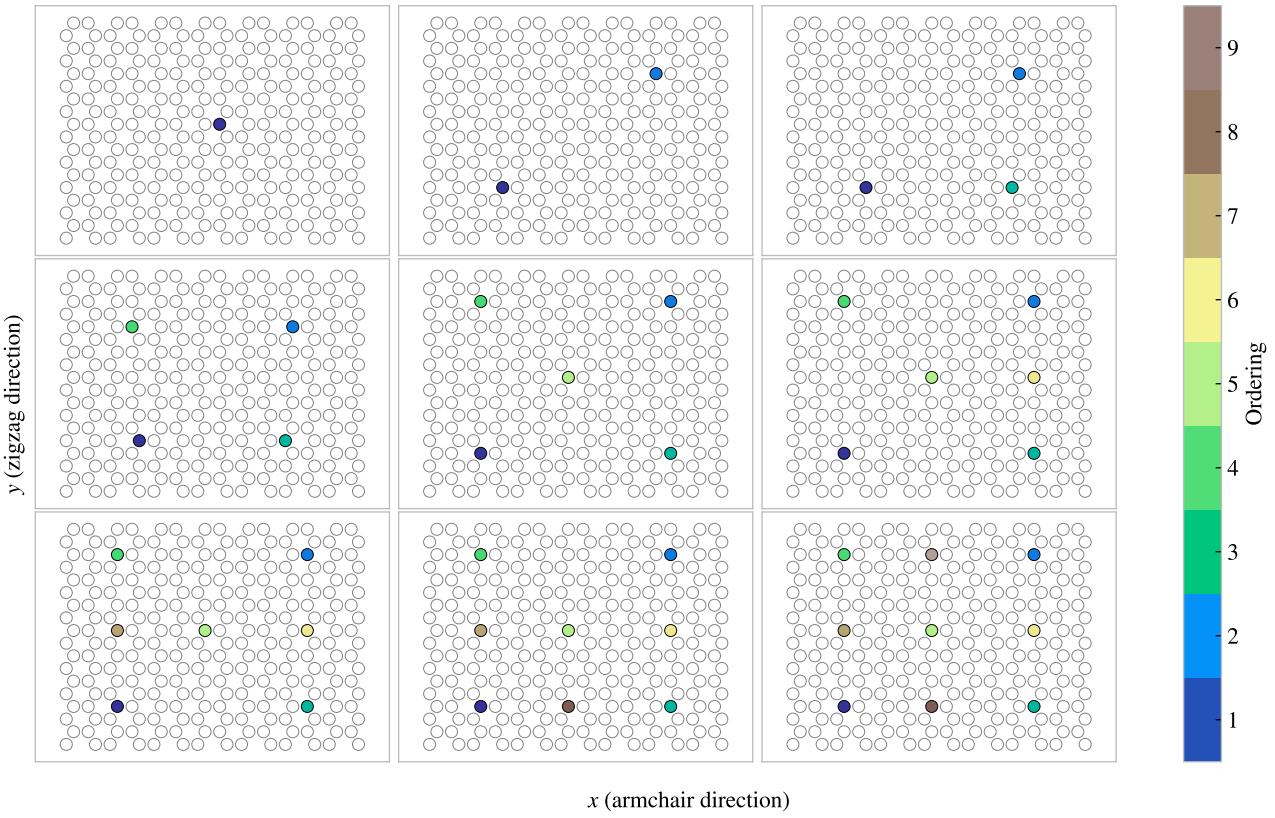


Figure 2.14: Populaiton of starting points with the centering parameter toggled on in a 14×18 sheet. This is shown for 1-9 number of random walkers with the color map conveying the order of the population.

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.

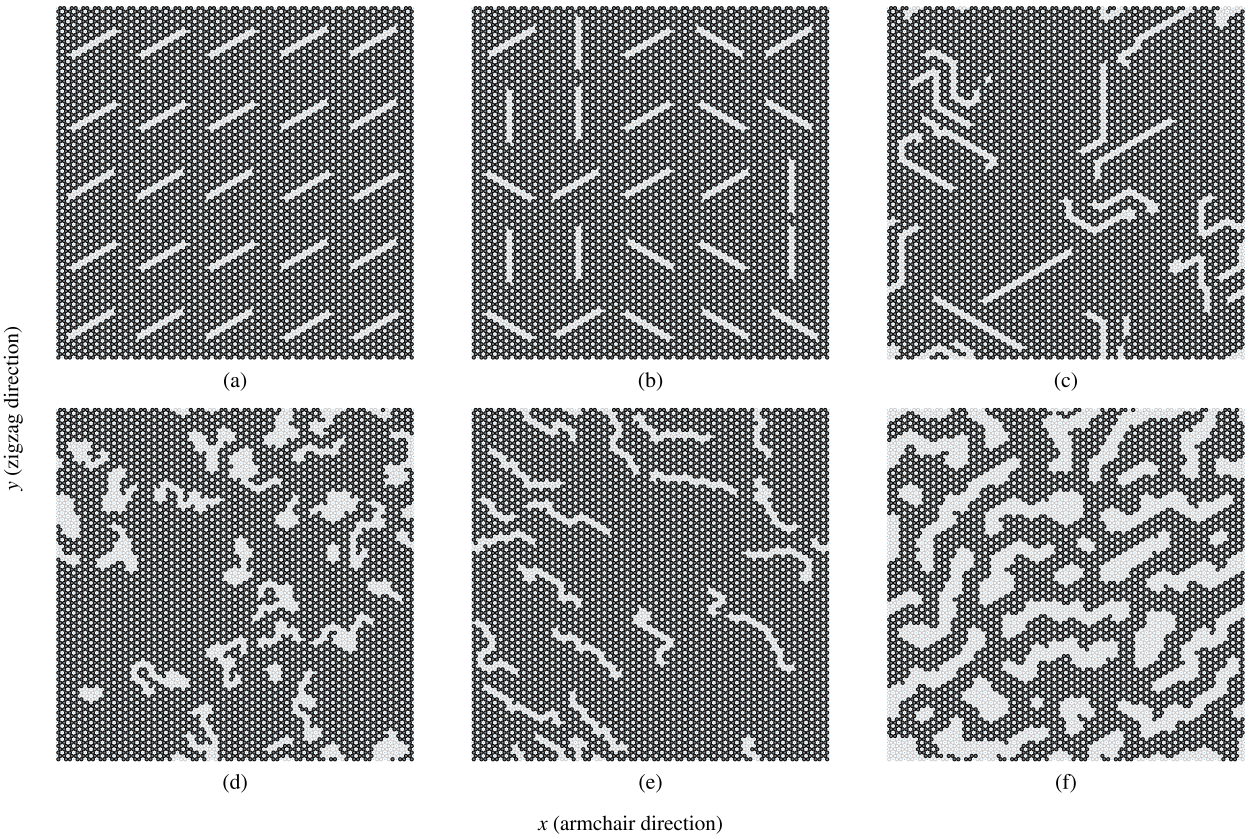


Figure 2.15: Some example uses of the random walking class. Give information of parameters?

Chapter 3

Main study

3.1 Generating data

The dataset consist of friction simulations of various cut configurations and combinations of normal load and stretch. For each configuration we sample 15 pseudo uniform (refer to relevant section here) strech values between zero and the rupture stretch found in the rupture test. The normal force is uniformly sampled in the range $[0.1, 10]$ nN. In total this gives 3×15 data points for each configuration. For the remaining parameters we use the values presented in the pilot study (see **??**). We generate 68 configurations of the Tetrahedron pattern type, 45 of the Honeycomb type and 100 of the Random walk type. A summary of the dataset is given in Table 3.1 while all configurations are shown explicitly in **??**. Notice that not all submitted data points "makes it" to the final dataset. This is due a small variation in rupture stretch points which were not anticipated during the creation of the numerical framework for submitting multiple simulation. After performing the rupture test the simulation is restarted with a new substrate size corresponding to the measured rupture stretch limit and also with new random velocity and thermostat initializations values. The sheet is then stretched and checkpoints of the simulation state (LAMMPS restart files) are saved for each of the targeted stretch samples. However, if the rupture points arrives slightly early than syggested by the rupture test, some sampled stretch values might not get a corresponding checkpoint file. Thus, these data points are not included in the data set even though they ideally should have been noted as a rupture event. This could quite easily have been mittigated by a rewrite of that part of the code, but it was first discovered after the dataset had been created. However, the dataset still includeds 11.57 % rupture events and it most likely that the most cases with a lost rupturer event have a rupture event stored for the preeceding stretch value instead which captures the information of the sheet stretch limit on its own.

Table 3.1: Summary of the number of generated data points in the dataset. Due to slight deviations in the rupture stretch and the specific numerical procedure not all submitted simulations "makes it" to the final dataset. Notice that the Tetrahedon (7, 5, 2) and Honeycomb (2, 2, 1, 5) from the pilot study is rerun as a part of the Tetrahedon and the Honeycomb datasets seperately. In the latter datasets the reference point for the pattern is randomized and thus theese configurations is not fully identical. This is the idea behind the difference of 2 in the total sum.

Type	Configurations	Submitted data points	Final data points	Ruptures
Pilot study	3	270	261	25 (9.58 %)
Tetrahedon	68	3060	3015	391 (12.97 %)
Honeycomb	45	2025	1983	80 (4.03 %)
Random walk	100	4500	4401	622 (14.13 %)
Total	214 (216)	9855	9660	1118 (11.57 %)

3.2 Data analysis

In order to gain insight into the correlations between variables associated to the simulations we calculate the correlations coefficients between all variable combinations. More specific, we are going to calculate the Pearson product-moment correlation coefficient (PPMCC) for which is defined, between data set X and Y, as

$$\mathrm{corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\langle (X-\mu_X)(Y-\mu_Y)\rangle}{\sigma_X \sigma_Y} \quad \in [-1,1]$$

where $\mathrm{Cov}(X,Y)$ is the covariance, μ the mean value and σ the standard deviation. The correlation coefficients ranges from perfect negative correlation (-1) through no correlation (0) to a perfect positive correlation (1). The correlation coefficients is shown in Fig. 3.1

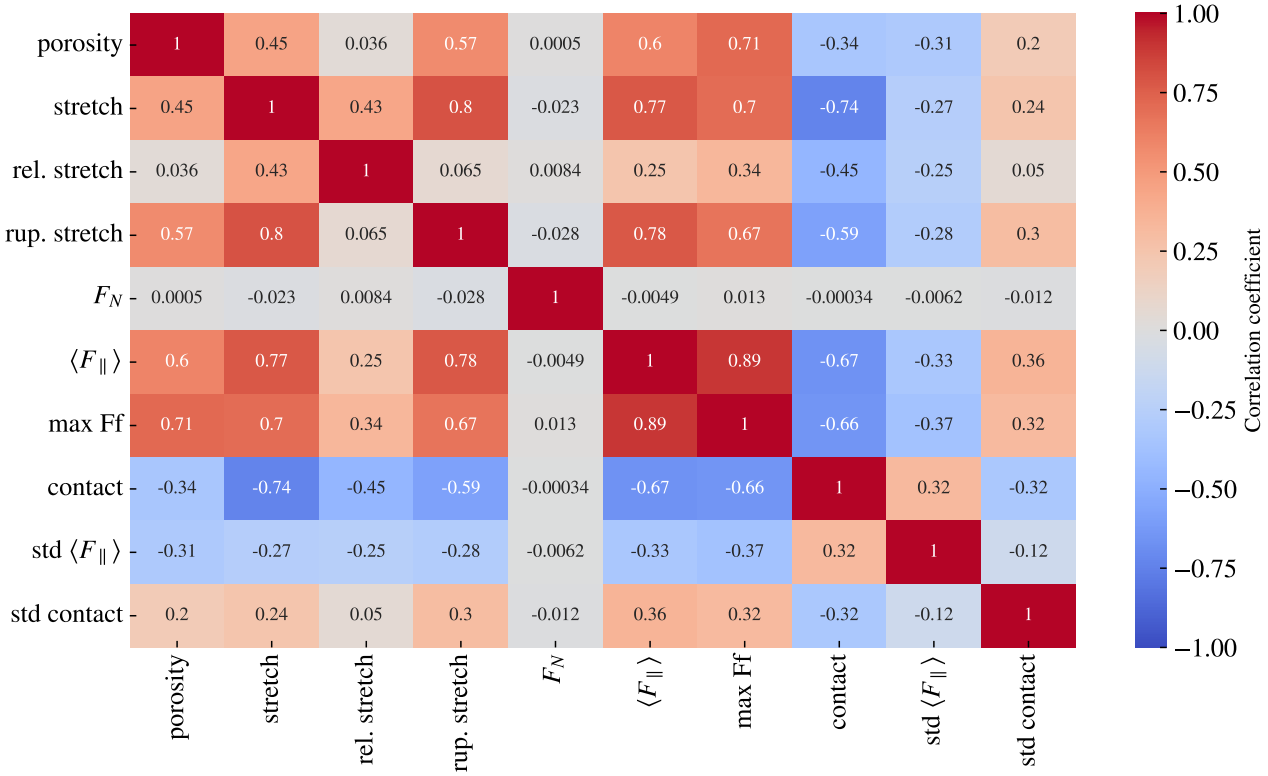


Figure 3.1: Pearson product-moment correlation coefficients for the full datset (see Table 3.1).

From Fig. 3.1 we especially notice that the mean friction force $\langle F_\parallel \rangle$ has a signifciant positively correlation with stretch (0.77) and porosity (0.60) (void fraction). However, the relative stretch, which is scaled by the rupture stretch, has a weaker correlation of only 0.25 which indicates that it is the absolute stretch value that has the most significant impact on the friction force increase during stretching. This is further supported by the fact that the mean friction and the rupture stretch is also strongly positively correlated (0.78). From figure Fig. 3.1 we also observe that the contact bond count is negatively correlated with the mean friction (-0.67) and the stretch value (-0.74) which is consistent with the trend observed in the pilot study **??** and **??**) of the contact decreasing with increasing stretch and mean friction. However, we must take note that the correlation coefficients is a measure of the strength and slope of a forced linear fit on the data. We clearly observed a non-linear relationship between stretch and mean friction for the tetrahedron and honeycomb pattern used in the pilot study **??**) where the relationship was partwise characterized by a postive correlation for some stretch ranges and partwise negative correlation for other stretch ranges. Hence, interesting strong regime-specific correlations might not be accurately highlighted by the correlation coefficients shown in Fig. 3.1.

In Fig. 3.2 we have visualized the data (excluding the pilot study) for chosen pairs of variables on the axes. In addition to a visual confirmation of how the given correlations look in a 2D plot we also get a feeling for the

coverage in various areas of the parameter space that we are eventually going to feed the neural network. The honeycomb pattern is spanning a significant larger range of stretch, contact and mean friction makes the data rather biased towards the Honeycomb pattern in thoose areas.

Uncommented to decrease loading time

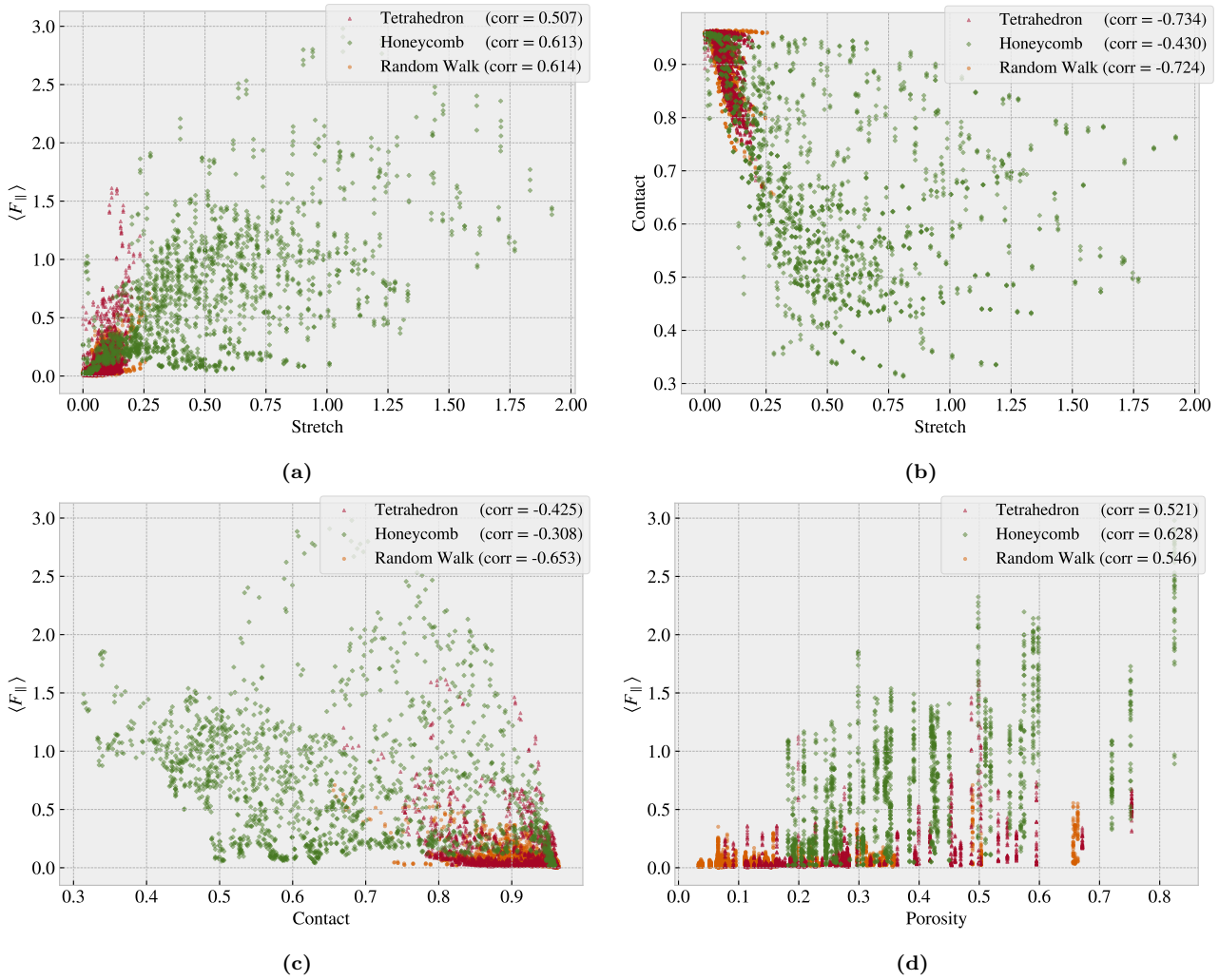


Figure 3.2: Scatter plot of the data sets Tetrahedron, Honeycomb and Random Walk (excluding the pilot study) for various variable combinations in order to visualize some chosen correlations of interest and distributions in the data

3.3 Properties of interest

From the Pilot study we discovered that it might be possible to achieve a negative friction coefficient for certain kirigami cut configurations under the assumption of a system with coupled normal force F_N and stretch S. This stands as the main property of interest to explore further in the dataset. However, it is not obvious how one should quanity this in a rigorous manner. The friction coefficient is by our definition (see theory sec XXX) given as the slope of the friction vs. normal force curve. For two data points $(F_{N,1}, F_{f,1}), (F_{N,2}, F_{f,2})$, $F_{N,1} < F_{N,2}$ we would evaluate the associated friction coefficient $\mu_{1,2}$ as

$$\mu_{1,2} = \frac{F_{f,2} - F_{f,1}}{F_{N,2} - F_{N,1}} = \frac{\Delta F_f}{\Delta F_N}$$

In the pilot study it became clear that the effects on friction under the change of F_N is neglible in comparison to the effects under change of S. Thus, by working under the assumption $F(F_N, S) \sim F(S)$ and a coupling

$F_N \propto R \cdot S$ with coupling ratio R we get

$$\mu_{1,2}(S_1, S_2) = \frac{\Delta F_f(S_1, S_2)}{R(S_2 - S_1)} \propto \frac{\Delta F_f(S_1, S_2)}{\Delta S}, \tag{3.1}$$

With the above reasoning we have in practive exchanged F_N with S in the expression for the friction coefficient. This means that we are interested in a negative slope on the friction vs. stretch curve which corresponds to a negative friction coefficient in our proposed coupled system. The next question remaining is then how to evaluate the strength of this property. By definition, the minimum slope value would give the lowest friction coefficient. However, for two data points with a small ΔS, corresponding to small denominator in Eq. (3.1), would potentially result in big $|\mu|$ without any signifiant decrease in friction. Hence, we choose to consider the drop in friction with increasing stretch. For a discrete dataset we can locate all local maxima and evaluate the difference to all succeeding local minema. The biggest drop will serve as our indicator for a signifcant negative friction coefficient. In this evaluation we do not guarantee a monotonic decrease of friciton in the range of the biggest drop, but when seearching among multiple configurations this is considered a descent strategy to highlight configurations of interest worthy of further investigation.

In addition to the biggest drop in friction we also look at minimum abnd maximum friction along with the difference between theese extrema. In Table 3.2 we summarized the extrema of these properties. The corresponding friction vs. stretch profiles and configurations are visualized for each property category in Fig. 3.3 to 3.6. The stretch profiles for all the configurations are shown in appendix **??**.

Table 3.2: Evaluation of the properties of interest for our dataset.

Tetrahedron	Configuration	Stretch	Value [nN]
Min F_{fric}	$(3, 9, 4)$	0.0296	0.0067
Max F_{fric}	$(5, 3, 1)$	0.1391	1.5875
Max ΔF_{fric}	$(5, 3, 1)$	$[0.0239, 0.1391]$	1.5529
Max drop	$(5, 3, 1)$	$[0.1391, 0.1999]$	0.8841

Honeycomb	Configuration	Stretch	Value [nN]
Min F_{fric}	$(2, 5, 1, 1)$	0.0267	0.0177
Max F_{fric}	$(2, 1, 1, 1)$	1.0654	2.8903
Max ΔF_{fric}	$(2, 1, 5, 3)$	$[0.0856, 1.4760]$	2.0234
Max drop	$(2, 3, 3, 3)$	$[0.5410, 1.0100]$	1.2785

Random walk	Configuration	Stretch	Value [nN]
Min F_{fric}	12	0.0562	0.0024
Max F_{fric}	96	0.2375	0.5758
Max ΔF_{fric}	96	$[0.0364, 0.2375]$	0.5448
Max drop	01	$[0.0592, 0.1127]$	0.1818

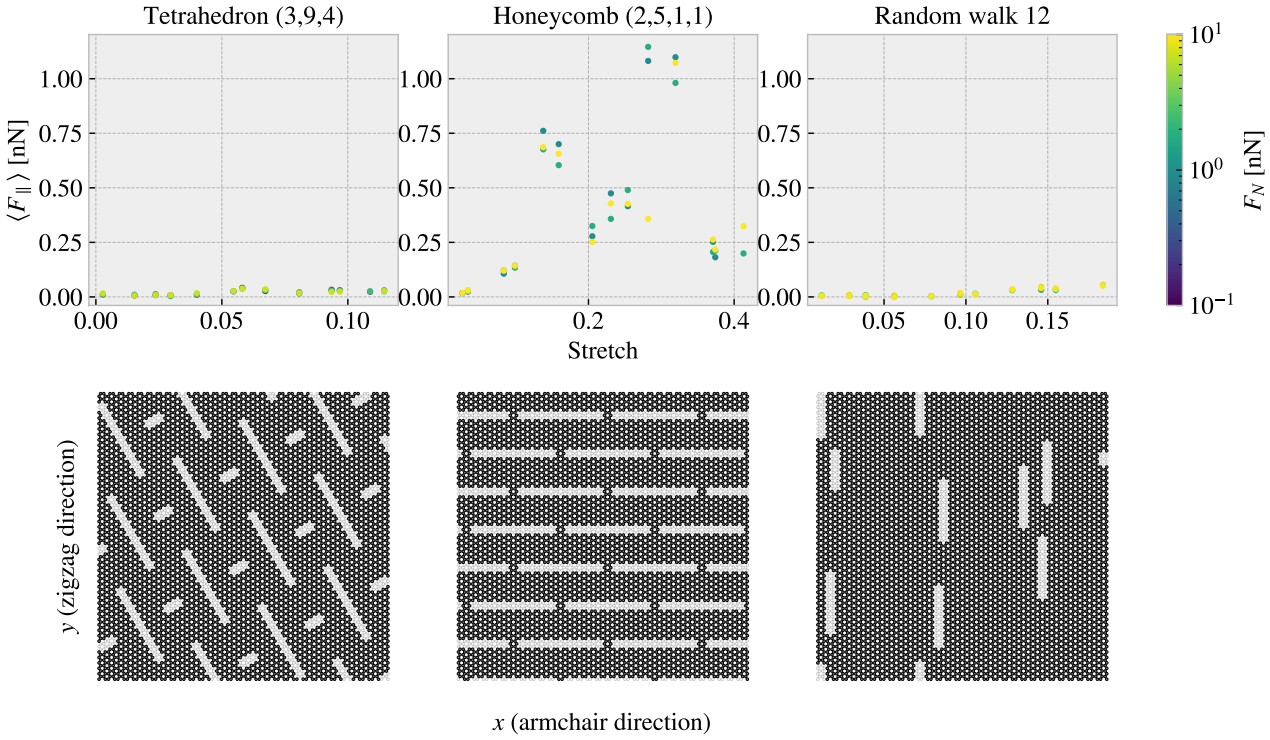


Figure 3.3: Minimum friction: Configurations corresponding to the minimum friction.

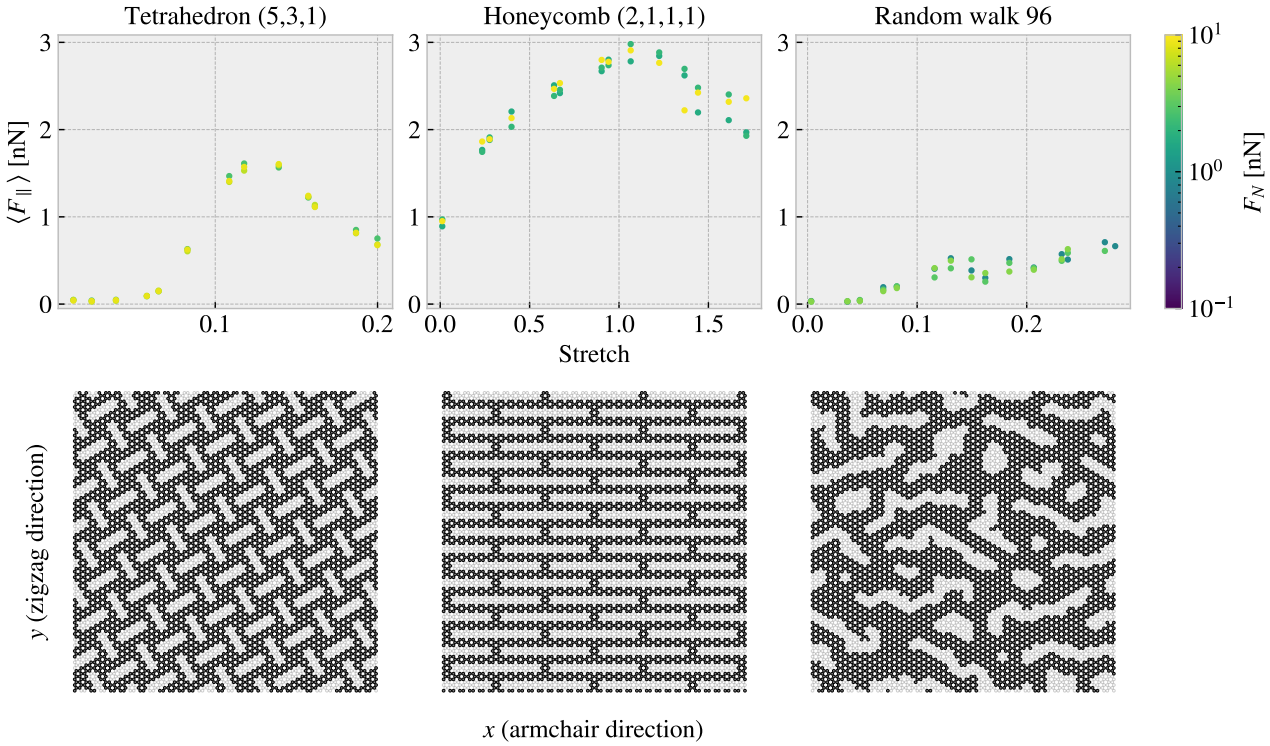


Figure 3.4: Maximum friction: Configurations corresponding to the maximum friction.

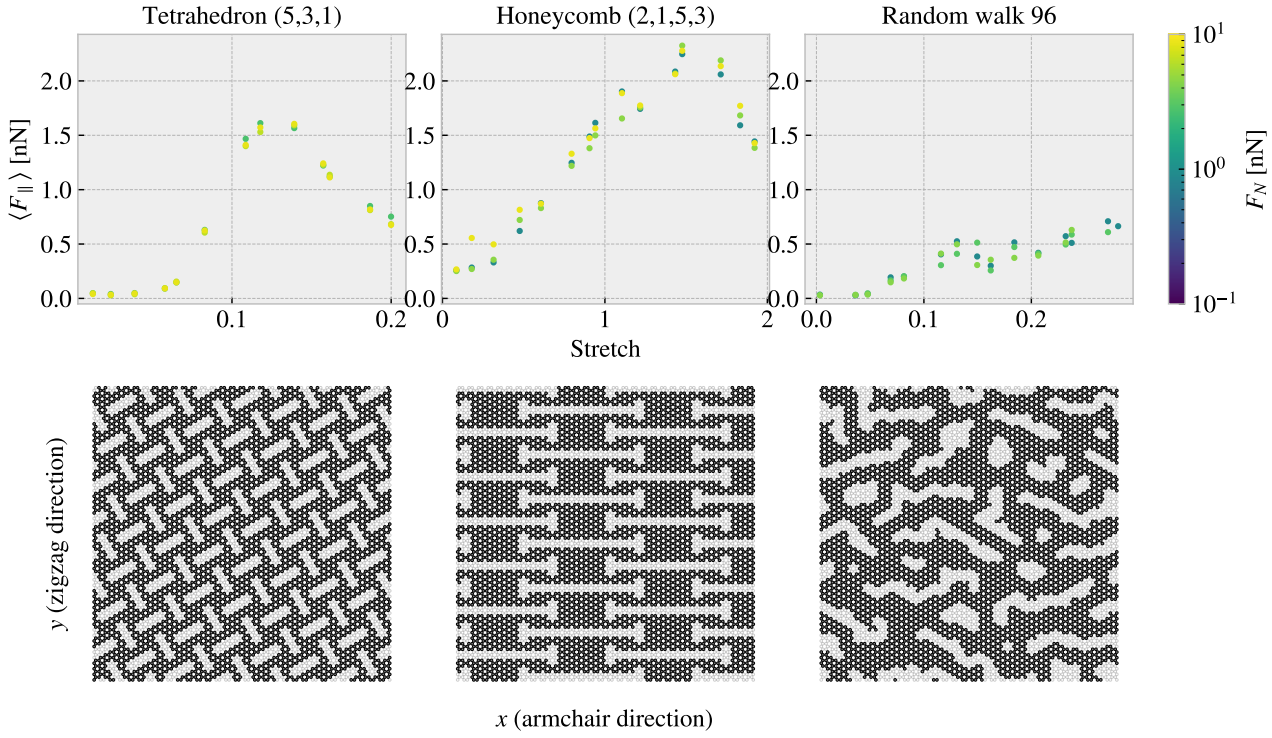


Figure 3.5: Maximum Difference: Configurations corresponding to the biggest difference in friction in the dataset for each pattern.

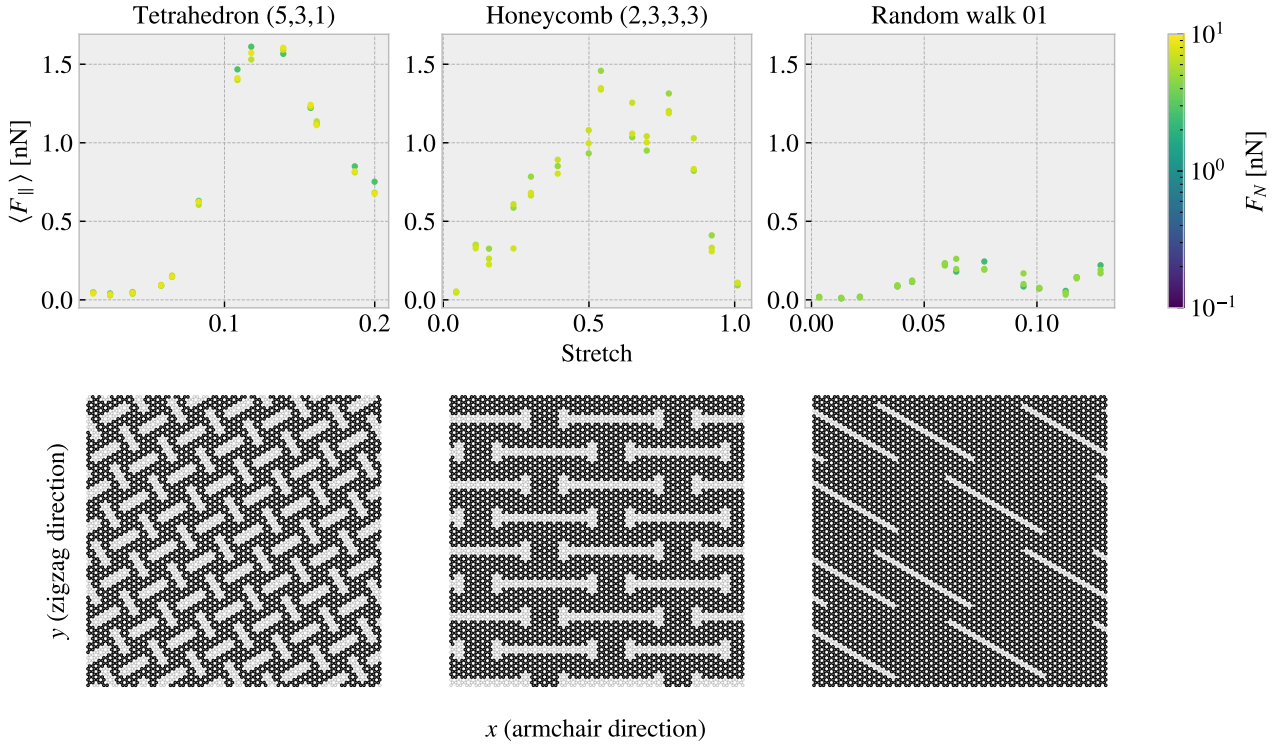


Figure 3.6: Maximum drop: Configuratiosn corresponding to the biggest friction drop in the dataset for each pattern.

3.4 Machine learning

General stuff to include. Remember to talk about batchnorm, optimizer, stopping at best epoch.

3.4.1 Architecture

Due to the spatial dependencies in the kirigami configurations we use a convolutional neural network (CNN). Studies on similar a similar system envolving the graphene sheet have used a VGGNet style of network, Hanakata et al. [6][7] and Wan et al. [8], which we adopt for this study as well. The VGGNet-16 architecture illustrated in Fig. 3.7 shows the key features;

- The image is processed through a series of 3×3 convolutional filters (the smallest size to capture spatial dependencies) using a stride of 1 with an increasing number of channels throughout the network. Each convolutional layer is followed by a ReLU acitivation.
- The spatial dimensions are reduced by a max pooling (2×2, stride of 2), which half the spatial resolution each time.
- The latter part of the network consist of fully connected part followed by a ReLU activation. The image is first processed in a 1×1 which performs a linear mapping to a series of fully connected layers.

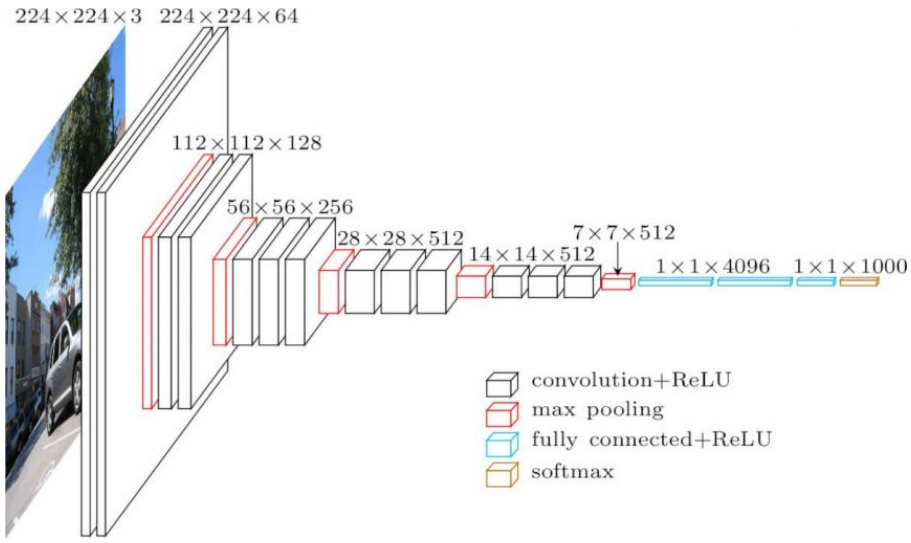


Figure 3.7: VGGNet 16. Source https://neurohive.io/en/popular-networks/vgg16/.

In contrast to the VGGNet-16 we restrict ourselves to building the convolutional part in terms of blocks of (convolution, ReLU, max pooling), thus not allowing for any connsecutive convolutional filters without performing a max pooling as well. The fully connected blocks is defined as (fully connected, ReLU) similar to that of the VGGNet model. Hanakata et al. and Wan et al. used a similar construction before settling on the models

$$\begin{aligned} \text{Hanakata et al. [6]} \quad & C16\ C32\ C64\ D64, \\ \text{Wan et al. [8]} \quad & C16\ C32\ D32\ D16, \end{aligned}$$

Where C denotes a convolutional block with the following number being the number of channels and D a fully conneted (dense) layer with the number denoting number of nodes. For the process of determing a suiting complexity for the architecture we adpot the approach by Wan et al. [8] who used a "staircase" pattern for combinning convolutional and fully connected blocks. By defining a starting number of channels S and network depth D we fill the first half with convoliutional blocks doubling in channel number for each layer and the latter half with fully connected blocks starting setting the number of nodes as the reverse pattern used for the number

of channels. Following this pattern a $(S = 4, D = 8)$ would take the form

$$\text{Input} \rightarrow \overbrace{\underbrace{C4}_{S=4}\ C8\ C16\ C32\ \underbrace{D32\ D16\ D8\ D4}_{D=8}}^{} \rightarrow \text{Output}.$$

This provides a simple description where S and S and can be varied systematically for a grid search over architecture complexity.

3.4.2 Data handling

3.4.2.1 Input

We use three variables as input: Kirigami configuration, stretch of the sheet and applied normal load. While the first is a two-dimensional input the latter are both scalar values. This gives rise to two main options for the data structure

1. Expand the scalar values (stretch and load) into 2D matrices of the same size as the kirigami configuration by copying the scaler value to all positions. This can then be merged into an image of three channels used as a single input.

2. Pass only the kirigami configuration through the CNN part of the network and introduce the remaining scalar values into the FC part of the network.

Both options utilize the same data, but the first emphasizes that the configurations should be processed in relation to the applied stretch and load, while the latter represent a more independent processing. We implemented the option to do both variations, but it quickly became clear that option 1 was producing the most promising result (hldo more rigorous presentation of this?).

3.4.2.2 Output

For the output we are mainly concerned about the mean friction and the rupture detection. In combination this make us able to produce a friction vs. stretch curve with an estimated stopping point as well. However, it has often been proven useful to introduce more variables in the output in order to strengthen the network training (get source). In addition, this gives us more options for exploring the relationship in the data later on. Therefore, we include maximum friction, contact count, porosity and rupture stretch in the output as well. Notice that rupture stretch refers to the value found in the rupture test without load, but as the sheet always ruptures before or just around this point in a loaded state this provides some information for the training to lean on, even though it is in the output state. In principle, we could add a penalty whenever the network predicts the sheet to be attatched for stretch values above the rupture stretch, but we found the performance of the rupture prediction to be satisficatory without such penalties. Notice that we weight the importance of these variables differently as explained in the section regarding the loss.

3.4.2.3 Data augmentation

In order to increase the ulility of the limited data availble, one can introduce data augmentation. For classification task this includes distortions such as color shift, zoom, flip etc. However, such distorions are only valid since the classification network should still classify a cat as a cat even though it is suddenly a bit brigther or flipped upside down. For our problem we can only use augmentation that matches a physical symmetry. Such a symmetry exist only for relfection accross the y-axis. We cannot do this across the x-axis as the sheet is translated in a positive y-direction meaning that the reflected version would not be sliding backwards for which we do not expect to be symmetric in results. We definitely expect a snow plow to perform differently when attached in reverse and thus by analogy we would expect the direction of sliding with respect to the configuration to be of importance.

3.4.3 Loss

The output contain two different types of variables: scalar values and binaray values (0: False 1: True)

For the scalar values we use the Mean Squared Error (MSE)

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

where N is the number of data entries and y are the true variables and $\hat{y}$ are the predicted values. For the binary output we use binary cross entropy

$$L_{\text{BCE}} = -\frac{1}{N}\left[\sum_{i=1}^{N}[t_i \log(p_i) + (1-t_i)\log(1-p_i)]\right],$$

where $t \in \{0, 1\}$ is the truth label. Does this belong in theory entirely?. We calculate the total loss as a weighted sum between the loss associated with each variable

$$L_{tot} = \sum_{v} W_v \cdot L_v.$$

We choose the weights to be $1/2$ for the mean friction and $1/10$ for the remaining 5 variables thus sharing evenly for the remaining 50% of the weight. During the introductionary phase of the training we tried varying these weights, but we found that the results varied little and concluded that the training is not very sensitive to this choice, and disregarded further tuning of this parameter.

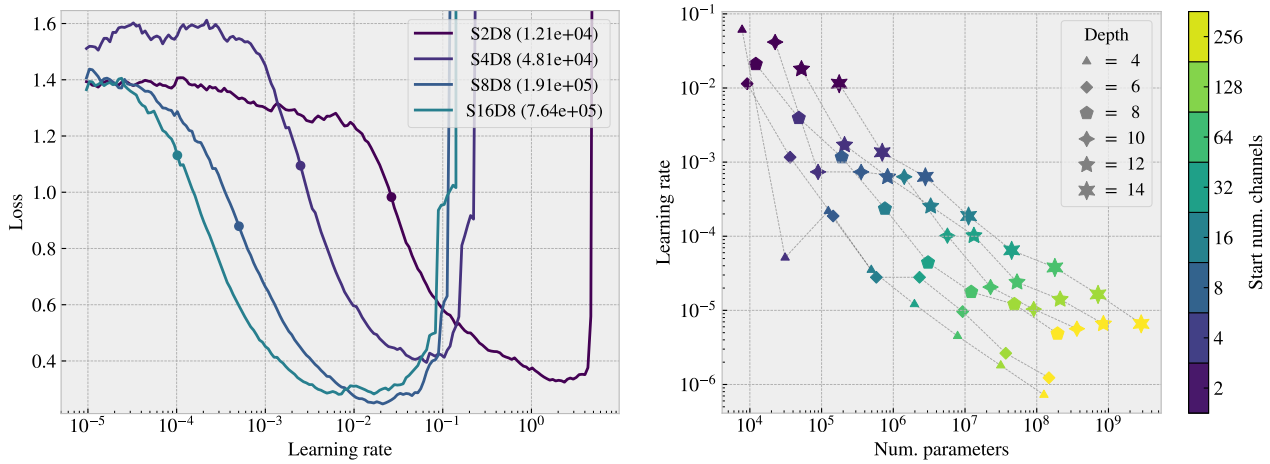
3.4.4 Hypertuning

For the hypertuning of ML parameters we focus on architecture complexity, learning rate, momentum and weight decay. We train with the adam optimizer with the default values of $beta_1 = 0.9, \beta_2 = 0.999$ and zero weight decay. For the batch size we use 32 for training and 64 for validation. We train the network for a maximum of 1000 epochs, but we save the best model during training based on lowest validation loss. Since the learning rate is considered to be one of the most important hyperparameters we will determine a suitable choice for the learning individually throughout the two major grid searches:

1. Architecture complexity grid search of S vs. D with individually chosen learning rates for each complexity combination.

2. Momentum vs. weight decay grid search with learning range chosen with regard to the momentum setting.

We consider first the architectures in the range $S \times D = \{2, 4, 8, 16, 32, 64, 128, 256\} \times \{4, 6, 8, 10, 12, 14\}$. For each architecture complexity we perform an initial learning rate range test, increaing the learning rate for each batch iteration until the training loss diverges. the suggested learning rate is then determined as the point for which the training loss decreases most rapidly. The learning rate is increased exponentially from 10^{-7} to 10 with increments for each training batch iteration. This is done for just a single epoch where a training batch size of 32 yields a total of 242 batches in the training data. This corresponds to an exponent increment of approximately $1/30$ giving a relative increase $10^{1/30} \sim 108\%$ per batch iteration. The learning rate range test is presented in Fig. 3.8. We notice that the suggested learning rate decreases with increasing number of model parameters. This decrease is further independent on the specific relationship between S and D.



(a) **(b)**

Figure 3.8: Learning rate range test for various model complexities. We increase the learning rate exponentially from nume-7 to 10 during one epoch corresponding to an exponent increment of roughly 1/30 per batch iteration. (a) shows a few examples of the training loss history as a function of learning rate. The examplatory architectures are S[2, 16]D8 with the corresponding number of model parameters shown in parentheses in the legend. The dot indicates the suggested learning rate at the steepest decline of the slope. (b) shows the full results of suggested learning rates depending on the number of model parameters with color coding differentiating the number of start channels and marker types differentiating different model depths.

With the use of the suggested learning rates from Fig. 3.8 we perform a grid search over the corresponding S and D parameters. We evaluate both the validation loss and the mean friction R_2 score which is shown in Fig. 3.9 together with the best epoch and the number of model parameters. Additionally, we evaluate the mean friction R_2 score for a selected set of configurations. This set consist of the top 10 configurations with respect to maximum friction drop for the Tetrahedron and Honeycomb pattern resepctively. This is done as a way of evaluating the performance on the non-linear stretch curve which showed to be the more difficult patterns to learn. The selected evaluation is shown in Fig. 3.10. Note that these patterns are already a part of the full datset and thus the data points related to these patterns are most likely present in both the training and the validation data set. Hence we cannot regard this as a validation set and the performance must be considered in conjuntion with the actual validation performance in Fig. 3.9.

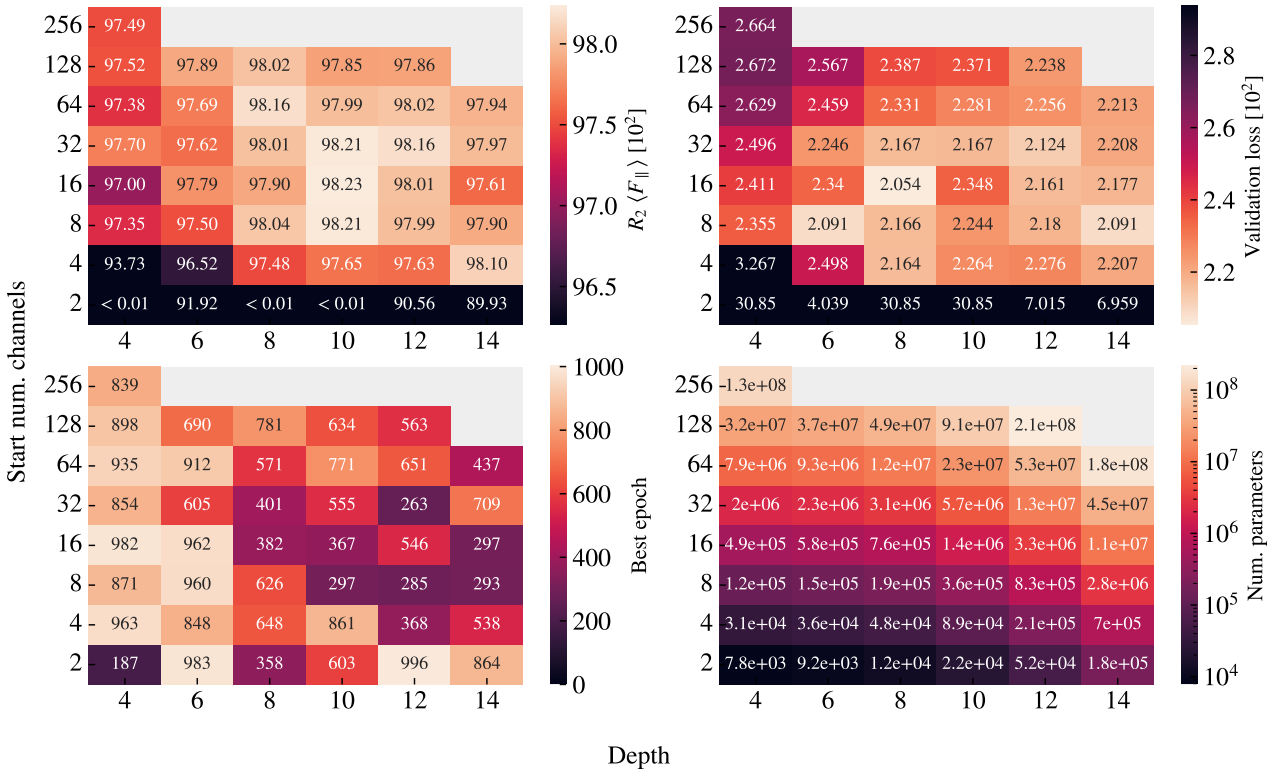


Figure 3.9: Architecture search.

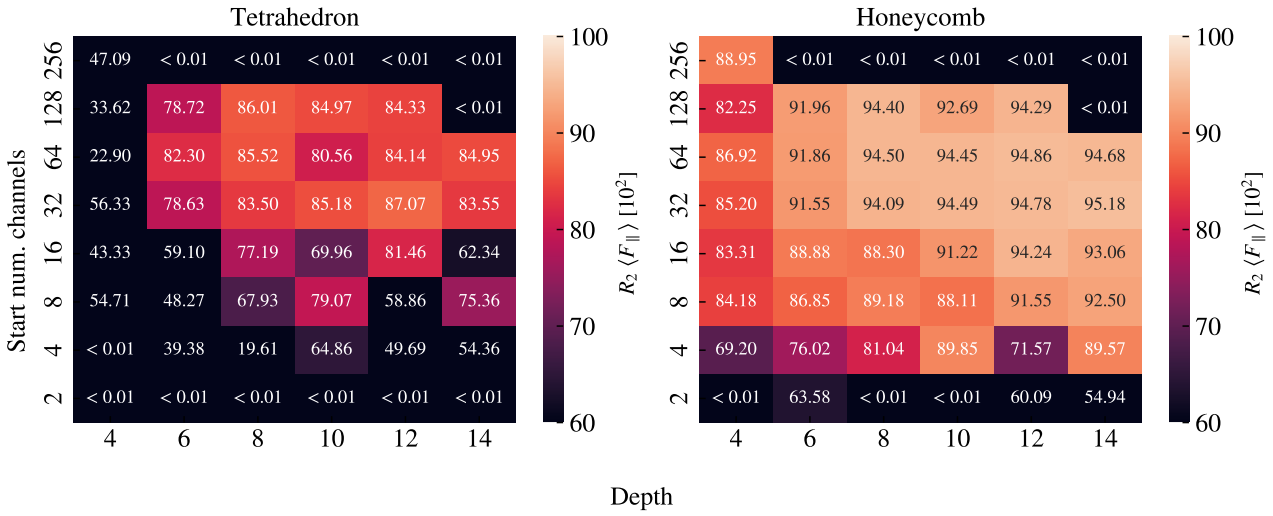
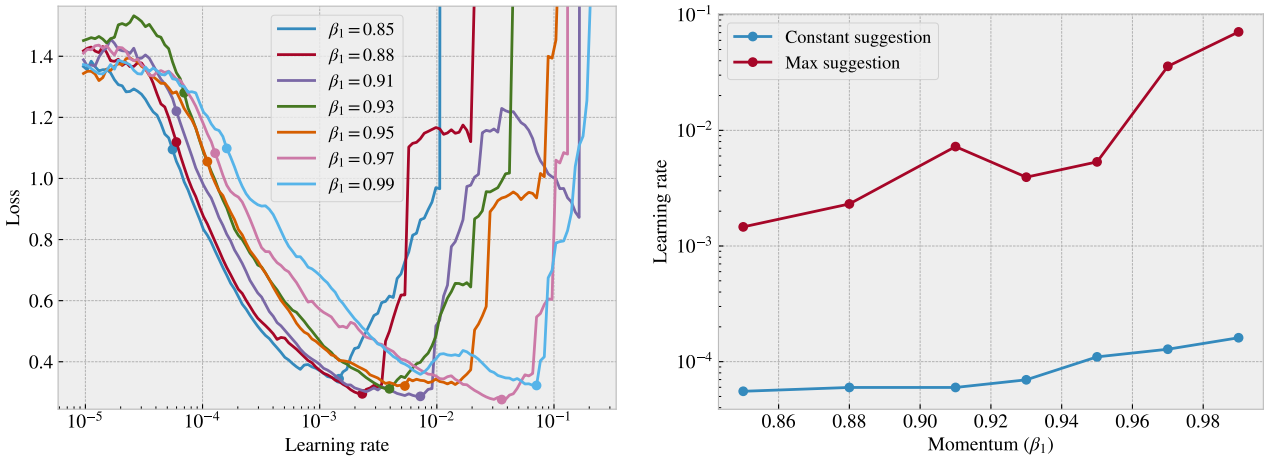


Figure 3.10: Selected pseudo validation set. Fix the missing grey fields in the top which are replaced by ¡ 0.01.

From the validation scores in Fig. 3.9 we find that models S(8-32)D(8-12) gives reasonble performance. When looking at the best epoch we find that models of low depth result in a later best epoch which is compatible with underfitting. As the depth is increased we find more models with a lower best epoch, in the range $\sim [300, 600]$, which on the other hand suggest cases of overfitting. Since our training stores the best model during training, we do not have to worry too much about overfitting, but we can take this transistion from underfitting to overfitting as a sign that out search is conducted in an appropiate complexity range. When consulting evaluation on the selected set in Fig. 3.10 we notice in general that we get lower R_2 scores, especially for the Tetrahedron pattern,

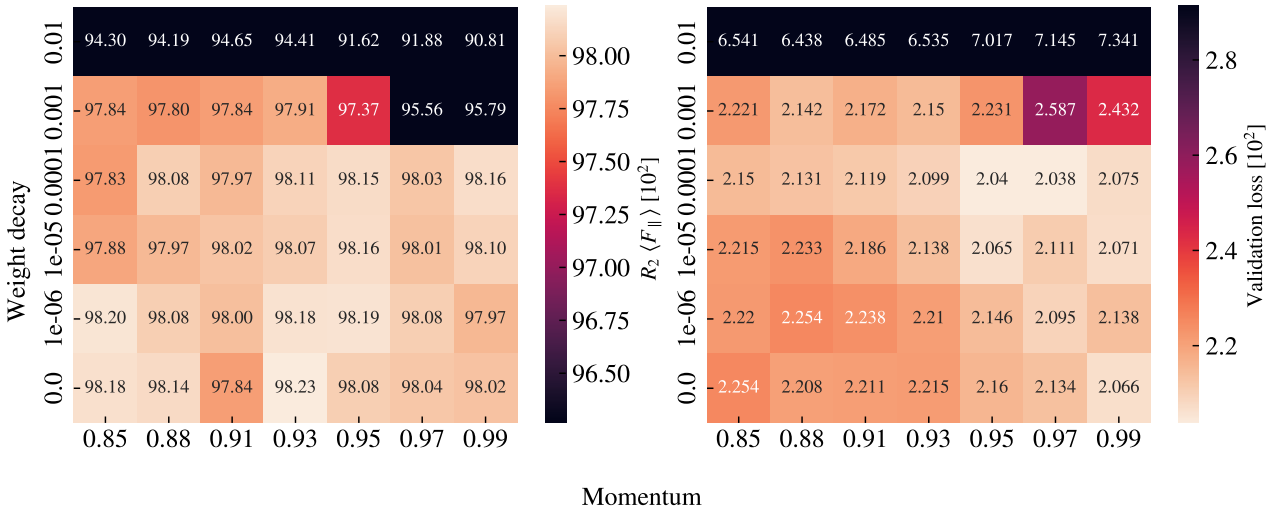
which shows that these constitute a challenge for the network. Considering that some of these datapoints is also present in the training data makes it even more clear that the network is not capturing the complexity in the data here. While the peak R_2 value for the validation score was found for the S16D10 model we see slightly preference for more complexity in the model with regard to the selected test. In the Tetrahedron grid search we find the best model to be S32D12, with a R_2 score of $\sim 87\%$. This model choice is more or less compatible with the overall performance as this is among the top candidates for the for R_2 and loss in Fig. 3.9 and the R_2 score for the Honeycomb pattern in Fig. 3.10 as well. Hence, we settle on this model moving on to the setting of the momentum and weight decay parameter.

We consider momentum m and weight decays wd in the range $m \in [0.85, 0.99]$ and $wd \in [0, 1e-2]$. An increased momentum is expected to decrease the appropiate learning rate, and thus we perform a new learning rate range test to determine a suitable choice for the learning rate for each momentum choice. We propose two learning rate schemes: A constant learning rate as used until this point and a one cycle policy. In the one cycle policy we set a maximum bound for the learning rate and start from a factor $1/20$ of this bound and increase towards the maximum bound during the first 30% of the training. We then decreases towards a final minimum being a factor $1e-4$ of the maximum bound during the remaining 70% of training. The increase and decrease is done by a cosine function. The suggested learning rate for the constant learning rate scheme is once again determined by the steepest slope on the loss curve while the maximum bound used for the one cycle policy is determined as the point of diverging. We find that the minimum point on the loss curve is as suitable choice that approaches the diverging point without getting to close and causing diverging learing. The learning rate range test for momentum is shown in Fig. 3.11. Using the results for the momentum learning rate range test we perform a grid search of momentum and weight decay. We examine again the validation loss and validation mean friction R_2 score in addition to the friciton mean R_2 score for the selected set of Tetrahedron and Honeycomb patterns. This is shown for the constant learning rate scheme in Fig. 3.12 and for the cyclic scheme in Fig. 3.13.

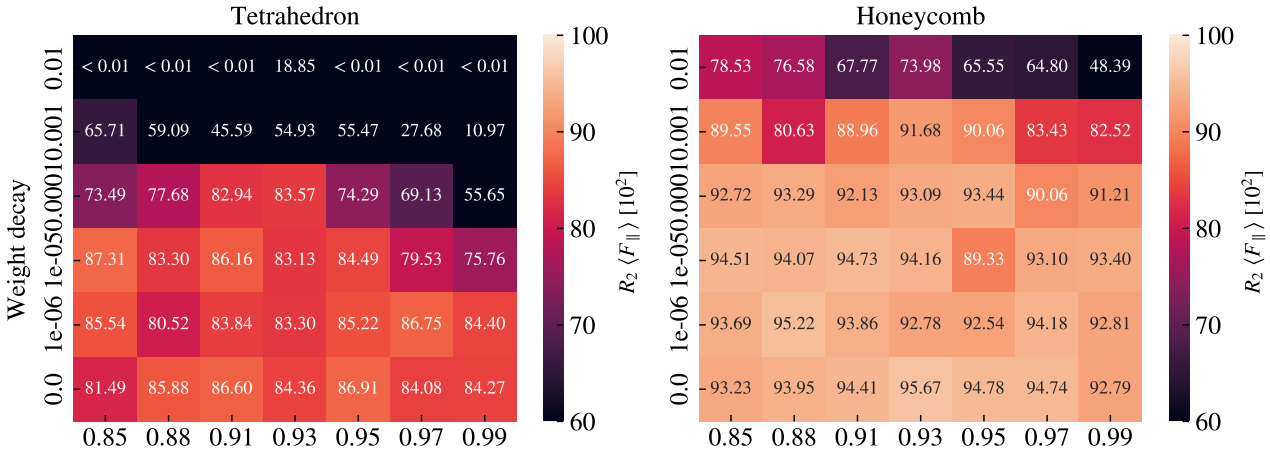


(a) **(b)**

Figure 3.11: Momentum learning rate range tets

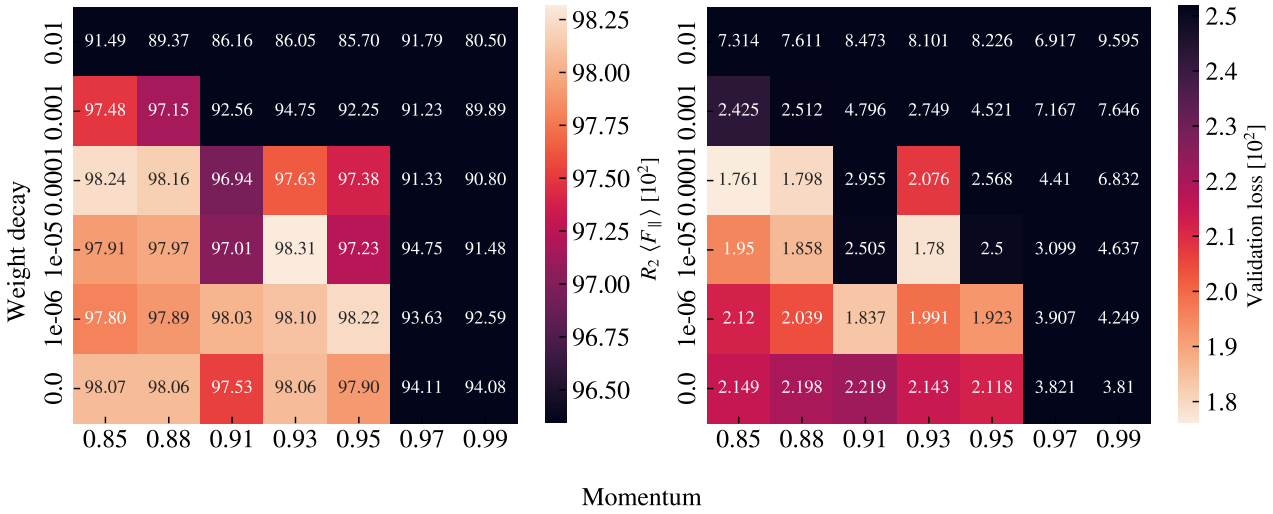


(a) Performance

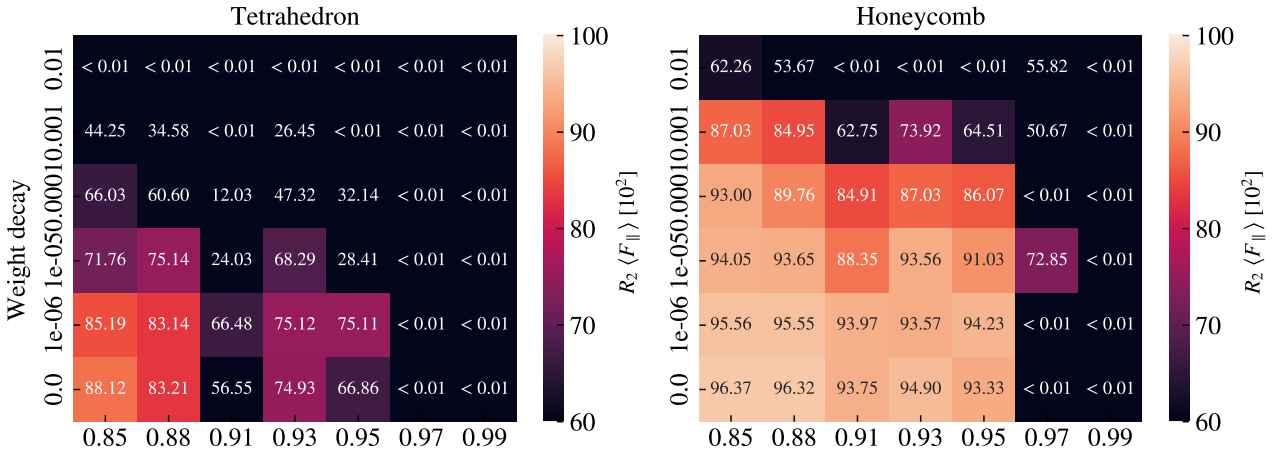


(b) Compare

Figure 3.12: Constant learing rate and momentum scheme



(a) Performance



(b) Compare

Figure 3.13: Cyclic learing rate and momentum scheme

The original validation scores, before varying momentum and weight decay, were a validation loss of 0.02124 and a mean friction R_2 score of 0.9816. By varying momentum and weight decay we are able to improve the performance metrics slightly for the constant learning rate scheme (loss: 0.02038, R_2: 0.9823) and even more for the cyclic scheme (loss: 0.0176, R_2: 0.9831), however notice that these scores a taking for their individual optimial hypersettings. The comparison among best scores are summarized in Table 3.3. In general the constant scheme shows reasonble stable results for all momentum settings $m \in [0.85, 0.99]$ in combination with a low weight decay $wd \leq 10^{-4}$. For the cyclic scheme the performance peaks towards a low momentum $m \leq 0.93$ and low weight decay $wd \leq 10^{-4}$. Looking at the summary in Table 3.3 we see that the cyclic scheme is able to produce a high score among all four performance metrics, however since these do not share common hyperparameters we need to make a choice here.

Table 3.3: Momentum and weight decay grid search using S32D12 model.

		Score [10^2]	Momentum	Weight decay
Validation loss	Original	2.124	0.9	0
	Constant	2.038	0.97	10^{-4}
	Cyclic	1.761	0.85	10^{-4}
Validation R_2	Original	98.16	0.9	0
	Constant	98.23	0.93	0
	Cyclic	98.31	0.93	10^{-5}
Tetrahedron R_2	Original	87.07	0.9	0
	Constant	87.31	0.85	10^{-5}
	Cyclic	88.12	0.85	0
Honeycomb R_2	Original	94.78	0.9	0
	Constant	95.67	0.93	0
	Cyclic	96.37	0.85	0

Look at overfitting via training history.

3.4.5 Final model

From the hypertuning study we choose the S32D12 model with a cylic training scheme with momentum 0.85 and weight decay 0. The mainperformance metrics are shown in Table 3.4. Since the porosity is a number between 0 and 1 we can interpret the absolute error as the percentage error similar to the relativ error for the rupture stretch. The rupture stretch is generally within a 13 % margin, but the metrics for the selected set might indicate that the relative error might be boosted due to some low stretch rupture cases in the data. The stretch curves for mean friction, max friction and contact is shown in Fig. 3.14 for the Tetrahedron (7, 5, 1) and Honeycomb (2,2,1,5) used in the pilot study. This gives a visual interpretation of how well the fits are for given R_2 scores, and we notice that a R_2 score above 0.98 is certainly promising for capturing the non-linear in the data.

Table 3.4: Mean values are used over different configurations.

	Loss [10^2]	R_2 [10^2]			Abs. [10^2]	Rel. [10^2]	Acc. [10^2]
	Total	Mean F_f	Max F_f	Contact	Porosity	Rup. Stretch	Rupture
Validation	2.1488	98.067	93.558	94.598	02.325	12.958	96.102
Tetrahedron	4.0328	88.662	85.836	64.683	01.207	05.880	99.762
Honeycomb	8.6867	96.627	89.696	97.171	01.040	01.483	99.111

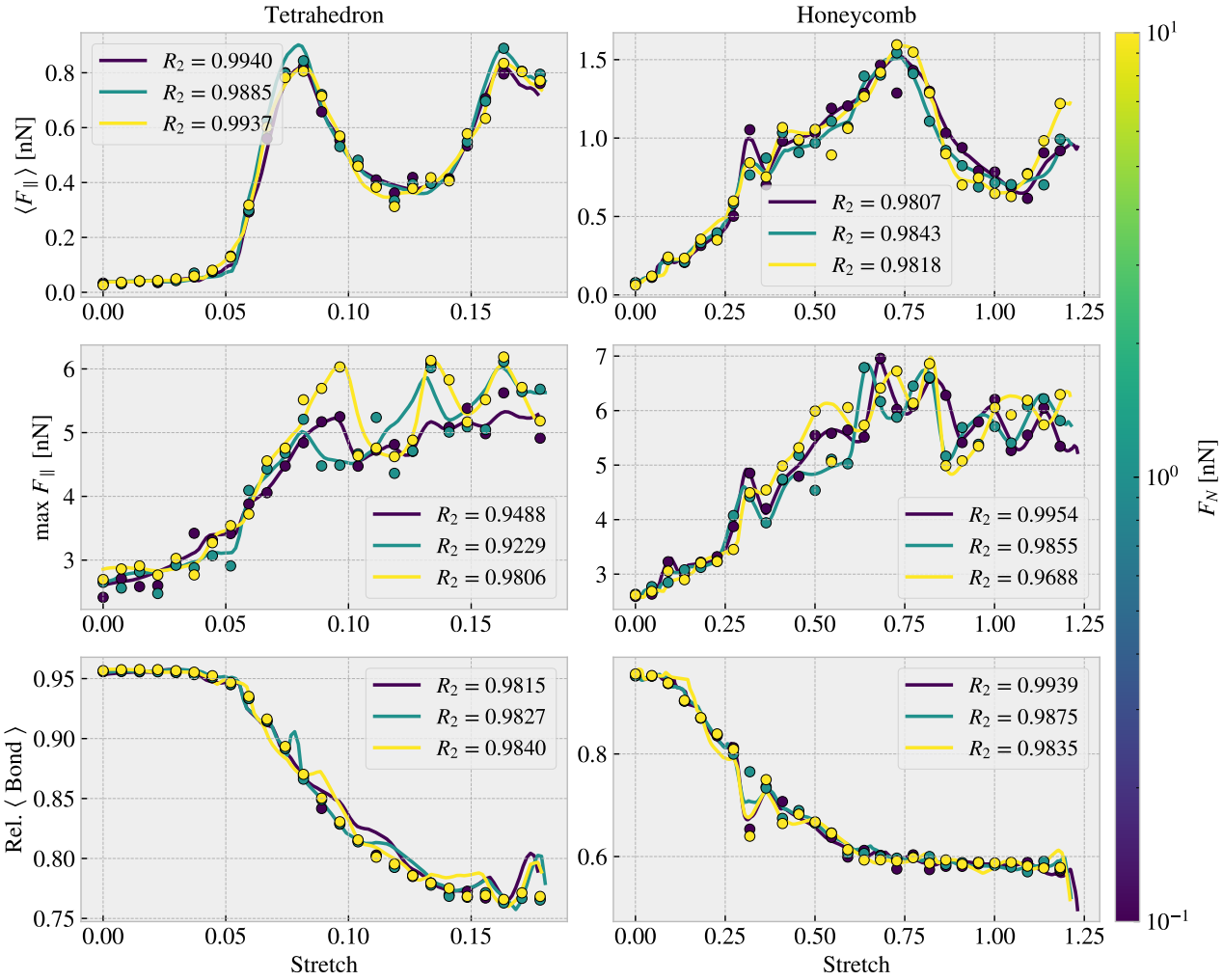


Figure 3.14: With 10^3 points in the stretch range [0, 1.5] and stopping after first rupture True prediction.

We take a look at how the ranking of the four properties of interest is ranked with the ML model as seen in Table 3.5 Table 3.6 Table 3.7. Generally the ranking for the minimum friciton is deviating the most. We also see negative values in the ML prediction which hints that the sensitivity to the low values is not good enough. Otherwise, the ML is pretty much able to get the top 5 rights for the remaning categories for the tetrehedron and honeycomb pattern. For the random walk it struggles a bit more. Looking at the values for the top candidates in the max-types properties we see that it is generally within a ~ 0.2 nN range.

Table 3.5: Tetrahedon

ML Rank	Data Config	Data Value [nN]	ML Config	ML Value [nN]	Data Rank
Min F_{fric}					
20	(3, 9, 4)	0.0067	(3, 1, 2)	0.0041	5
5	(3, 1, 3)	0.0075	(1, 3, 4)	0.0049	11
6	(5, 3, 4)	0.0084	(1, 3, 3)	0.0066	6
21	(1, 7, 3)	0.0084	(3, 1, 4)	0.0066	8
1	(3, 1, 2)	0.0097	(3, 1, 3)	0.0078	2
Max F_{fric}					
1	(5, 3, 1)	1.5875	(5, 3, 1)	1.5920	1
2	(1, 3, 1)	1.4310	(1, 3, 1)	1.2739	2
4	(3, 1, 2)	1.0988	(9, 3, 1)	1.1162	4
3	(9, 3, 1)	1.0936	(3, 1, 2)	0.7819	3
5	(7, 5, 1)	0.7916	(7, 5, 1)	0.7740	5
Max ΔF_{fric}					
1	(5, 3, 1)	1.5529	(5, 3, 1)	1.5578	1
2	(1, 3, 1)	1.3916	(1, 3, 1)	1.2331	2
4	(3, 1, 2)	1.0891	(9, 3, 1)	1.0807	4
3	(9, 3, 1)	1.0606	(3, 1, 2)	0.7778	3
5	(7, 5, 1)	0.7536	(7, 5, 1)	0.7399	5
Max drop					
1	(5, 3, 1)	0.8841	(5, 3, 1)	0.8603	1
2	(3, 5, 1)	0.4091	(3, 5, 1)	0.3722	2
4	(7, 5, 1)	0.3775	(1, 1, 1)	0.2879	5
5	(9, 7, 1)	0.2238	(7, 5, 1)	0.2478	3
3	(1, 1, 1)	0.1347	(9, 7, 1)	0.1302	4

Table 3.6: Honeycomb

ML Rank	Data		ML		Data Rank
	Config	Value [nN]	Config	Value [nN]	
Min F_{fric}					
1	(2, 5, 1, 1)	0.0177	(2, 5, 1, 1)	0.0113	1
9	(2, 4, 5, 1)	0.0187	(2, 5, 5, 3)	0.0149	7
7	(2, 4, 1, 1)	0.0212	(2, 5, 5, 1)	0.0182	4
3	(2, 5, 5, 1)	0.0212	(2, 5, 3, 1)	0.0186	5
4	(2, 5, 3, 1)	0.0226	(2, 4, 1, 3)	0.0198	15
Max F_{fric}					
1	(2, 1, 1, 1)	2.8903	(2, 1, 1, 1)	2.9171	1
2	(2, 1, 5, 3)	2.2824	(2, 1, 5, 3)	2.4004	2
6	(2, 1, 3, 1)	2.0818	(2, 1, 5, 1)	2.1060	5
4	(2, 1, 3, 3)	2.0313	(2, 1, 3, 3)	1.9458	4
3	(2, 1, 5, 1)	2.0164	(2, 4, 1, 1)	1.9381	6
Max ΔF_{fric}					
1	(2, 1, 5, 3)	2.0234	(2, 1, 5, 3)	2.1675	1
2	(2, 1, 1, 1)	1.9528	(2, 1, 1, 1)	2.0809	2
3	(2, 4, 1, 1)	1.8184	(2, 4, 1, 1)	1.9157	3
4	(2, 1, 3, 3)	1.7645	(2, 1, 3, 3)	1.6968	4
5	(2, 4, 1, 3)	1.4614	(2, 4, 1, 3)	1.5612	5
Max drop					
1	(2, 3, 3, 3)	1.2785	(2, 3, 3, 3)	1.3642	1
2	(2, 1, 3, 1)	1.1046	(2, 1, 3, 1)	0.9837	2
3	(2, 3, 3, 5)	0.8947	(2, 3, 3, 5)	0.9803	3
4	(2, 1, 5, 3)	0.8638	(2, 1, 5, 3)	0.9556	4
13	(2, 5, 1, 1)	0.8468	(2, 4, 5, 3)	0.8999	8

Table 3.7: RW

ML Rank	Data Config	Data Value [nN]	ML Config	ML Value [nN]	Data Rank
Min F_{fric}					
1	12	0.0024	12	-0.0011	1
24	76	0.0040	06	0.0036	27
6	13	0.0055	14	0.0074	23
31	08	0.0065	05	0.0082	19
26	07	0.0069	63	0.0085	?
Max F_{fric}					
3	96	0.5758	99	0.5155	2
1	99	0.5316	98	0.4708	3
2	98	0.4478	96	0.4356	1
4	97	0.3624	97	0.3503	4
11	58	0.3410	55	0.2817	7
Max ΔF_{fric}					
3	96	0.5448	99	0.4669	2
1	99	0.4769	98	0.4314	3
2	98	0.4085	96	0.4128	1
4	97	0.3268	97	0.3080	4
78	57	0.2978	55	0.2542	7
Max drop					
3	01	0.1818	00	0.1883	3
2	96	0.1733	96	0.1654	2
1	00	0.1590	01	0.1532	1
11	37	0.1022	04	0.0591	8
28	34	0.0879	56	0.0552	20

3.5 Accelerated Search

Having a network model that can predict friction force for a given configuration are able to search for some desired properties. Low and high friction and maximal negative friction coefficients

Here we pursue two different approaches for finding

1. Generate an enlarged dataset and run it through the ML model
2. Genetic algorithm

3.5.1 Generation search

Judging from the ranking in Table 3.5 Table 3.6 Table 3.7, the ML model showcases a reasonable performance with respect to giving a qualitative pointer towards interesting configurations.

We perform a search through the pattern generator class. For the tetrahedron and honeycomb repeating each pattern 10 times with random reference points. For the random walk we take a Monte Carlo approach by doing random sampling in a parameter range. Here we use the repair functions for the sheet. For each ML prediction we use 100 points in the stretch range 0%-200%. Say more about how many configurations and setting in general.

In the search we got very mixed results and even if we only searched through patterns used in the dataset the ranking was suddenly completely different. The only changed aspect were the reference point which was now randomized. Thus, by looking at all reference point options systematically for some selected configurations we discovered that it is extremely sensitive to this parameter.This can be attributed either to overfitting in the model or simply a lack of data, but since a translation of the pattern is not guaranteed to give similar result, we have no way of knowing whether this is actually physical valid. The system could be very sensitive to edge effects and we would simply have to simulate this in order to get more information about this.

This sensitivity was most significant for the max drop property. Thus we look at the change in the predictions for all reference positions for the top candidates for the Tetrahedron (5,3,1) and the Honeycomb pattern (2,3,3,3) repsectively as shown in Fig. 3.15.

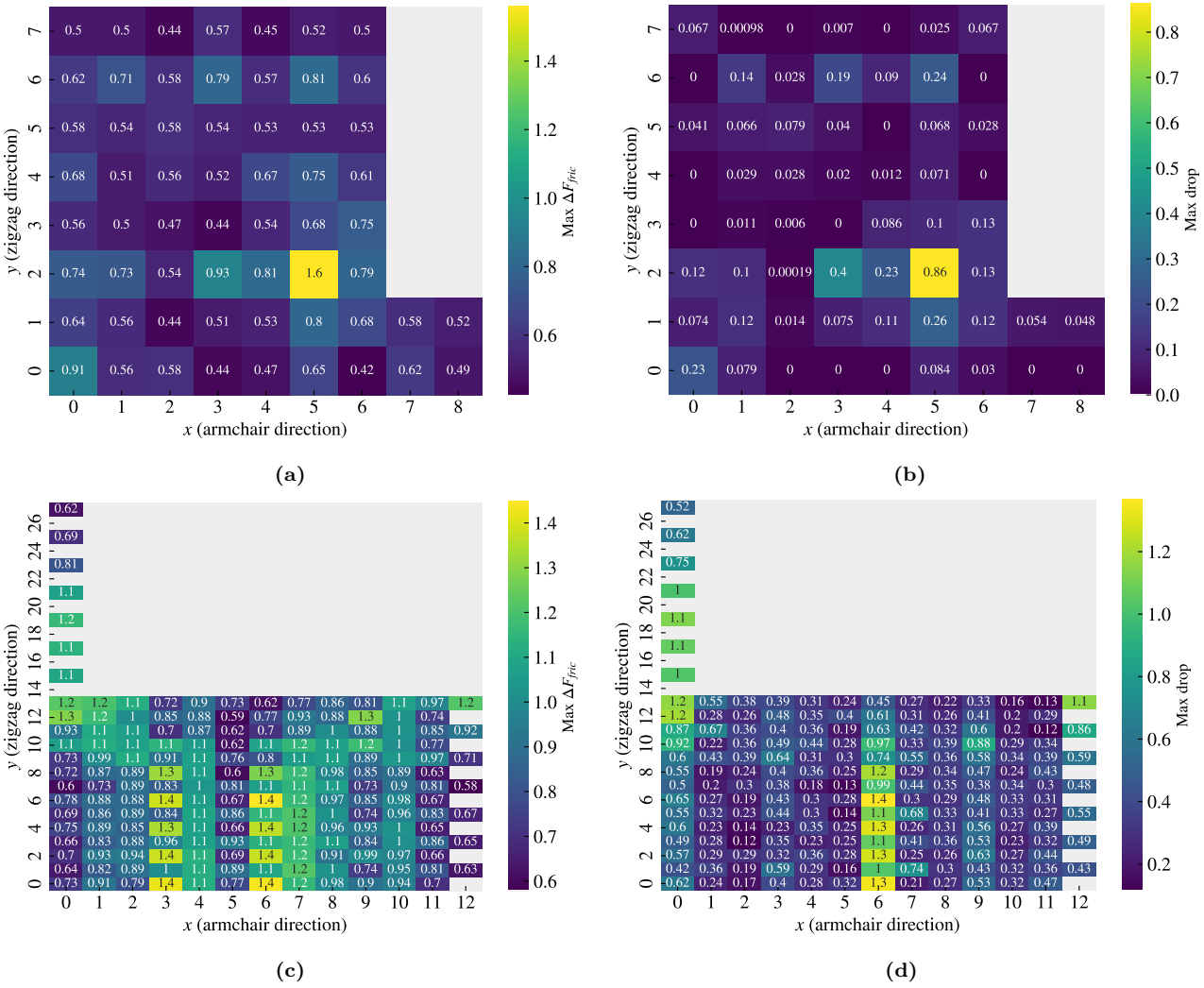


Figure 3.15: ...

First of all, the search gave a rather consistent result regarding the minimization of friction where the top candidates all shared the same tendency of being very sparsly cut. This leads to the immediate interpretation that the no cut configuration is representating the lowest friction response that we can get within this regime. The ranking of the configurations did not strictly favor the least amount of atoms removed, but considering that the minimum values found for the Tetrahedron, Honeycomb and Random walk patterns were -0.062, -0.109 and -0.061 it is quite obvious that the model is not reliable in this domain. By asking for the lowest friction we essentially (optimize) uncertain parts of the configuration space which can leads to unphysical predictions of negative vaues. By somehow penalizing or ignoring negative we would simply disguise the problem but not solve it as the certainity nessecary to provide interesting results regarding low friction is not in place for this model. That would require a different training set altogether.

3.5.2 Genetic algorithm search

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

[15]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`.

[16]*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`.