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

Designs for a negative friction coefficient.

Mikkel Metzsch Jensen

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

Department of Physics
Faculty of mathematics and natural sciences

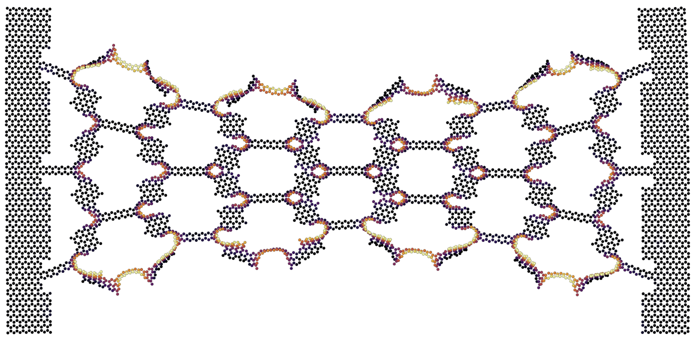
UNIVERSITY OF OSLO

Spring 2023

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

Designs for a negative friction coefficient.

Mikkel Metzsch Jensen



Abstract

Abstract.

Acknowledgments

Acknowledgments.

List of Symbols

F_N	Normal force (normal load)

Acronyms

MD	Molecular Dynamics. 1, 2, 3, 7
ML	Machine Learning. 2, 3
MSE	Mean Squared Error. 8
SGD	Stochastic gradient descent. 9

Contents

Chapter 1

Introduction

1.1 Motivation

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

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

On the other hand, the reduction of friction is not the only sensible application for tribological studies. Controlling frictional properties, besides minimization, might be of interest in the development of a grasping robot where finetuned object handling is required. While achieving a certain "constant" friction response is readily obtained through appropriate material choices 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 these lines are the gecko feet. More precisely, the Tokay gecko has received a lot of attention in scientific studies aiming to unravel the underlying mechanism of its "togglable" adhesion properties. Although geckos can produce large adhesive forces, they retain the ability to remove their feet from an attachment surface at will [5]. This makes the gecko able to achieve a high adhesion on the feet when climbing a vertical surface while lifting it for the next step remains relatively effortless. For a grasping robot, we might consider an analog frictional concept of a surface material that can change from slippery to rough on demand depending on specific tasks.

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

In materials science such MD-based numerical studies have been used to explore the concept of so-called *metamaterials* where material compositions are designed meticulously to enhance certain physical properties [6–11]. This is often achieved either by intertwining different material types or removing certain regions completely.

In recent papers by Hanakata et al. [6, 7] numerical studies have showcased that the mechanical properties of a graphene sheet, yield stress and yield strain, can be altered through the introduction of so-called *kirigami* inspired cuts into the sheet. Kirigami is a variation of origami where the paper is cut additionally to being folded. While these methods originate as an art form, aiming to produce various artistic objects, they have proven to be applicable in a wide range of fields such as optics, physics, biology, chemistry and engineering [12]. Various forms of stimuli enable direct 2D to 3D transformations through folding, bending, and twisting of microstructures. While original human designs have contributed to specific scientific applications in the past, the future of this field is highly driven by the question of how to generate new designs optimized for certain physical properties. However, the complexity of such systems and the associated design space make for seemingly intractable problems ruling out analytic solutions.

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

Hanakata et al. attribute the variety in yield properties to the non-linear effects arising from the out-of-plane buckling of the sheet. Since it is generally accepted that the surface roughness is of great importance for frictional properties it can be hypothesized that 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 the stretching of the sheet and frictional properties. If significant, this could give rise to a control of friction behavior beyond manufacturing. For instance, the grasping robot might apply such a material as artificial skin for which stretching or relaxing of the surface could result in a changeable friction strength; Slippery and smooth when interacting 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 coefficients which might in theory also take on negative values given the right response from stretching. The latter would constitute a rarely found 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. However, 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 coefficient by more than one order of magnitude. This supports the idea that kirigami designs can be used to alter friction, but we believe that taking this concept to the nanoscale regime would involve a different set of underlying mechanisms and thus contribute to new insight in this field.

1.2 Goals

In this thesis we investigate the possibility to alter 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

Chapter 2

Machine Learning

We will use machine learning as a tool for designing kirigami configurations that alter the friction behavior in different manners. Through MD simulations we will provide data for which the machine learning model can learn from. If successful we can utilize the model to predict the friction behavior of unseen configurations which makes us able to skip the MD and speed up the search process. It is not obvious that the model can capture the physical mechanism governing the friction process and thus a succeeding model might invite further and more advanced study within this domain.

We aim to implement a basic machine learning approach as an investigation whether this is useful.

2.1 Neural network

A neural network is of the foundational concepts in machine learning. It consists of an input $\mathbf{x}$ some so-called *hidden layers* and an output $\hat{\mathbf{y}}$. We reserve $\mathbf{y}$ for the actual true output that the model is going to estimate. The input vector $\mathbf{x} = x_0, x_1, \ldots, x_{n_x}$ contains the input *features*. Each input x_i is connected to each of the *nodes* in the first hidden layer. For a given note $a_j^{[1]}$ in the first hidden layer the input is processed as

$$a_j^{[1]} = f(\sum_i w_{ij} x_i + b_j^{[1]}),$$

where w_{ij} is the weight representing the connection strength from the input feature x_i to the (first) hidden layer node $a_j^{[1]}$, $b_j^{[1]}$ is a bias associated to the specific hidden layer node and $f(\cdot)$ is an *activation function*. Notice, that we have individual weights for each connection in the network but we ommit any notation for the layers it connects as it is obvious from the equations so far. The activation function is an crucial part of the neural network as it provides a non-linear mapping to the inputs to the node. Without this, the resulting framework is nothing more than a linear transformation. By introducing the activation function the model is capable of capturing complex trends SOURCE. The signal is carried through the hidden layers in a similar fashion from hidden layer $l-1$ to l as

$$a_j^{[l]} = f\left(\sum_i w_{ij} a_j^{[l-1]} + b_j^{[1]}\right).$$

Keep in mind that weight is specific to the connection between layer $l-1$ and l. Finally, the last hidden layer, $L-1$, connects densely to the output $\hat{\mathbf{y}}$ as for the remaining of the network. However, here the activation function can be omitted or exchanged with a different mapping, e.g. the sigmoid which maps the output to $(0,1)$. The whole process of sending data through the model is called *forward propagation* as it constitutes the process of sending information through the network. The specific values of the weights and biases will govern the final output.

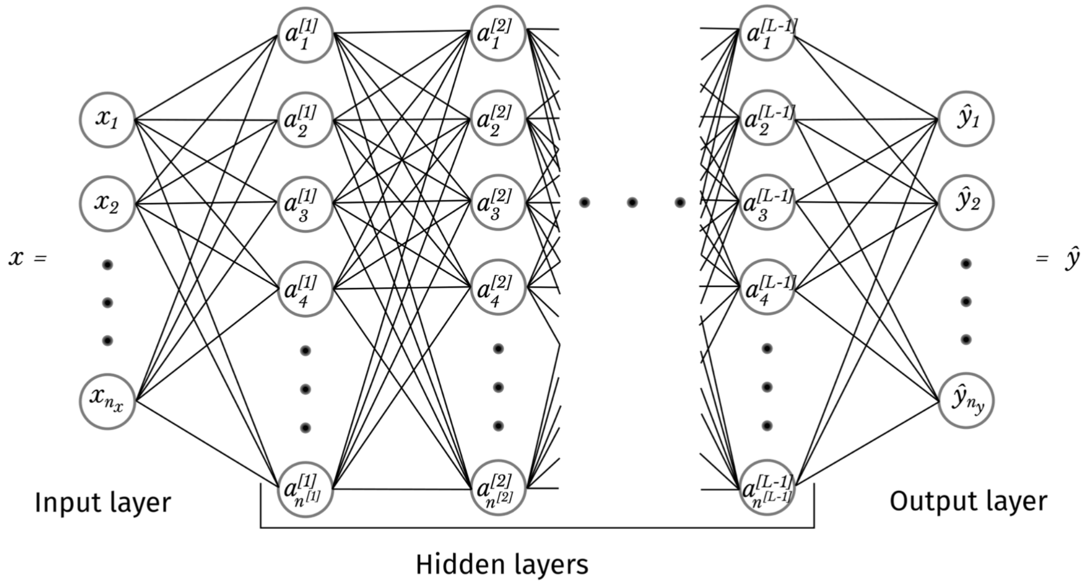


Figure 2.1: From overleaf IN400

The network is trained through the process of *backpropagation* and *gradient descent* optimization. Backpropagation starts with a definition of the error associated with the model output. A so-called *loss function* L compares the output $\hat{\mathbf{y}}$ with the expected result $\mathbf{y}$ and calculates the error otherwise known as the loss. For a continuous scalar output, we might simply use the mean squared error (MSE)

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

For classification, where we expect an output of True or False (1 or 0), a common choice is binary cross entropy (BSE)

$$L_{\text{BSE}} = -\sum_{i=1}^{n}\Big[y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\Big].$$

This arrises for information theory or something similar. Without going into details with the derivation we can convince ourselves that the error is minimized (0) for a perfect prediction and maximized for a completely wrong prediction. When $y_i = 1$ we essentially add the negative term $-\log(\hat{y}_i)$. With a prediction given as a number between 0 and 1 (through the sigmoid function), we can interpret the output as a probability of the given classification. When the output approaches 1, $\hat{y}_i \to 1$, in the correct case of $y_i = 1$ the loss contribution goes to towards zero. However, if the output approaches 0, $\hat{y}_i \to 1$, the loss contribution diverges $\hat{y}_i \to \text{inf}$. Similar apply for the case of $y_i = 0$ with opposite directions.

Given a loss function we can calculate the loss gradient with respect to each weight and bias in the model. That is, how is the error effected is we tweak a certain model parameter, weight or bias. The overall idea is then to “nudge” each parameter in the appropriate direction and repeat the process of forward and backpropagation. For a given paramter θ_t at time t, measured in epoch for instance (sets of forward and backward repetitions) we can update it through gradient descent

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta_t),$$

where the new parameter θ_t is given by stepping in the direction of the loss gradient $\nabla_\theta L(\theta_t)$ with a step length proportional to learning rate η. In practice, gradient descent is done in batches. That is, for a batch of training data, consisting of input data and true output data, a single step is taken for the average gradient. If a small batch is considered the path towards the optimal parameters will be governed by a fluctuation in loss space while a large batch will result in a more direct path towards lower error. This is the key concept behind stochastic gradient descent (SGD) which creates random batches from the training data and performs gradient descent. Even though a stable descent toward a minimum error sounds promising, the strength in SGD lies in the fluctuations of its descent. Without it, learning would be stuck at the first local minimum in the loss space.

2.2 Optimizers

The name *optimizers* covers a variety of gradient descent methods which can be utilized for model training. In our study we will use the ADAM optimizer (short?) which extends the concept of stochastic gradient descent. ADAM combinnes several tricks from other succesfull optimizers by including an exponetial moving average and momentum term.

We extend the gradient descent by the introduction of a momentum term m_t such that we get

$$\theta_{t+1} = \theta_t - m_{t+1}, \qquad m_{t+1} = \alpha m_t + \eta \nabla_\theta L(\theta_t) \tag{2.1}$$

with $m_0 = 0$. If we introduce the shorthand $g_t = \nabla_\theta L(\theta_t)$ we find

$$\begin{aligned}
m_0 &= 0 \\
m_1 &= \alpha m_0 + \eta g_0 = \eta g_0 \\
m_2 &= \alpha m_1 + \eta g_1 = \alpha^1 \eta g_0 + \eta g_1 \\
m_2 &= \alpha m_2 + \eta g_2 = \alpha^2 \eta g_0 + \alpha \eta g_1 + \eta g_2 \\
&\vdots \\
m_t &= \eta \left(\sum_{i=0}^{t-1} \alpha^{t-1-i} g_i \right).
\end{aligned}$$

Hence m_t is a weighted average of the gradients which an exponentially decreasing weight. This act as a memory of the previous gradients and aid in passing of local minema of plateaus in the loss space. A variation of momentum can be achieved with the introduction of the exponetial moving average (EMA)

$$\begin{aligned}
\mathrm{EMA}(g_0) &= 0 + (1-\alpha) g_0 \\
\mathrm{EMA}(g_1) &= \alpha \mathrm{EMA}(g_t) + (1-\alpha) g_1 \\
&\vdots \\
\mathrm{EMA}(g_t) &= \alpha \mathrm{EMA}(g_{t-1}) + (1-\alpha) g_t = \sum_{s=0}^{t} \alpha^{t-s} (1-\alpha) g_t,
\end{aligned}$$

which is quite similar to that of momentum, but here we have the explicit weighting by $(1-\alpha)$.

Another trick in the book is the root mean square propagation (RMSProp) which is motivated by the problem of passing long plateaus in the loss space. Since the size of the updates are proportional to the norm of the gradient

$$\theta_{t+1} = \theta_t - \eta g_t \implies ||\theta_{t+1} - \theta_t|| = \eta ||g_t||$$

we might get the idea of normalizing the gradient step by dividing with the norm $|||g_t|$. However, this does not immediately solve the problem of long plateaus as we need to consider multiple past gradients, and thus we use the EMA instead. When reentering a steep region again we need to quickly downscale the gradient steps. By using the squared norm $||g_t||^2$ for the EMA we make it more sensitive to outliers and achieve just that. This corresponds to the RMSProp update scheme

$$\theta_{t+1} = \theta_t - \eta \frac{g_t}{\sqrt{\mathrm{EMA}(||g_t||^2) + \epsilon}}, \tag{2.2}$$

where ϵ is simply a small number to avoid division by issues. ADAM merges the idea behind the root mean square propagation technique in Eq. (2.2) with a EMA momentum term m_t and second order RMSProp term v_t

$$m_t = \beta_1 m_{t_1} + (1-\beta_1) g_t,$$
$$v_t = \beta_2 v_{t_1} + (1-\beta_2) g_t^2,$$

These are initially set to zero and will be biased toward that. ADAM correct for this by applying a scaling term $(1-\beta_1^t)$ and $(1-\beta_2^t)$ respectively

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \qquad \hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.3}$$

(2.4)

2.2.1 Weight decay

By adding a l^2 norm of the parameters to the cost function we get a penalty for high parameters.

2.2.2 Batch normalization

2.2.3 Learning rate decay strategies

2.3 Convolutional neural network

2.4 Hypertuning strategies

2.5 Prediction explanation

Looking at feature maps and gradient maps.

2.6 Accelerated search using genetic algorithm

2.6.1 Markov-Chain Accelerated Genetic Algorithms

2.6.1.1 Talk about traditional method also?

2.6.1.2 Implementing for 1D chromosone (following article closely)

We have the binary population matrix $A(t)$ at time (generation) t consisting of N rows denoting chromosones and with L columns denoting the so-called locus (fixed position on a chromosome where a particular gene or genetic marker is located, wiki). We sort the matrix rowwise by the fitness of each chromosono evaluated by a fitness function f such that $f_i(t) \leq f_k(t)$ for $i \geq k$. We assume that there are a transistion probability between the current state $A(t)$ and the next state $A(t+1)$. We consider this transistion probability only to take into account mutation process (mutation only updating scheme). During each generation chromosones are sorted from most to least fitted. The chromosone at the i-th fitted place is assigned a row mutation probability $a_i(t)$ by some monotonic increasing function. This is taken to be

$$a_i(t) = \begin{cases} (i-1)/N', & i-1 < N' \\ 1, & \text{else} \end{cases}$$

for some limit N' (refer to first part of article talking about this). We use $N' = N/2$. We also defines the survival probability $s_i = 1 - a_i$. In thus wau a_i and s_i decide together whether to mutate to the other state (flip binary) or to remain in the current state. We use s_i as the statistical weight for the i-th chromosone given it a weight $w_i = s_i$.

Now the column mutation. For each locus j we define the count of 0's and 1's as $C_0(j)$ and $C_1(j)$ resepctively. These are normalized as

$$n_0(j,t) = \frac{C_0(j)}{C_0(j)+C_1(j)}, \quad n_1(j,t) = \frac{C_1(j)}{C_0(j)+C_1(j)}.$$

These are gathered into the vector $\mathbf{n}(j,t) = (n_0(j,t), n_1(j,t))$ which characterizes the state distribution of j-th locus. In order to direct the current population to a preferred state for locus j we look at the highest weight of row i for locus j taking the value 0 and 1 respectively.

$$\begin{aligned} C_0'(j) &= \max\{W_i | A_{ij} = 0;\ i = 1, \cdots, N\} \\ C_1'(j) &= \max\{W_i | A_{ij} = 1;\ i = 1, \cdots, N\} \end{aligned}$$

which is normalized again

$$n_0(j,t+1) = \frac{C_0'(j)}{C_0'(j)+C_1'(j)}, \quad n_1(j,t+1) = \frac{C_1'(j)}{C_0'(j)+C_1'(j)}.$$

The vector $\mathbf{n}(j,t+1) = (n_0(j,t+1), n_1(j,t+1))$ then provides a direction for the population to evolve against. This characterizes the target state distribution of the locus j among all the chromosones in the next generation. We have

$$\begin{bmatrix} n_0(j,t+1) \\ n_1(j,t+1) \end{bmatrix} = \begin{bmatrix} P_{00}(j,t) & P_{10}(j,t) \\ P_{01}(j,t) & P_{11}(j,t) \end{bmatrix} \begin{bmatrix} n_0(j,t) \\ n_1(j,t) \end{bmatrix}$$

Since the probability must sum to one for the rows in the P-matrix we have

$$P_{00}(j,t) = 1 - P_{01}(j,t), \quad P_{11}(j,t) = 1 - P_{10}(j,t)$$

These conditions allow us to solve for the transition probability $P_{10}(j,t)$ in terms of the single variable $P_{00}j,t$.

$$\begin{aligned} P_{10}(j,t) &= \frac{n_0(j,t+1) - P_{00}(j,t)n_0(j,t)}{n_1(j,t)} \\ P_{01}(j,t) &= 1 - P_{00}(j,t) \\ P_{11}(j,t) &= 1 - P_{10}(j,t) \end{aligned}$$

We just need to know $P_{00}(j,t)$. We start from $P_{00}(j,t=0) = 0.5$ and then choose $P_{00}(j,t) = n_0(j,t)$

2.6.1.3 Repair function

Talk about it here or in random walk section?

Part II

Simulations

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).