Computational analysis of twist defect motion and nucleosome repositioning induced by remodelers

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author :	Davey Plugers
Student ID :	S2946661
Supervisor :	Prof. dr. ir. S.J.T. Van Noort
2^{nd} corrector :	Prof dr. H. Schiessel

Leiden, The Netherlands, October 10, 2022

Computational analysis of twist defect motion and nucleosome repositioning induced by remodelers

Davey Plugers

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

October 10, 2022

Abstract

The repositioning of nucleosomes in DNA plays a vital role in accessing the encoded information. This repositioning can be induced through remodelers sitting on the DNA wrapped around the histone cores. These introduce twist defects which can propagate through the nucleosomal DNA and escaping on one end which leads to single base pair repositioning of the nucleosome. However the precise dynamics at play here and how it interacts with the larger DNA sequence are not fully clear. Here we will demonstrate some of the properties of the twist defects on the wrapped DNA and how this affects the nucleosome repositioning along a sequence. This is done through setting up a Monte-Carlo Markov chain for the rigid base pair model. We find that a build-up of twists occurs on hard to traverse DNA regions and that this causes multiple steps in quick succession. We also see that the positional preference of the nucleosome with remodeler is less sensitive to the global landscape as opposed to free nucleosome repositioning. This quick succession of steps may explain an earlier work by Sabantsev et al. [1] where this effect was seen in an experiment without an explanation. The current work can be used as guidance for future studies investigating the dynamics of simultaneous defects around the nucleosome.

Contents

1 Introduction **1**

2 Theoretical background **3**
2.1 The Rigid Base Pair Model (RBP) 4
2.2 Nucleosomal DNA and defect geometry 5
2.3 Defect movement, energies and displacement damping 8
2.4 Expected defect lifetime and duplet/triplet states 11
2.4.1 Single particle lifetime 11
2.4.2 Double Particle lifetimes 13
2.4.3 Duplet and triplet defect states 23
2.5 Gillespie Markov Chain algorithm 24

3 Simulation **26**
3.1 Dependencies and functions 26
3.1.1 Global code structure 26
3.1.2 Full DNA Energy Landscape 27
3.1.3 Nucleosome Transition Function 29
3.1.4 Supporting functions 30
3.2 Setup Nucleosome Transition simulation 31
3.3 Setup final simulation 32
3.3.1 Code Explanation and goal 32
3.3.2 Parameter explanation 32

4 Results **35**
4.1 Defect energy landscapes 35
4.2 Defect lifetimes 41
4.3 Nucleosome occupancy 45

Version of October 10, 2022– Created October 10, 2022 - 09:08

4.4 Duplet/Triplet state ejection 50

5 Discussion **54**
5.1 Future development of the code 54
5.2 Semi-Analytical lifetime model 57

6 Conclusion **59**

A Appendix **64**
A.1 Validity testing 64
A.1.1 base pair probabilities 64
A.1.2 Defect geometry 65
A.1.3 Energy plots 66
A.1.4 Occupancy tests 69
A.2 Additional figures and plots 70
A.2.1 Occupancy plots 70
A.2.2 Other lifetime plots 72
A.3 Sequences used 73

Chapter 1

Introduction

DNA plays a vital role in who we are, from creation of life to the synthesis of proteins. To do this, the DNA contains a lot of information inscribed in its bases: adenine (A), cytosine (C), guanine (G) and thymine (T). This DNA then has to be neatly packaged into a more ordered structure. This can be done by wrapping 147 base pairs of DNA around 8 histone proteins. These histone proteins form the cylindrical core with 14 binding sites for the DNA and create a nucleosome with an 11 nm diameter [2]. The nucleosomes can then be stacked together forming fractal shapes which create our chromosomes to package 2 meters of DNA into a $10\mu m$ nucleus [3]. However as the DNA gets packaged it becomes impossible to access the inscribed information. To still transcribe the information in the DNA the nucleosomes have to move around to create free regions [4]. One such mechanism is through single base pair defects which propagate around the nucleosome [5–7]. This creates a caterpillar like motion and allows the nucleosome to slowly move. We will be investigating these defects, their propagation around the nucleosome and how they affect the positioning on a given sequence.
There has been a lot of works investigating the properties of nucleosomes. There have been many experiments to understand the structure of the histones (nucleosome core) and the wrapping of the DNA around them [2]. One such method to analyse the structure is by using X-ray crystallography [8, 9]. These results are able to give us a better understanding of the interactions at play. They show the DNA also binds every 10 steps to the histone core in binding sites. With these insights into the structure, the wrapped DNA can be modelled in a simple way. The Rigid base pair

Version of October 10, 2022– Created October 10, 2022 - 09:08

model (RBP) gives a coarse-grained description of the base pair steps as simple plates. This model then only looks at the position, orientation and base pairs involved to create an estimate of the energy costs [10, 11]. This model is also able to estimate the cost of having an extra or missing base pair between two binding sites. These are called twist defects due to the DNA having to over- and undertwist when there is a missing or extra base pair. These twists defects are able to propagate and make the nucleosome slide along the sequence [5]. Molecular dynamics simulations have also been used to verify the stability of these defects and the sliding of the nucleosome [7]. While there have been many novel insights and innovations so far, there are still plenty of questions left. A recent paper by Sabantsev et al. [1] has analysed both nucleosome ends simultaneously using FRET while a remodeler was present. These remodelers are able to use ATP to introduce defects onto the nucleosome and induce nucleosome sliding [12]. They observed a build-up of defects on one end of the remodeler. It was assumed that this build-up was to ensure nucleosomal stability during remodeling. However, the exact purpose of it is not yet clear.
We will be using a simplified version of the RBP model to calculate energy costs of these twist defects. Using these energy costs we will be able to construct an energy landscape in which these defects may propagate through using thermal fluctuations. An initial simulation will be made to calculate the lifetime of the defects which we introduce with a remodeler. We look how these lifetimes change and the delay between defects leaving the system. After this we create a second simulation which will allow the nucleosome itself to position along a sequence. The goal for this is to analyse if the strong positioning preference is still present during remodeling. Finally, we will be using both simulations to analyse and try to find the presence of multiple defects present simultaneously.
All code together with plots and brief usage explanation will be made available on the GitHub page DaveyPlugers/DnaTwistDefect.

Version of October 10, 2022– Created October 10, 2022 - 09:08

Chapter 2

Theoretical background

Modelling the DNA wrapped around a nucleosome can be done through different methods. Molecular dynamic simulations provide excellent accuracy but require too much computing power for long durations [13]. More simple models can therefore be introduced to approach the dynamics of the system. One such model is the rigid base pair model (RBP) [10, 14] which will be used in this project. Using this model it is possible to define a corresponding energy cost for a certain geometric configuration of the DNA. One could therefore introduce defects by changing the underlying geometry of the DNA and calculate the energy cost.

We used this energy model to reproduce the base pair sequence dependency as was done [11, 14] and observed [15] in earlier works and to create the energy landscapes of having twists present at different locations. We used these energy values to set up a dynamic Monte Carlo simulation by using the Gillespie algorithm [16]. Doing this allowed us to evolve the system when defects are present and calculate the lifetime of the defect in the nucleosome.

2.1 The Rigid Base Pair Model (RBP)

It is possible to model the DNA by using a coarse grained description. This coarse graining is done by treating each individual base pair as a plate and defining the geometry of a base pair step. This geometry consists of three translational and three rotational degrees of freedom. These then determine the position of the following base pair with respect to the previous one (see Section 2.2). One could write down these six parameters of the step into a vector for convenience.

$$\vec{V} = (V_1, ..., V_6) = (x_{shift}, x_{slide}, x_{rise}, \theta_{tilt}, \theta_{roll}, \theta_{twist}) \tag{2.1}$$

One can then also find the intrinsic geometric values of the base pair step. This is the displacement that a base pair step would have if it was free to move and not subject to any force or displacement. This rest position of the base pair step is uniquely determined by the base pairs involved:

$$\vec{V_0} = (\bar{x}_{shift}, \bar{x}_{slide}, \bar{x}_{rise}, \bar{\theta}_{tilt}, \bar{\theta}_{roll}, \bar{\theta}_{twist}) \tag{2.2}$$

So an AA step would have a different rest position when compared to an AG step. This creates 16 intrinsic rest positions for all 4^2 combinations possible, however this gets reduced to 10 unique values due to a symmetry argument for the other side of the base pair step [10] (AA step is TT along the other side).

By using the experimentally determined stiffness of the base pair steps [10] it becomes possible to calculate the energy cost of the geometry. This stiffness is, just like the intrinsic geometry, dependent on the base pairs involved in the step. The stiffness is given by a matrix which also links the different deformations cross-terms together:

$$E = \frac{1}{2}(\vec{V} - \vec{V}_{ij,0})^T \mathbf{K}_{ij} (\vec{V} - \vec{V}_{ij,0}) \qquad i, j \in [A, T, C, G] \tag{2.3}$$

We simplify our calculation to only take into account the quadratic terms for: rise, tilt, roll and twist. Cross-terms are neglected and the shear modes (shift and slide) are not considered here.

We also need to get an estimate for the energy cost of unbinding a binding site on the nucleosome. This binding site is modelled by creating a smooth potential in which the binding site can sit. We will be using 12 kT for all sites as our potential depth as given in [11]. There are however some other

papers with a similar energy model which give different depths depending on the position [17]. These potential depths are still around 12 kT and to simplify our model we continue to use the same depth for all binding sites.

2.2 Nucleosomal DNA and defect geometry

Wrapping the DNA around the histone octamer to form a nucleosome is a process that has been studied a lot. There are many papers analysing the structure through X-ray crystallography [8, 9], electron microscopy [18, 19] or molecular dynamics simulations, see e.g. [6, 7]. They show how base pair steps rotate around with a 10 base pair periodicity around the nucleosome. The minor groove of the DNA is then located in such a way that it is connected to the binding site on the histone.

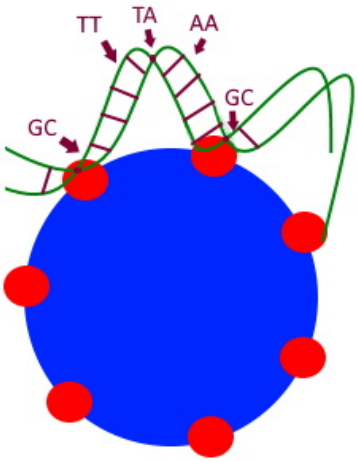

Figure 2.1: *Representation of DNA wrapping around the nucleosome. The minor groove is on the binding sites (red) and there are base pair steps indicated with their preferred position. The DNA fully twists around between binding sites and repeats this 14 times. In total the DNA is wrapped 1.75 times around the nucleosome.*

The DNA itself also has preferred positions for the base pair steps. This preference in position is due to the base pair steps having their intrinsic bending values which causes them to prefer high or low bending positions. This sequence dependence is very important because it causes the nucleosome position to have a preferred position as well. Nucleosome locations on DNA are often found in steps of 10. If it would take a smaller

step, the positional preference of the sequence would be disturbed which gives rise to higher energy costs. This is not just an artifact of the RBP model but an actual observation seen in experiments [20]. In Section 2.3 we will discuss this change in energy for different steps further.

The locations between the binding sites are called Super Helical locations (SHL). Since there are 14 binding sites there are 13 SHL which range from -6 to 6. The binding sites themselves are often referred to as half-integers of SHL's. For example -5.5 would be the binding site between -6 and -5. Later we will also define the position of twist defects through the SHL. Here an integer location would correspond to a defect constrained to a single SHL and a half-integer to a defect in two SHL's.

The geometry of the base pair steps is given by the six aforementioned parameters. These either translate or rotate around a given axis. By modelling the base pair as a rectangle and defining the values for every step we are able to generate the DNA in the shape we want.

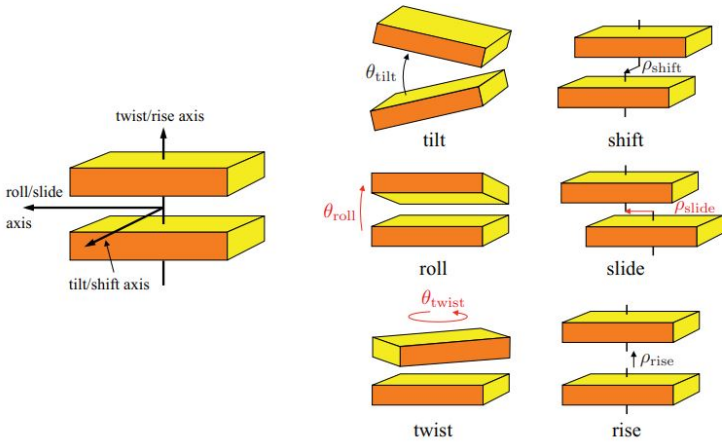

Figure 2.2: *Geometric parameters used to determine the position and rotation of the next base pair step. (Image courtesy of H. Schiessel [11])*

To calculate the energy values in our model we need to have a geometry associated with every step. This means we have to recreate the structure of the DNA around a nucleosome. We follow the derivation in chapter 4 of [11] and take a constant rise but no shifting or sliding of the base pair steps:

$$[x_{shift}, x_{slide}, x_{rise}] = [0, 0, 3.4\text{Å}] \tag{2.4}$$

Then we will need to determine the rotations of the base pair steps. There are a few effects that need to be captured to obtain the correct geometry. First off, DNA has an iconic double helix shape. This means we need to add a twist value so we do not just have a stack of parallel rectangles as our DNA. Next we want to be able to bend our DNA into a circle. This can either be done by sudden kinks or by smoothly rolling. However the latter is more energetically favourable due to quadratic energy dependence as long as the bending is not too strong. This smooth rolling is done by periodically changing the tilt and roll parameter.

$$[\theta^i_{tilt}, \theta^i_{roll}, \theta^i_{twist}] = \left[\Gamma\, sin\left(\frac{2\pi i}{10} - \phi\right), \Gamma\, cos\left(\frac{2\pi i}{10} - \phi\right), \theta_{twist}\right] \tag{2.5}$$

We use i to represent the index of our step and multiply by a factor of $2\pi/10$ to get a ten step periodicity. Here ϕ is an arbitrary phase factor, in literature it is taken to be $147\pi/10$ to center a maximum roll in the middle of the sequence. However we used it to set the binding sites on positions that are a multiple of 10. Γ is a parameter that determines the bending of our base pair step. The value can be determined through the radius of the histone octamer and the height of the base pair step. Finally there is θ_{twist}, which is the rotation of our DNA to form a double helix. Normally we would take $\theta_{twist} = 36^\circ$ since we have a 10 step periodicity for our DNA rotation, however we want to wrap the DNA in a non-planar way around the nucleosome. Because of this the value is taken to be slightly lower to form a superhelical structure. From [21] we then get $\theta_{twist} = 35.575^\circ$ and $\Gamma = 4.46^\circ$. This geometry then gives a basic structure on which we can define defects and calculate the ground state energy level.

We then defined the defect geometries. There are two types of defects and two different sizes possible. The two types are overtwist and undertwist, which make the base pair step twist more or less than the ground state respectively. The different sizes are defined by how many SHL are taken up by the defect. We can have a defect constrained to a single SHL or two SHL's when a binding site opens. We will refer to these as K10 and K20 defects respectively like done in [5]. To make a K10 overtwist defect, we replace the 10 base pair steps in between with only 9 base pair steps. Doing this also causes all of the base pairs to shift position which we discuss in depth in section 2.3. In a similar fashion we can define an undertwist by replacing the 10 base pair steps with 11 steps instead. The same idea can be done for a K20 defect but now going from 20 to 19 and 20 to 21 base pair steps.

Version of October 10, 2022– Created October 10, 2022 - 09:08

To define these geometries we need to multiply all the steps by an adjustment factor and make a correction to the index i. We will write this out for a K10 undertwist, the main defect used in this project:

$$\begin{aligned}
[x_{shift}, x_{slide}, x_{rise}] &= \frac{10}{11} * [0, 0, 3.4\text{Å}] \approx [0, 0, 3.09\text{Å}] \\
[\theta^{i'}_{tilt}, \theta^{i'}_{roll}, \theta^{i'}_{twist}] &= \frac{10}{11} * \left[\Gamma\ sin\left(\frac{10}{11}\frac{2\pi(i'-\Xi)}{10} - \phi\right), ... \right. \\
&\quad \left. ..\Gamma\ cos\left(\frac{10}{11}\frac{2\pi(i'-\Xi)}{10} - \phi\right), \theta_{twist}\right] \\
\Xi &= BeginIndexDefect - 1
\end{aligned} \tag{2.6}$$

We have multiplied all values with an adjustment factor in front since we are taking one extra step now. We also need to adjust the index however to make sure that our index now corresponds with positions in between our previous steps. This is done by subtracting Ξ from the index which moves the index in such a way that we ignore earlier steps. Doing this it becomes trivial to just multiply our index by a factor of $10/11$ to make sure these 11 steps are the indices between the original 10 steps.

One can then simply repeat this process for an overtwist by replacing all factors of 11 by 9. Doing the same process for a K20 defect replaces 11 by 21 or 9 by 19 and all factors of 10 by 20 except for the denominator for our tilt and roll value. This is because this 10 corresponds to our original 10 step periodicity factor $2\pi/10$.

2.3 Defect movement, energies and displacement damping

Now that this defect geometry is defined it becomes possible to define this with our ground state geometry and introduce a defect into our system. However we need to be careful when combining these since it is not possible to just introduce a new base pair in the middle of a sequence. Instead we have to take it from one of the two sides. One possibility is that we have a defect being introduced at one edge of the nucleosome. Doing this would take a free base pair and shift it into the nucleosomal geometry. This defect is then able to move back out at the original end it was introduced or move all the way through to the other side of the nucleosome. If the defect leaves on the end it was introduced it undoes the earlier step it made. However if it is successful to go all the way through the DNA

Version of October 10, 2022– Created October 10, 2022 - 09:08

around the nucleosome made a single base pair step. This is either a step towards the introduction site if it was an undertwist or to the exit site if vice versa.

We could however also try to do this in the middle of the nucleosome. We would always have to introduce an undertwist and overtwist together creating a pair. If these were to *move into each other* in the same SHL the defects annihilate and the ground state is recovered. However when using a remodeler it becomes possible to introduce these pairs while having an impermeable boundary stopping them from meeting each other. This allows the defects to both leave and the system will make a step towards the site on which the overtwist left.

An important aspect of these defects moving through our nucleosome is that they displace the base pairs out of their preferred position. As the defect leaves at the other end it will cause the entire sequence around the nucleosome to be shifted by a single index. Due to this the energy can be higher or lower depending on if it is moving towards or away from a preferred position. This effect can be seen on a smaller scale too, for example when plotting the energy costs of having a defect in a certain position. The energy landscape can rise up or fall down as our defect approaches one end. This is because it is slowly but surely shifting the entire sequence out of position.

This effect is only visible however for sequences that are created using positional dependent probabilities for the base pairs. When we try to repeat this process for a sequence with equal probabilities for all positions we do not observe any rise or fall in energy. This is due to in the former example we will move base pairs away or towards their lowest energy position simultaneously. While in the latter base pairs are spread at random, this causes on average no net change in the energy due to this shift.

To actually generate this energy landscape we need to define how to get the defect energy cost. We do this by summing over the energy for all the base pair steps and computing a total energy. We do this for our ground state and the defect geometry. We can also add the cost of unbinding if we were to use a K20 defect. We then simply subtract the energy difference to get an estimate for the defect cost.

$$Energy_{Geom} = \sum_{n=1}^{N=146} E_n = \sum_{n=1}^{N=146} \frac{1}{2}(\vec{V}_n - \vec{V}_{ij(n),0})^T \mathbf{K}_{ij(n)} (\vec{V}_n - \vec{V}_{ij(n),0}) \quad (2.7)$$

Version of October 10, 2022– Created October 10, 2022 - 09:08

Here we let $\vec{V}_{ij(n),0}$ be the intrinsic geometry for the base pairs involved in the base pair step on position n. We let the vector $\vec{V}_n$ denote the geometry that we have given to our system. This can either be undefected or with a defect somewhere in the system. We can simply insert a defected ($E_{Defected}$) and an undefected (E_{Ground}) geometry and take the difference to get an estimate for the defect energy costs.

$$Energy = Energy_{Defected} - Energy_{Ground} \; (\; + \; Energy_{Unbind}) \tag{2.8}$$

The energy that we get from taking this difference consists of two contributions. The energy cost of having a defect instead of a more relaxed system and the energy cost to shift the sequence. While it is not possible to perfectly split these contributions, since in the defect position we are already shifting, it is possible to split the majority of it. This can be done by splitting the energies for different indices. For example doing this for an 11 base pair undertwist between position 59 and 69 we would get:

$$E_{Geom} = \sum_{n=1}^{N=146} E_n = \left(\sum_{n=1}^{N=58} E_n + \sum_{n=70}^{N=146} E_n \right) + \sum_{n=59}^{N=69} E_n = E_{Shift} + E_{Defect} \tag{2.9}$$

The reason it becomes important to split these energies is because of a big issue with the RBP model. The energy values are exaggerated due to the system not being able to relax internally. This causes the shifting energy cost in our model to be much higher than what is observed in molecular dynamics simulations [6] or less constrained RBP models [5] which has a repositioning barrier up to 20 kT as opposed to our 90 kT. As a solution we will introduce a damp factor which will lower the energy values of the shifting while leaving the defect energy undisturbed. The reason we choose not to multiply the defect energy with this same value is due to $Energy_{Unbind}$.

We want the energy cost to release a binding site to be larger than but still close to the energy difference between a K20 and a K10 defect. The reason is that defects are semi-stable between SHL's however they can transfer between with small probability. Once a defect has been introduced by a remodeler it goes through the system according to the Arrhenius equation ($k \propto e^{\frac{-\Delta E}{k_B T}}$) [22]. These energy barriers which the defect has to overcome need to be small enough, such that it happens in a reasonable time frame. We use the unbinding costs of 12 kT as we discussed before. If we do not adjust the defect energies we find that this gives energy barriers between K10 states around 1 to 3 kT which is what we would hope to see.

Version of October 10, 2022– Created October 10, 2022 - 09:08

The exact value of the damp factor is a subject for discussion by itself. Lowering the shifting energy causes the energy difference between preferred and disturbed sequences positions to become small. If this becomes too small we will start to observe the sequence outside of the preferred position too often. It also causes the energy landscape for the defects to not rise and fall as much, while it can bring us closer to reality, if this is overcorrected we would no longer be able to observe any effects in our model which might be present in real life. Because of this it will happen that we make plots for different damp factors to see how they compare:

$$E_{Geom} = \alpha * E_{Shift} + E_{Defect} \tag{2.10}$$

2.4 Expected defect lifetime and duplet/triplet states

Keeping track of the defects and their motion is a very important aspect of this project. One thing we look at in detail is the time spent inside of the nucleosome for a defect. We simply look at when it was introduced and when it left the system. However we would also like to have an analytic expression to compare this with. Writing one out would however be overly complicated and unenlightening due to all the different transition rates in the system. Instead we will try to write out a formula that gives us the expected lifetime through recursion.

2.4.1 Single particle lifetime

We start of by showing a naive calculation of how we could get the expected duration for the defect to take a step to the right. We start at an arbitrary position denoted by n and model our system letting it only take steps to the right (+) or steps to the left (-). We can go all the way to the left to position 1 which is at the remodeler where the defect cannot pass through. We can also go all the way to the end position τ where if the defect takes a step to the right it escapes the nucleosome. These steps are given by the rates $k_n^{\pm}$ and their expected lifetimes are $(k_n^{\pm})^{-1}$ where we will use n as the positional index. In any position n we are able to step to

Version of October 10, 2022– Created October 10, 2022 - 09:08

the right which finishes our process or take a step to the left.

$$E[\text{Right (n)}] = \frac{k_n^+}{k_n^+ + k_n^-}\frac{1}{k_n^+ + k_n^-} + ...$$
$$\frac{k_n^-}{k_n^+ + k_n^-}\sum_i \left[Pr(\text{path i}) * \left(\frac{1}{k_n^+ + k_n^-} + E[\text{path i}] \right) \right]$$

Here the first term consists of the probability a step is taken to the right, multiplied by the expected time of a step. The expected time of the step is always given by the inverse of the sum of all rates. This step to the right terminates the path and we have the lifetime of this contribution. The second term is a step to the left which needs to be continued until it takes a step to the right in position n. For this we have our initial probability to go left and then the probability of each path. Then it sums the initial step duration with the path duration to get each contribution. Writing out all possible paths explicitly would be a very tedious process akin to determining the airspeed velocity of an unladen swallow. We can also not just neglect paths over a certain length since we do not have an estimate for the convergence rate.

A solution for this would be to use recursion by using the previous expected step duration. Then one could just have a binomial expression of the left and right steps:

$$E[\text{Right (n)}] = \frac{k_n^+}{k_n^+ + k_n^-}\frac{1}{k_n^+ + k_n^-} + ...$$
$$\frac{k_n^-}{k_n^+ + k_n^-}\sum_{i=0}^{\infty} \left[\frac{k_n^+}{k_n^+ + k_n^-} \left(\frac{k_n^-}{k_n^+ + k_n^-} \right)^i * \left(\frac{2+i}{k_n^+ + k_n^-} + (1+i) * E[\text{right (n-1)}] \right) \right]$$

Here i is the amount of times our system goes back to the left, it can be slightly rewritten to take into account the first step to the left as well.

$$E[\text{Right (n)}] = \frac{k_n^+}{k_n^+ + k_n^-}\frac{1}{k_n^+ + k_n^-} + ...$$
$$\sum_{i=1}^{\infty} \left[\frac{k_n^+}{k_n^+ + k_n^-} \left(\frac{k_n^-}{k_n^+ + k_n^-} \right)^i * \left(\frac{1+i}{k_n^+ + k_n^-} + i * E[\text{right (n-1)}] \right) \right]$$

This expression still looks complicated but convergence is guaranteed due to the power being on a term between 0 and 1. Using the following sum

Version of October 10, 2022– Created October 10, 2022 - 09:08

identities it becomes easy to write the final expression.

$$\begin{gathered}\sum_{r=1}^{\infty}\alpha^r=\frac{\alpha}{1-\alpha}\,,\ \sum_{r=1}^{\infty}r\alpha^r=\frac{\alpha}{(1-\alpha)^2}\ \text{for}\ |\alpha|<1\\ E[\text{Right (n)}]=\frac{1}{k_n^+ + k_n^-}+\frac{k_n^-}{k_n^+}\left(\frac{1}{k_n^+ + k_n^-}+E[\text{right (n-1)}]\right)\end{gathered}\tag{2.11}$$

Since this is a recursive method we would have to define the value for $E[\text{right (1)}]$. This would be the time to take a step to the right when the defect is right next to the remodeler. The expected duration is given by $1/k_1^+$ since k_1^+ is the rate of the only step we allow at this position.
This approach seems to work fine for a simple one particle system but things can quickly get more complicated if we try to introduce two particles. The solution to this issue is repeating what we just did but in a much more general way. We can define a few general paths which can be calculated recursively to get back to our original position n. On top of this we can completely skip over the summation we just did by just inserting $E[\text{right (n)}]$ after taking a step back.

$$\begin{gathered}E[\text{Right (n)}]=\frac{k_n^+}{k_n^+ + k_n^-}\frac{1}{k_n^+ + k_n^-}+\ldots\\ \frac{k_n^-}{k_n^+ + k_n^-}\left(\frac{1}{k_n^+ + k_n^-}+E[\text{Right (n-1)}]+E[\text{Right (n)}]\right)\end{gathered}$$

Rewriting this expression then gives:

$$\begin{gathered}E[\text{Right (n)}]=\frac{k_n^+ + k_n^-}{k_n^+}\left[\frac{k_n^+}{k_n^+ + k_n^-}\frac{1}{k_n^+ + k_n^-}+\ldots\right.\\ \left.\frac{k_n^-}{k_n^+ + k_n^-}\left(\frac{1}{k_n^+ + k_n^-}+E[\text{Right (n-1)}]\right)\right]\end{gathered}\tag{2.12}$$

Comparing this with equation 2.11 we see they give the exact same result. This recursive inserting of paths will now be used to get an expression for two-particle system lifetimes.

2.4.2 Double Particle lifetimes

An interesting concept is the appearance of multiple defects in the system. While a single defect mostly moves freely, having two defects present will limit the possible movement. This is due to the extremely high energy cost for if we were to have two defects on the same SHL. Having multiple of

Version of October 10, 2022– Created October 10, 2022 - 09:08

these defects present then obviously also changes the lifetime of a defect in the system. However at the same time it makes the analytical calculation of the lifetime extremely complicated. In this section we will be talking at our attempt at writing an approximation to the analytical solution for two particle systems.

When exploring double particle states it becomes important to remember what particle we are trying to follow. We will use black to denote the particle and its rate while we use red for a secondary particle to the left or blue for a secondary particle to the right. Take not that the rates of these defects are different since they will be pushing different base pairs. Due to this we have to be careful not to get them mixed. Having either a secondary particle in front or behind gives us two distinct states that we need to analyse. This analysis will consist of the following problems:

- Which steps can our system take in an attempt to make our particle step to the right?
- What are their probabilities and duration?
- Are there simplified recursive states which we can use to simplify the steps in our system?
- What is the chance there are two particle presents?
- What is the chance they are next to each other?

We will leave the fourth problem for the end of this section since its result requires an expansion of the model. For now we start by discussing a system with our primary defect in position n and a secondary defect to the left of it as shown in figure 2.3.

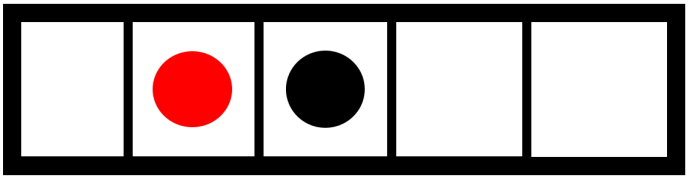

Figure 2.3: *Primary defect (black) and secondary defect (red) being represented by particles in a grid. Note that the grid on which the defects move is larger than the 5 boxes that are shown here.*

We now need to calculate the expected time to take a step to the right for

this two particle system. We denote this expected time as $E(n-1,n)$. Remembering that defects cannot move into each other, we will have two options for how this system may now evolve. Either the primary defect takes a step to the right, which finishes our process, or the secondary defect takes a step to the left.

$$E(n-1,n) = \sum_i Pr(Path_i) * E[Path_i] = \frac{k_n^+}{k_{n-1}^- + k_n^+}\frac{1}{k_{n-1}^- + k_n^+} + \frac{k_{n-1}^-}{k_{n-1}^- + k_n^+}\left[\frac{1}{k_{n-1}^- + k_n^+} + E(n-2,n)\right] \tag{2.13}$$

By writing out the probabilities of the paths and multiplying by their expected duration we can get an estimate for the primary defect to reach position n+1. However right now this process is incomplete because it is not yet clear what the duration is for $E(n-2,n)$. Once again we will write out all the different paths, get their probability and durations and try to reduce to a simplified recursive model.

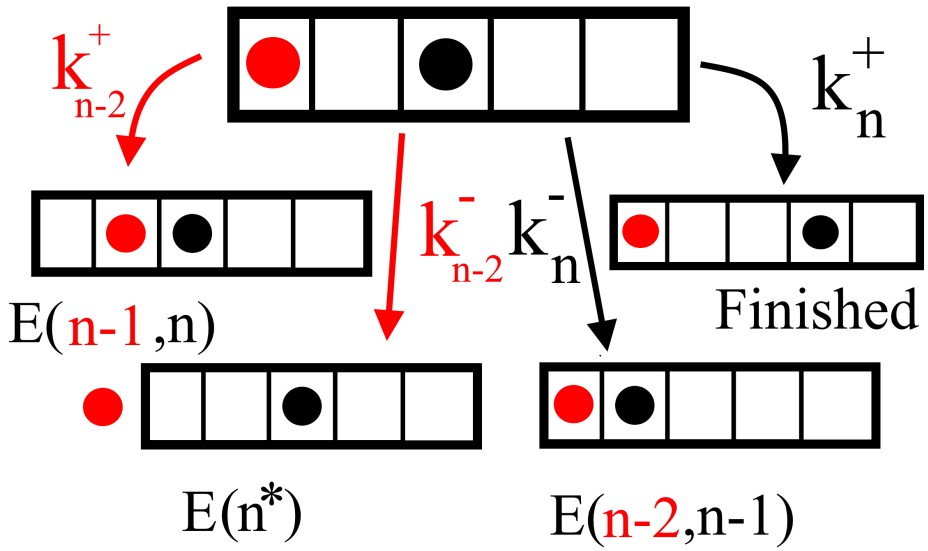

Figure 2.4: *All movement options of $E(n-2,n)$, the rates of these processes and the following expected lifetime duration underneath. For these following expected durations, we recover one step that finishes the process, two steps that give us a recursive model and a situation where our particles are split far from each other which we denote with $E(n^*)$.*

Version of October 10, 2022– Created October 10, 2022 - 09:08

Starting from the left and moving counter clockwise in figure 2.4, we start out by recovering our initial position. We already saw in section 2.4.1 that we can simply reinsert this expected value and take this term to the other side of the equality. For the next term we find that there is now a large separation between the particles. There are two options to continue this step, either we repeat the process we just did for $E(n-2,n)$ by writing out all the new paths, or we treat it as a single particle step. We will discuss both of these options in a moment once we finish writing out $E(n-2,n)$. Next up we find the same pattern we started with but shifted one to the left. However since we plan to recursively calculate all of these values we should already know what this is. We do need to define an initial position duration which would be given by $E(1,2) = 1/k_2^+$ (where 1 denotes the position against the remodeler) since this is the only movement possible in this position. Finally we recover another step that finishes our process by letting the primary defect take a step to the right. We now fill in these paths and rewrite it using the earlier probability and duration of the k_{n-1}^- path to recover the familiar Pr(path)*E[path] order.

$$
\begin{aligned}
E(n-1,n) = {} & \frac{k_n^+}{k_{n-1}^- + k_n^+}\frac{1}{k_{n-1}^- + k_n^+} + \\
& \frac{k_{n-1}^-}{k_{n-1}^- + k_n^+}\frac{k_{n-2}^+}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+}\left[\frac{1}{k_{n-1}^- + k_n^+} + \frac{1}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+} + E(n-1,n)\right] + \\
& \frac{k_{n-1}^-}{k_{n-1}^- + k_n^+}\frac{k_{n-2}^-}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+}\left[\frac{1}{k_{n-1}^- + k_n^+} + \frac{1}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+} + E(n^*)\right] + \\
& \frac{k_{n-1}^-}{k_{n-1}^- + k_n^+}\frac{k_n^-}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+}\left[\frac{1}{k_{n-1}^- + k_n^+} + \frac{1}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+} + E(n-2,n-1) + E(n^*)\right] + \\
& \frac{k_{n-1}^-}{k_{n-1}^- + k_n^+}\frac{k_n^+}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+}\left[\frac{1}{k_{n-1}^- + k_n^+} + \frac{1}{k_{n-2}^- + k_{n-2}^+ + k_n^- + k_n^+}\right] +
\end{aligned}
\tag{2.14}
$$

We have now rewritten our problem into five different paths, wrote out their probability and gave an estimate for their durations. Note however that for the fourth path (where we go to $E(n-2,n-1)$) we have added an extra term $E(n^*)$. This is done because once the primary particle in state n-1 takes a step to the right it is still in position n and we are not yet finished moving.

We are almost finished discussing the lifetime of our double particle state however we still need to explain this $E(n^*)$ term. The idea of this term is that it is just a single defect lifetime in position n, but we take into account that there is another defect present to the left. We do this by adding a prob-

ability that moving to the left will fail. This probability (denoted by $\eta_{n-1,n}$) will be calculated by assuming the secondary defect's occupancy can be calculated using a dynamical equilibrium. How to do this will be discussed when talking about the probability of having two defects present next to each other.

$$\begin{aligned} E(n^*) = \frac{k_n^+}{k_n^- + k_n^+}\frac{1}{k_n^- + k_n^+} + \\ (1-\eta_{n-1,n})\frac{k_n^-}{k_n^- + k_n^+}\left[\frac{1}{k_n^- + k_n^+} + E(n-1^*) + E(n^*)\right] + \\ (\eta_{n-1,n})\frac{k_n^-}{k_n^- + k_n^+}\left[\frac{1}{k_n^- + k_n^+} + E(n^*)\right] \end{aligned} \tag{2.15}$$

In equation 2.15 we write out the three cases. Taking a step to the right and finishing, taking a step to the left because it was unoccupied and taking a step to the left but failing because the other defect was present. Obviously the expression can be rewritten again by taking $E(n^*)$ to the other side. Note that we cannot define $E(1^*)$ as there would not be space to have a secondary particle. Calculating $E(2^*)$ would also be done with $\eta_{1,2} = 1$ and gives us $E(2^*) = 1/k_2^+$ like we saw before.

You may ask at this point, what was the point of going through all the different paths if we could simply use this formula? The issue is that this expression is not that accurate since we do not have the real probability $\eta_{n-1,n}$ available. Instead this value is assumed through what would be expected in an equilibrium state and does not take into account that the space in which the defect can move is constantly changing. It also does not take into account that if the secondary defect where to be introduced it cannot instantly interact with defect at the end of the nucleosomal DNA. This is also the biggest problem for this model, calculating these probabilities of having multiple defects present and if they are next to each other have to be done through approximations or simplifications.

For example calculating the probability of having a second defect introduced when the first is still in the system is a time dependent calculation. However we need to approximate this by averaging it out over the primary defect's lifetime to get an estimate of how often there will be two particle interactions. This can be done through using the properties of the

Version of October 10, 2022– Created October 10, 2022 - 09:08

Poisson distribution of the defect introduction:

$$\begin{aligned} P(\text{k events in interval t}) &= \frac{(rt)^k e^{-rt}}{k!} \\ P(>0 \text{ events in interval t}) &= 1 - e^{-rt} \end{aligned} \tag{2.16}$$

Here t denotes the time and r is the rate at which a process occurs, this then corresponds with our introduction rate. If we were to integrate this over the primary defects lifetime we get an estimate of the probability of having another defect introduced.

$$Pr(\text{Introduction}) = \frac{\int_0^{E[Defect]} 1 - e^{-rt} dt}{E[Defect]} = 1 - \frac{1 - e^{-r*E[Defect]}}{r * E[Defect} \tag{2.17}$$

Intermezzo

However this is where issues start to arise, we could even have two or three defects being introduced instead of only one. So while this approximation works fine as long as the defect lifetime is smaller than the inverse of the introduction rate, it will break down once they approach each other. As much as I would have hoped to give a magical solution to this issue to make the model work, sadly I do not have it. As we would expand our model to take into account three defects simultaneously we can only hope that the lifetime of defects becomes small enough that we do not need to expand to four defect states.

End of Intermezzo

For now we will still finish explaining how this model would work and show the same calculation for a secondary defect to the right. Having a defect present in the system is good to know but we would also like to know if it is currently neighbouring the primary defect. To calculate this we *smear out* our secondary defect and use its probability distribution in different positions to find an equilibrium state. This is done by making sure the outflow of a given state is equal to its inflow. For example taking a four wide space for our defect to move in and using $\eta_{i,5}$ to denote the probabilities (5 as this is the position of our *defect wall*). We can find the

probabilities by solving the following series of equations:

$$\begin{aligned} k_1^+ \eta_{1,5} &= k_2^- \eta_{2,5} \\ (k_2^- + k_2^+)\eta_{2,5} &= k_1^+ \eta_{1,5} + k_3^- \eta_{3,5} \\ (k_3^- + k_3^+)\eta_{3,5} &= k_2^+ \eta_{2,5} + k_4^- \eta_{4,5} \\ k_4^- \eta_{4,5} &= k_3^+ \eta_{3,5} \\ \sum_i^4 \eta_{i,5} &= 1 \end{aligned} \tag{2.18}$$

If interested one could write out the analytical expression but for convenience we just use a computational matrix solver to get $\eta_{n-1,n}$.

Now that we have finished describing our first possibility we can start to talk about the second scenario our defect can be in. The defect can be significantly slowed down if it was introduced while another was still present. Modelling this is a bit more complicated as we end up with much more paths that need to be considered. We start out with our primary defect in position n and the secondary defect in position n+1:

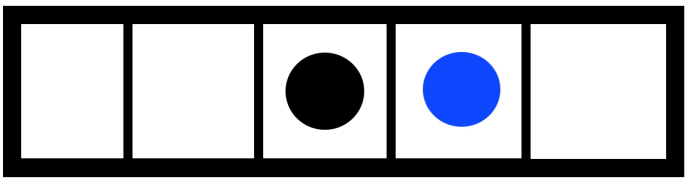

Figure 2.5: *Primary defect (black) and secondary defect (blue) being represented by particles in a grid.*

It already becomes clear that it will be a difficult process to get the primary defect to take a step to the right. Nevertheless, we will show the calculation that takes into account all second order steps that can be done. We start of by writing both initial steps:

$$\begin{aligned} E(n, n+1) = \frac{k_n^-}{k_n^- + k_{n+1}^+} \left[\frac{1}{k_n^- + k_{n+1}^+} + E(n-1, n+1) \right] + \\ \frac{k_{n+1}^+}{k_n^- + k_{n+1}^+} \left[\frac{1}{k_n^- + k_{n+1}^+} + E(n, n+2) \right] \end{aligned} \tag{2.19}$$

We observe that we have not yet terminated any path, this means there will be 8 paths that need to be considered at this point in time for our defects. In figure 2.6 we show all paths for $E(n-1, n+1)$.

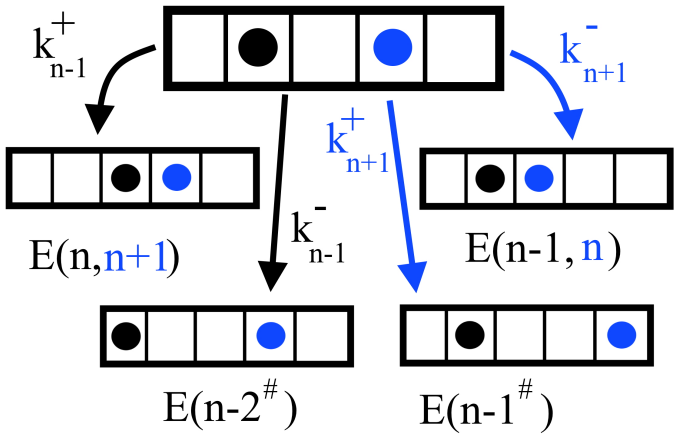

Figure 2.6: *All movement options of $E(n-1, n+1)$, the rates of these processes and the following expected lifetime duration underneath. We find that none of these paths terminate but the top two paths are what we are solving now or have already solved. We also find a similar situation where our particles are split far from each other which we denote with $E(n-2^{\#})$ and $E(n-1^{\#})$ depending on the primary defect position.*

While these paths do not terminate we can simply repeat what has been done before. We fill in earlier paths that we have already calculated ($E(n-1, n)$), we bring all $E(n-1, n+1)$ terms to the other side of the equality and we make a recursive model for all remaining paths.

In figure 2.7 we show all paths for E(n,n+2) and do the same process. We finally observe a path that terminates, we also recover another $E(n-1, n+1)$ term. Finally we have two more remaining paths, one of which that has the potential to terminate ($E(n^{\#})$), that can be written using the recursive model.

There is however a big issue with this solution, due to us having almost no terminating paths (k_n^+ and $E(n^{\#})$) it takes extremely long to resolve these paths. While this makes sense when a defect is blocking its path, we need to have a good estimate for the probability of having a secondary defect to the right of our primary defect.

Version of October 10, 2022– Created October 10, 2022 - 09:08

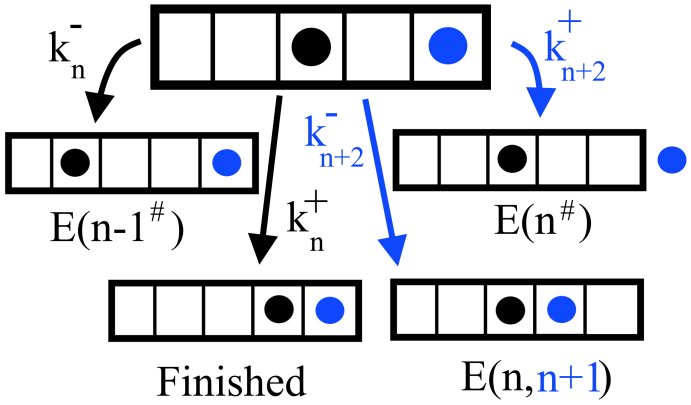

Figure 2.7: *All movement options of $E(n, n+2)$, the rates of these processes and the following expected lifetime duration underneath. We find one terminating path and one path that has the potential to terminate $E(n^{\#})$. We also recover another $E(n, n+1)$ term and one more situation where particles are split far apart from each other.*

$$
\begin{aligned}
E(n,n+1) = \frac{k_n^-}{k_n^-+k_{n+1}^+}\frac{1}{k_{n-1}^-+k_{n-1}^+ k_{n+1}^-+k_{n+1}^+} * \Bigg\{ \\
k_{n-1}^+\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_{n-1}^-+k_{n-1}^+ k_{n+1}^-+k_{n+1}^+}+E(n,n+1)\right]+ \\
k_{n-1}^-\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_{n-1}^-+k_{n-1}^+ k_{n+1}^-+k_{n+1}^+}+E(n-2^{\#})+E(n-1^{\#})+E(n^{\#})\right]+ \\
k_{n+1}^+\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_{n-1}^-+k_{n-1}^+ k_{n+1}^-+k_{n+1}^+}++E(n-1^{\#})+E(n^{\#})\right]+ \\
k_{n+1}^-\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_{n-1}^-+k_{n-1}^+ k_{n+1}^-+k_{n+1}^+}+E(n-1,n)+E(n^{\#})\right]\Bigg\}+ \\
\frac{k_{n+1}^+}{k_n^-+k_{n+1}^+}\frac{1}{k_n^-+k_n^+ k_{n+2}^-+k_{n+2}^+} * \Bigg\{ k_n^-\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_n^-+k_n^+ k_{n+2}^-+k_{n+2}^+}+E(n-1^{\#})+E(n^{\#})\right]+ \\
k_n^+\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_n^-+k_n^+ k_{n+2}^-+k_{n+2}^+}\right] \\
+k_{n+2}^+\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_n^-+k_n^+ k_{n+2}^-+k_{n+2}^+}+E(n^{\#})\right]+ \\
k_{n+2}^-\left[\frac{1}{k_n^-+k_{n+1}^+}+\frac{1}{k_n^-+k_n^+ k_{n+2}^-+k_{n+2}^+}+E(n,n+1)\right]
\end{aligned}
\tag{2.20}
$$

We can then finally write out an expression for $E(n, n+1)$ in equation 2.20, we simply reorder all terms like before and add steps where needed

to make them terminate (this can be done by adding $E(n-1^{\#})$ and $E(n^{\#})$ terms where appropriate). Now it becomes time to combine all results to get an estimate of the lifetimes. This is done by multiplying the single defect lifetime and the two defect lifetimes with their respective probabilities.

We repeat a similar calculation to determine $E(n^{\#})$ all the way up to $E(\tau - 1^{\#})$ where τ denotes the last position the defect can be present on before escaping the nucleosome.

$$\begin{aligned} E(n^{\#}) = (1-\eta_{n+1,n})\frac{k_n^+}{k_n^- + k_n^+}\left[\frac{1}{k_n^- + k_n^+}\right] + \\ \eta_{n+1,n}\frac{k_n^+}{k_n^- + k_n^+}\left[\frac{1}{k_n^- + k_n^+} + E(n^{\#})\right] + \\ \left[\frac{1}{k_n^- + k_n^+} + E(n-1^{\#}) + E(n^{\#})\right] \end{aligned} \tag{2.21}$$

We obviously cannot have $E(\tau^{\#})$ since there is no space to the right for another defect. We also need to adjust $E(\tau - 1^{\#})$ since we could never move out as we do not let the secondary particle evolve in this calculation. To fix this one could manually change the value of $\eta_{\tau,\tau-1}$ and $\eta_{\tau-1,\tau-2}$ to lower values to adjust for this escaping effect.

Finally we could calculate the probability of still having a second defect in the system by integrating over the lifetime of the primary defect. Here things get more complicated again since we do not have a clear expression as to when a defect would leave. One could assume it is given by a Poisson process with as rate the inverse of the expected lifetime. We could then integrate over the probability density of our primary defect being introduced while integrating over the life of the primary defect as well.

Now we can combine the probability of having a secondary defect in the system and multiply it with our $\eta_{n-1,n}$ and $\eta_{n+1,n}$ to get an estimate for $Pr(n-1,n)$ and $Pr(n,n+1)$. Using all of this we can then finally write out a first approximate expression for the expected time to take a step to the right on an arbitrary position n.

$$\begin{aligned} E[\text{Right (n)}] = Pr(n-1,n)E[(n-1,n)] + Pr(n,n+1)E[(n,n+1)] + \\ (1 - Pr(n-1,n) - Pr(n,n+1))E[\text{Right (n) single}] \end{aligned} \tag{2.22}$$

Version of October 10, 2022– Created October 10, 2022 - 09:08

Sadly we did not have enough time to continue this process for triple defect states, we also do not observe the rise and fall in defect lifetimes that we hoped to achieve from this model. The main problem has to do with the calculations of having multiple defect states present. If we were to sum the probability of having a double state on the left and one on the right together we would exceed 1 which means we need to use triplet states as well. A possible solution to these issues would be to start exploring more paths, calculating expected durations for triple defect states and improving the probability calculations. See section 5.2 for a more detailed description.

2.4.3 Duplet and triplet defect states

The actual presence of these double or even multiple defect states are worth investigating more. They have the ability to greatly affect the DNA movement timescale depending on the introduction rate of the remodeler. The lifetime of defects in the system can have very different timescales. This is due to the defect having to overcome an energy hill or going down when moving away or into a preferred position for the sequence. If a defect were to take a long time to leave the system, introducing a secondary defect afterwards will reduce the lifetime. This is due to the secondary defect acting as a wall preventing it from moving too far away from the exit. Conversely, for the other defect introduced while there is still one present it can increase the defect lifetime. This is due to the first defect preventing the second from reaching the exit until it leaves.

According to this reasoning, a build-up of defects could thus happen when there is a strong energy hill it has to overcome. This build-up of defects then lowers the amount of time it would take for the initial defect to leave the system and let the sequence take a step. This build-up has been observed experimentally by detecting both DNA ends with FRET [1]. It was assumed this build-up was present to ensure stability during ATP-dependent remodeling. They also observed a delay between the release of the defects on both ends of the nucleosome. An interesting observation made was that this time delay was more pronounced for limiting ATP concentrations. While our model may not be too representative of reality, it seems to mimic similar trends and could potentially explain why this happens.

One possible explanation would be that the limited ATP concentrations

corresponds to a smaller intro rate for our system. This in turn then directly affects the lifetimes of the defects present. We give a more in depth explanation on how these lifetimes change in Section 4.2. This increase in lifetime for smaller intro rates would be more pronounced for the side of the remodeler where the defects have more SHL's to overcome. So even though both ends will release their defects slower, this will be much more pronounced on one side. This way our system can mimics this delay for limited ATP concentrations.

2.5 Gillespie Markov Chain algorithm

To simulate our system we will need to use a Markov Chain Monte Carlo (MCMC) algorithm. We decide to use the Gillespie algorithm since this allows us to keep track of different possible transition with different rates very effectively. We will now give a brief summary of this method as described in [16].

When using static time steps to determine if an event has happened one has to be careful to take a good value for the time step. This is needed to be accurate enough such that our step size is not too big. But we also need it to be large enough that it works efficient enough. If we were to have f(t) as our probability density function, we could calculate the probability of an event by f(t)dt in the interval [t, t+dt]. We can write out our probability function by dividing t=n*dt. If we assume an event with rate k occurs in [t, t+dt] but not before, we can write out the following:

$$f(t)dt \approx (1-kdt)(1-kdt) * ... * (1-kdt) * k * dt = (1-kdt)^n kdt \quad (2.23)$$

which follows from the taylor expansion of e^{-kdt} and $1-e^{-kdt}$. Filling in dt and taking the limit then gives:

$$f(t)dt = \lim_{n\to\infty}(1-\frac{kt}{n})^n kdt = e^{-kt}kdt \quad (2.24)$$

It is easy to verify this is indeed a probability distribution function by integrating from 0 to infinity and finding that it is 1. Now that we have a probability density function we can set up the cumulative probability density function to an arbitrary time t. Doing this allows us to link a random number to the time at which an event occurs.

$$F(t) = \int_0^t f(\xi)d\xi = \int_0^t e^{-k\xi}kd\xi = 1-e^{-kt} \Rightarrow t = -\frac{ln(1-F(t))}{k} \quad (2.25)$$

We have rewritten our equation to give us the time given the rate and the value of our cumulative distribution function. By simply generating a random number for this distribution function we can directly calculate a random time for an event with rate k.

We can use this method too if we have multiple transition possible. To do this we would have to adjust the probability density function to accommodate all transitions. However since we model all our events as Poisson distributions, we can simply use the Poisson distribution of the sum of all the rates.

$$X_i \sim Pois(k_i) \text{ for } i = 1, .., n \Rightarrow \sum_i^n X_i = Pois\left(\sum_i^n k_i\right) \tag{2.26}$$

We can then define the individual probabilities for the transitions to have occured by diving their rate by the total transition rate.

$$P_i = \frac{k_i}{k_1 + k_2 + ... + k_n} = \frac{k_i}{k} \tag{2.27}$$

Then by generating a second random number one simply has to look in which of the following intervals it ends up in to find which transition happens.

$$\left\{ \left[\sum_{i=0}^{r} P_i, \sum_{i=0}^{r+1} P_i \right] \;\middle|\; r \in \mathbb{Z}_n, P_0 = 0 \right\} \tag{2.28}$$

The big advantage of this method is that different processes with widely ranging rates can be combined without sacrificing time or accuracy. This is especially important later when we use a remodeler, here the rates of introducing a defect are of the order 10^{-6} while defect motions are around 3. Using this method it becomes trivial to introduce defects while also simulating defect motion.

A big downside to this method is that this requires extra computing and keeping track of all the positions and possible rates. However due to our system being very limited (13 SHL's) we do not have to worry about this.

Chapter 3

Simulation

This chapter will consist of an explanation of the code written and a description of the final simulations. The code itself consists of two main simulations and some supporting functions. In 3.1 we will introduce these functions, how and where the code uses these. In 3.2 we talk about the first major simulation which simulates the twists defects propagation throughout the nucleosome. In 3.3 we discuss the main simulation that was used to make the nucleosome move over a DNA sequence and what the parameters do. A deeper discussion of the current issues and what can be done to further extend the code can be found in 5.1.

3.1 Dependencies and functions

3.1.1 Global code structure

The code itself, shown in figure 3.1, consists of one main script, some small supporting functions and two main functions that serve to calculate energy values and the actual simulation of the defect motion. Both of these functions have their own sub-functions which are shown in figures 3.2 and 3.3. Our first major simulation is encapsulated entirely into the second main function. The second simulation is done by repeatedly using our first simulation as a function. We will now shortly explain what these subfunctions do and what they depend on. In Section 3.2 and 3.3 we will go more in depth on what the main functions do and what we are simulating.

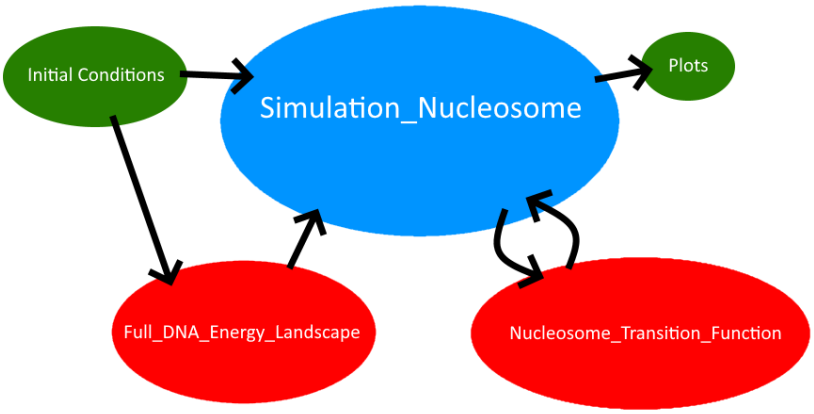

Figure 3.1: *Simplified flowchart showing interactions in the main code. Main simulation (blue) uses the two main functions (red) together with some initial conditions to create the desired plots.*

3.1.2 Full DNA Energy Landscape

The main goal of this function is to calculate an energy landscape. This will create the energy barriers that the defects will have to traverse to move between SHL's.

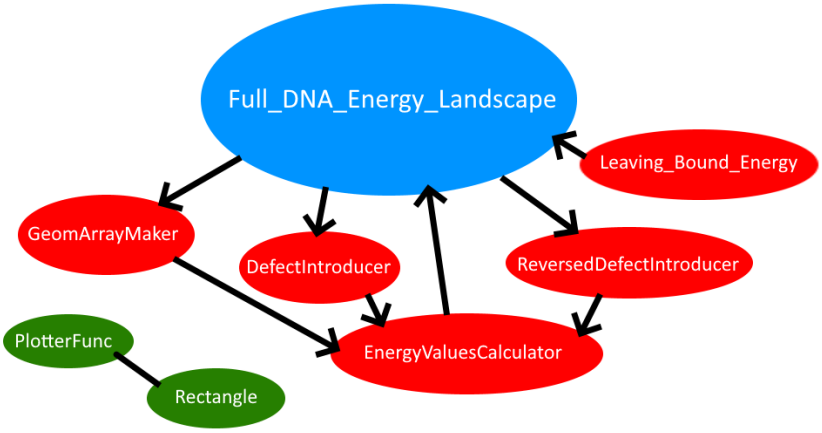

Figure 3.2: *Flowchart of how the main energy function (blue) uses its sub-functions (red) and support functions (green) to calculate what the energy landscape of the defects around the nucleosome looks like.*

Geom Array Maker

As seen in chapter 2, we use the rigid base pair model where we simply use 6 spatial parameters to determine the energy costs. In this sub-function an array is made using the geometry described in equations 2.4 and 2.5. It then outputs this array and uses equation 2.7 to calculate the undefected energy values.

Defect Introducer

The defect introducer takes the geometric array, a position and the defect type (K10 or K20) and generates an undertwist defect according to equation 2.6. We obviously cannot just insert a base pair so we would have to choose from which side we take a base pair to create a defect. We choose to move all base pairs left of the defect by one to the right and all base pairs to the right of the defect are undisturbed.

This defected geometry is then inserted into the Energy Values Calculator and we subtract the undefected energy to get the defect cost.

Reversed Defect Introducer

The Reversed Defect Introducer does the exact same, however now it takes a base pair from the right while leaving the left side undisturbed. This function could be merged by just adjusting the defect type number however writing it like this makes it easy to follow too.

Energy Values Calculator

This function takes any geometric array, defected or undefected, together with the DNA sequence and calculates the energy costs for every base pair step. Currently the slide and shift contributions are ignored due to their effect being negligible.

Leaving Bound energy

The final energy barrier to leave the DNA cannot be calculated the same way as the previous barriers. This is due to there not being a constraint on the geometry on the other side of the binding site. This function calculates a constrained geometry on one side and uses this to calculate what the energy cost would be to go over the final binding site.

3.1.3 Nucleosome Transition Function

This function takes an energy landscape and simulates what it would be like to have a remodeler sit on it and introduce twist defects.

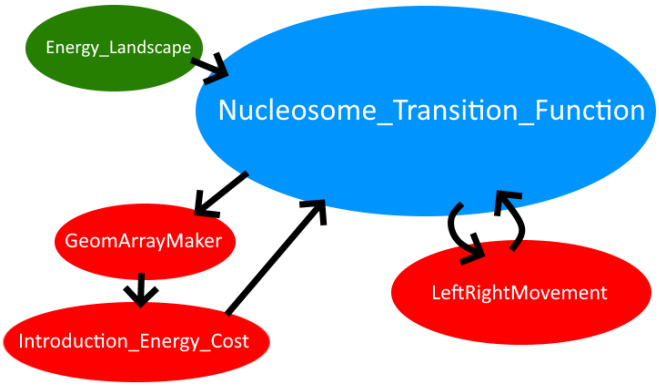

Figure 3.3: *Flowchart of how the main simulation function (blue) uses its sub-functions(red) and an energy landscape (green) as input. This is done to simulate the movement of the defect around the nucleosome.*

Geom Array Maker

The same as described before in section 3.1.2. Here the geometry however is inserted into a second function to calculate an introduction rate.

Introduction Energy cost

This function calculates the energy cost to have an overtwist and undertwist present right next to each other. This is done to model the movement a remodeler would do. The purpose of this function is to give the option to have our defect introduction rate depend on the actual energy cost of moving it into that position. We estimated this by looking how the energy landscape of introducing a defect with an ATP-remodeler changes for a molecular dynamics simulation [6]. They observed it roughly halves the energy barrier so we make the variable introduction rate equal to half of the energy cost of introducing an overtwist and undertwist at both ends of the remodeler. For this we observe our variable intro rate comes close to the introduction rates used in this thesis ($\approx 10^{-6}$). It should be noted that even though we have added this possibility, we did not use it that frequently and might be of interest for further investigation.

Left Right Movement

This function is used to make the defect simulation go more smoothly. Using the Gillespie algorithm we need to know every action that can be taken. This function tells us which defects can move left or right. This is then sent back to the Nucleosome Transition Function to take a single step somewhere which is then repeated for as long as required.

3.1.4 Supporting functions

There are also some minor functions which do not directly serve a purpose in the code but have been used to make sure the code can run.

PlotterFunc

This function takes the geometric array and an array of binding positions and creates a visible geometry of the DNA superhelix where base pairs are represented by rectangles. This is used to verify that the used geometry and the defect geometry are in fact correct. When we plot the undefected and defected DNA over each other, we see them perfectly overlap except for the defect location. Note that the defect introducer function ”cuts out” the first base pair so this should be taken into account when plotting.

Rectangle

This is a sub-function of the PlotterFunc. It is responsible for plotting the rectangles and colouring of the binding sites.

PositionalDependentSequence.py

This is a Python script written to make DNA sequences. These sequences can be random with equal probability for all base pairs independent of the position, or random with positional preference as observed in real DNA. It uses the probabilities of a base pair step to occur in a certain position and uses conditional probability on the previous base pair to create a sequence. The probabilities to occur are taken from the following MATLAB script.

Probability

This script uses the theory of Transfer matrices as described in [11]. It can then calculate the probabilities for the base pair steps to occur.

3.2 Setup Nucleosome Transition simulation

Here we shall explain our first main simulation. The goal of this simulation is to keep track of the duration for a twist defect to propagate through the nucleosome and leave on one end. This is done by inserting an energy landscape into the simulation.

This energy landscape is taken to be 10 step periodic since we are not interested in investigating combinations of sequences yet. Instead we will be using a few different landscapes, first of all an all AAAA-sequence to verify our code. Then we will use a fully random sequence where we will average all energy values modulo 10 so we are not just looking at a single sequence. Finally we will be using a randomly generated positional dependent sequence where we average the energy values modulo 10 again. The two previous random sequences are therefore taken to be much longer than 147 base pairs. Sometimes multiple 147 length sequences will also be used for the same goal.

We then make use of the *LeftRightMovement* function in combination with a Gillespie algorithm to evolve our system. This function detects all possible actions that can be taken. The position of the nucleosome is carefully tracked to see which energy barrier the defects feel. Multiple defects can be present at the same time while individually feeling different energy barriers. Movements of defects into each other are not allowed due to the high energy costs. Defects also cannot pass through the remodeler in this simulation but must escape on the other end.

These defects will feel their own energy landscape which can be going up or down depending on the nucleosome position. This energy landscape then determines the lifetime of the defect in our simulation. We track the time for when defects are introduced and leave the system to calculate their lifetime. This is then plotted and compared to a theoretical calculation for single and double particle systems as described in section 2.4.

There are three different scripts that are used to do this. First of all *Twist Lifetime Single*, this is a script that simulates only on one side of the remodeler and has a calculation for the theoretical lifetimes implemented into it. Secondly, *Twist Lifetime Double* which simulates particles on both sides of the remodeler. And finally the function *Chromatin Transition Function* which is used for the Nucleosome simulation when moving over the DNA. This final script does not average energy values but instead uses the

pre-calculated energy barriers of the entire sequence.

3.3 Setup final simulation

3.3.1 Code Explanation and goal

Now we shall discuss the second main simulation. The goal of this simulation is to allow the nucleosome to reposition itself on a given DNA sequence. We calculate the entire energy landscape and allow the remodeler to move the nucleosome in two directions.

The energy landscape is calculated by going over every nucleosome position. We then insert the K10 and K20 defects on every SHL. This is then repeated in the other direction as well where we use a different defect function. These energy values then pre-calculate all the barrier heights which are saved and returned to the main simulation.

We then start on a random position and repeat the following process. We choose a random direction. We calculate a random lifetime according to a Poisson distribution. We give the right parameters to our *Nucleosome Transition Function*. The current iteration of this function only simulates one side of the remodeler and something we wish to update, see Section 3.4 for more details. This gives us the new position, time passed and the time spent in different positions. We save these values and repeat this process until our total duration has passed. Once complete we calculate the fractional time spent in every position and plot it together with an uniform distribution. We also have the option to repeat this simulation multiple times with some parameters changed and plot these together.

The goal of this simulation is to see how the positional occupancy of the nucleosome looks like. We wish to compare this plot with how a Boltzmann distribution would look like. We would also like to see how the occupancy can change by varying our parameters or even the DNA sequence.

3.3.2 Parameter explanation

The parameters used in the final simulation should receive a bit of explanation to make it easier to follow what is being done. While some of them are trivial, others require a bit more background or information.

DNA

Here we can change the actual DNA sequence itself. We will mostly be using the same randomly generated DNA sequence however there are a few simulations where we use a different random sequence or an all AAAAA... DNA sequence.

Total Duration

This is simply the amount of time we wish to run our simulation for. When choosing this value we have to make sure we have a sufficient amount of sampling on the different positions for a good statistical result. We also wish to suppress statistical noise on simulations where differences are small. In general we can approximate how long this should be by putting a constraint on how often we wish to sample every position. For example for 100 times this gives:

$$T_{total} = 100 * N_{Positions} * T_{single\ step} \approx 100 * 2 * N_{DNA} * E[t_{Introduction}] \quad (3.1)$$

where $E[t_{introduction}]$ is simply the inverse of the introduction rate and the factor 2 comes from having 2 directions. Taking our sequence of length 350 and an intro rate of 10^{-6} we get roughly a total time of 10^{11}. Note that this approximation breaks down for high intro rates since our approximation is no longer valid, for these simulations one should check the console outputs to make sure sufficient steps are taken.

Right now a 10^{11} simulation takes approximately 20 minutes, however this is very much dependent on all the parameters used. Changing the intro rate obviously has a large effect on how long we need to wait for each defect. But changing the damp factor could also make defects fall of quicker which again causes a decrease in our simulation duration.

Twist Duration

We use this value to determine how long the remodeler will stay on. This is done by using a Poisson distribution with parameter = Twist Duration.

Intro Rate

This is simply the introduction rate which we use for our remodeler. Changing the intro rate also requires us to change the total duration and the twist duration to make sure our simulation still works well.

In general dividing our intro rate by a factor of 10 requires us to multiply the Twist Duration and the Total Duration by a factor of 10. This makes it so we still have sufficient sampling and our remodeler does not detach too often before a defect can even be introduced.

Damp factor

This is the damp factor that we used to change the energy values of displacements in our model.

Variable Introduction

This allows for a method to have a variable intro rate as mentioned in section 3.1.3.

Passable Chromatin

This is a boolean that makes it possible for twists to go back through the remodeler and leave on the wrong side undoing a twist defect.

Chromatin Barrier

This is the rate at which a twist defect could pass through the remodeler if Passable Chromatin is set to true.

Position remodeler

Here we can change which SHL the remodeler attaches to. In the literature we find that the remodeler attaches itself to SHL=2 [6, 12]. It is then able to create two defects on SHL=1 and SHL=2. For our interests we focus on the defect that has the most space to move around. For this we would use SHL=-1 since we neglect the left part of the remodeler and only simulate what is to the right which would be the 8 SHL's the defect created on SHL=1 (now SHL=-1 in how we orient it). There has been a mistake however in the project and we used SHL=-2 throughout, while originally this made sense the way the defects were introduced was not properly adjusted yet.

Version of October 10, 2022– Created October 10, 2022 - 09:08

Chapter 4

Results

In this chapter we will discuss some of the results we have obtained. We shall start out in 4.1 by showing the energy landscapes we obtained. These serve as the foundation for how our defects propagate through the system. We will be using a few different sequences to see how they affect the landscape. In 4.2 we show the defect lifetimes we got through our first simulation. We compared these with our analytical one and two-state solution but were unable to fully model the two-state regime. Then in 4.3 we show the occupancy plots obtained from our second simulation. We will show the interesting parameter sweeps and compare it with the Boltzmann distribution of a free nucleosome. Finally in section 4.4 we discuss the Duplet and Triplet states and how we see them in our results. More plots and validity tests have also been done and can be seen in the Appendix.

4.1 Defect energy landscapes

We used the RBP model to calculate the associated energy values of our DNA. We did this by creating a geometry associated with undefected or defected DNA wrapped around a nucleosome. These defected DNA geometries were generated using the two *DefectIntroducer* functions depending on the direction. We verified the geometry by plotting them over the undefected geometry using the *PlotterFunc*. This result is shown in the appendix and gives a perfect overlap except for on the defected SHL.

We inserted this defected geometry into the energy values calculator to get the energy for each base pair according to our RBP model. This was

then summed over all base pairs and subtracted by the energy of the undefected DNA. We do this to get an estimate of the defect cost however this is not necessary for the simulation since it only looks at the energy difference between two neighbouring states. We repeated this for our entire sequence, thirteen K10 defects and twelve K20 defects. For the K20 defects we also had to sum the unbinding cost to release from the nucleosome since we now have an open binding site. We decided to use 12 k_BT for all binding sites as this is the estimate used in [11] and gives reasonable energy barriers for our model.

Using this process we then decided to make some plots showing what the energy landscape looks like when introducing an undertwist defect into the system. This is done for different types of sequences to show the positional dependence of the DNA. For a deeper analysis of the positional dependence, see Appendix A.1.1

As a test we first tried some uniform and repeating sequences. These are chosen because they should be able to give a simple flat and repetitive landscape when inserted.

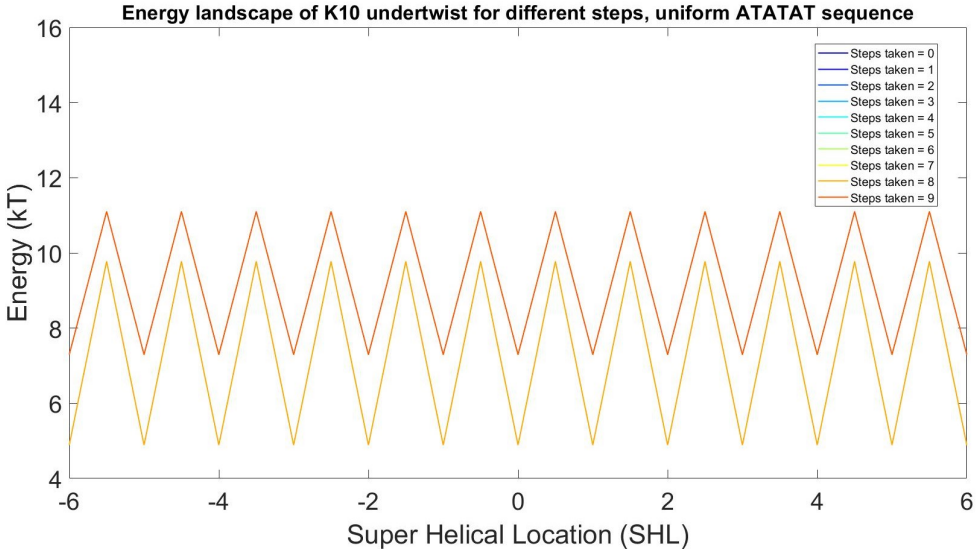

Figure 4.1: *Repeating energy landscape of ATATAT... sequence. The energy landscape is repeated modulo 2 so even though we have 10 graphs we only see two distinct landscapes. We use the convention that K10 defects correspond to integer SHL values and K20 to half integers.*

We observed that the energy landscape does not depend on the SHL on

which the defect is located. We also see that the energy landscape repeats itself every two steps. In general the energy landscape looks good, K20 defects cost more energy than K10 defects which is what we expected to see. However in this simplified landscape we will not be able to observe any interesting properties of the DNA.

After this we decided to use a fully random DNA sequence and plot its energy landscape (see Appendix A.1.3). This sequence does not take into account the positional preferences of the DNA base pair steps but generates them all uniformly at random. We did not observe any dependence on the SHL, nor did the energy landscape change for different steps taken. This is again what we would expect since shifting a fully random sequence does not change the randomness.

Now we come to figure 4.2, the most important plot of this section, the energy landscape for a positional dependent sequence. Here we observed the positional dependence giving us a non-flat landscape and inducing a preferred position.

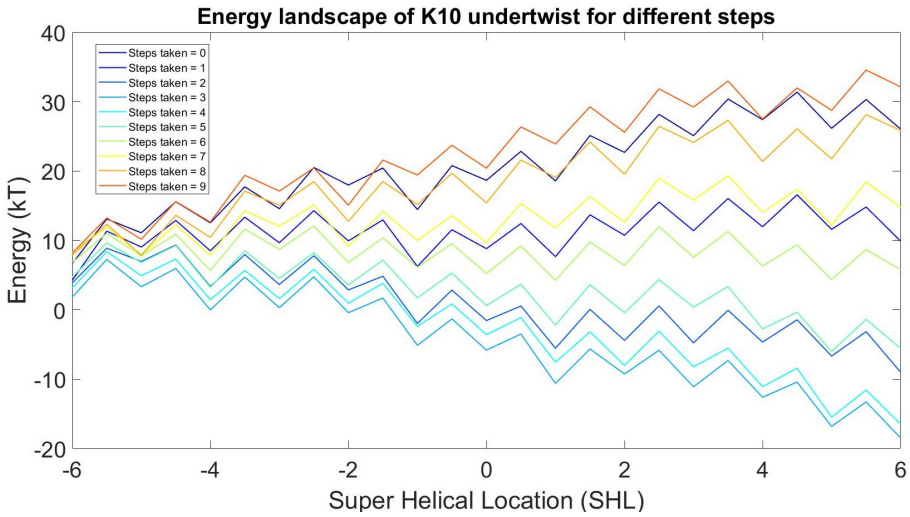

Figure 4.2: *Energy landscape of positional dependent sequence. The energy landscape depends very strongly on the SHL and the steps already taken. Note that there is a counter-intuitive displacement in this plot. If we were to follow one colour from SHL=-6 to SHL=6 we may try to estimate the local minima or maxima of our system. However we should note that after SHL 6, even though we moved our entire sequence by one around the nucleosome, we actually have one less step taken (n_{steps} -= 1). This is caused by an undertwist propagating being comparable to the movement of a vacancy and not an actual base pair.*

We also averaged out this energy landscape by taking all the contributions modulo 10 (See Appendix A.1.3). This gives a much nicer and clear landscape to see what is happening. For the first defect lifetime simulation we used the averaged landscape, however for the actual Nucleosome occupancy we ended up using the non-averaged landscape. The reason for this is that we did not want to look into too much detail for the twist lifetime plots of specific sequences but a general idea, while for the Nucleosome occupancy we care about where exactly we are on the sequence.

The rise and fall of the energy is due to the DNA sequence having position dependence. Moving parts of the sequence for a defect causes an index shift further along the DNA sequence. In this plot we start with a defect at SHL=-6 and make it move from here all the way to SHL=6. Along the way it disturbs and moves all base pairs one to the left (since it is an undertwist we step opposite to where our defect moves), and causes the energy to go uphill or downhill depending on the amount of steps already taken. Doing this one can find that the local extrema of the full energy landscape are located at the steps with a flat landscape. This gives us $n_{steps} = 1$ and $n_{steps} = 6$ as our minimum and maximum respectively.

This differentiation between minima and maxima can be made by looking at the rise or fall of the landscape one step later. If the energy landscape of the later step seems to rise (like $n_{steps} = 7$ and 8), it means the current step is a higher energy level and thus a maximum. If it were to go down (like $n_{steps} = 2$ and 3), it means the current step is a lower energy level and thus a minimum.

We decided to then even further verify this result by plotting it together with the energy of the nucleosome. We simply take the total defect energy without subtracting the base DNA energy cost and flipping the energy horizontally. Our SHL positions are then sub-positions of the base pair step position so we can combine the plots.

Doing this we hoped to verify if the energy costs of moving our DNA sequence by small steps all the way through the nucleosome would end up giving the same energy cost to move the entire DNA sequence by one step. This should obviously hold, once the defect gets removed out of the system we end with the same energy level however it is always a good idea to make sure.

Version of October 10, 2022– Created October 10, 2022 - 09:08

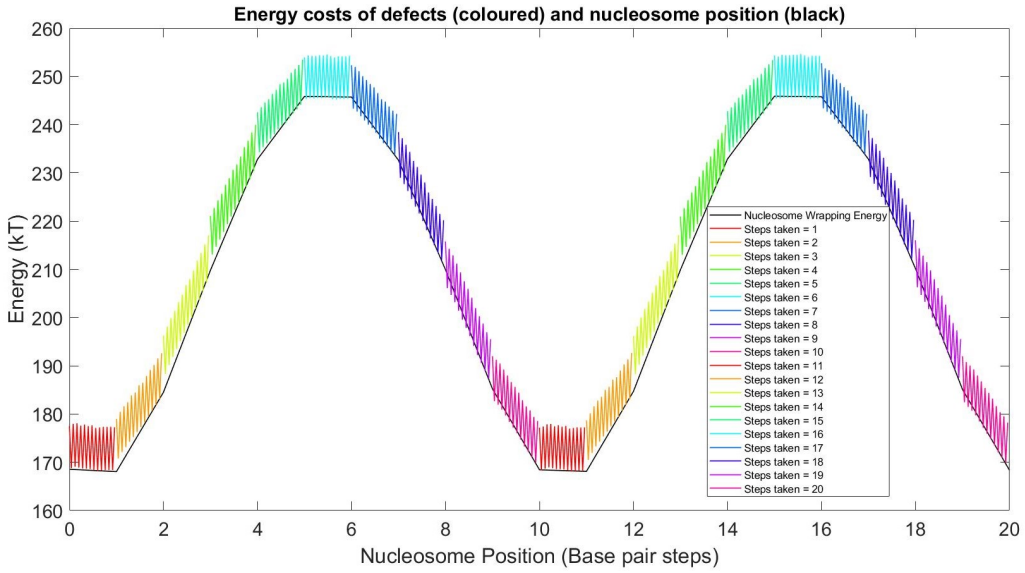

Figure 4.3: *Correspondence between the defect energy for different SHL positions and the energy of the DNA wrapping around the nucleosome. We see that the rise in energy for the defects through the SHL is the same as the rise in energy to take a full base pair step. We also observe that the position 1 modulo 10 corresponds with a minima and 6 modulo 10 with a maximum like we assumed. The slightly higher energy for the steps taken are due to them having a defect present in the system which has a cost around 10kT. Note that the K20 defects are plotted without the unbinding contribution*

In figure 4.3 we observed that the energy of the defects follow the Nucleosome wrapping costs almost perfectly. The difference along the y-axis is caused by the energy costs of the defects in our system. However the total energy barrier of our nucleosome wrapping seemed to be very high and not correspond to what has previously been observed [23] and calculated [14]. The reason for this is due to our fixed geometry, heterogeneous stress cannot be locally distributed along DNA sections.

We decided to introduce a damp factor on the energy cost of the displaced base pairs. By multiplying the energy for the displaced base pairs but not on those that are in a defect we are able to retain energy barriers around 3 kT between K10 and K20 defects while also lowering the total energy barrier that has to be overcome to a more realistic value. We chose to use a total energy barrier around 15 kT which leads to a damp factor of about 0.18-0.19 for our DNA.

Throughout this project we have used different damp factors, either to analyse what a realistic damp factor will do (Damp = 0.18) or to get a more exaggerated idea of what is happening in our system (Damp = 0.8). We used this last one when we were not interested in getting a realistic estimate but rather to get an idea of how something affects the system. Small differences might be hidden by statistical noise so by increasing the energy hill the defects have to overcome, we were able to see these effects much easier. Later simulations using more realistic damp factors were then repeated and run for longer once we knew there was something interesting.

Finally we calculated some examples of how the energy landscape looks if we were to use different damp factors. Here we show an energy landscape of our realistic damp factor of 0.18. We also have some additional energy landscapes in Appendix A.1.3 showing a damp factor of 0 and its average landscape.

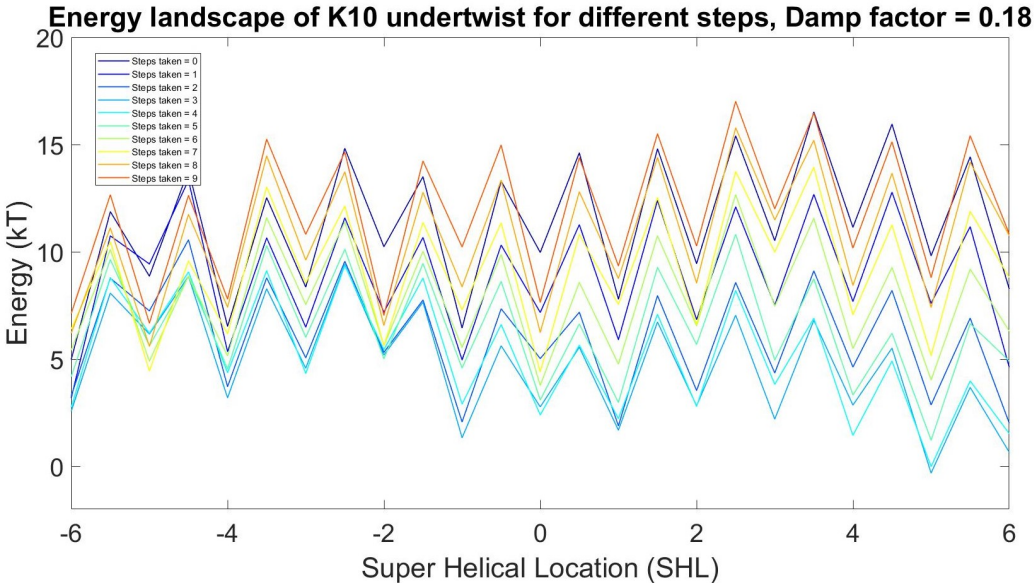

Figure 4.4: *The rise and fall of the energy for realistically damped DNA. The rise and fall might be a bit harder to see but looking at the averaged out value in A.1.3 we find that it is still there.*

Using this damp factor we were able to recover more realistic energy landscapes. The issue with this is that we might not be able to see all the effects as clearly when our landscape is too close to flat. However using an undamped landscape can cause issues where our K10 defects have a higher

energy cost than the neighbouring K20 defect. While this is not seen in Figure 4.2, it has been seen before and caused the Gillespie algorithm to crash. Due to this we used a slight damp factor (0.8) even when we wanted to see an exaggeration of what would happen. We then combined this with setting the energy barriers to 0.1 kT if they were negative otherwise.

4.2 Defect lifetimes

Once we made the energy landscapes we continued to simulate the movements of the defects. This is done through our first simulation, the nucleosome transition simulation as described in Section 3.2. When simulating this we will keep track of the time defects were introduced and when they left the system. We then use this to calculate the lifetime of the defect and the time difference between the release of defects. The lifetime will be plotted and compared to our theoretical model while the time difference will be used to find duplet and triplet states.

The energy landscape used for this simulation consists of 10 individual energy landscapes. Each individual landscape is the energy a single defect would feel when traversing through the different SHL's. These individual landscapes are defined to be at a certain n_{step} which denotes how many steps the DNA has already taken. The total landscape thus gives us the energies the defects feel for $n_{step} \in \{0, ..., 9\}$. After taking 10 steps the DNA has made a full movement and returned to its initial positioning. While some of the DNA has fallen off and some got introduced this does not matter for our first simulation. Instead what we will do is calculate the averaged landscape when considering all positional dependent base pair probabilities. This is done by averaging the landscapes for each n_{step}. So a sequence of length 10*N would be averaged with N different landscapes. Doing this, we sample all different base pairs and their frequency for every n_{step} which then creates an average energy landscape.

There are three main factors which we can change to affect the defect lifetimes. We could try to use different sequences, doing this gives a different energy landscape which in turn causes the defect lifetimes to be different. If we were to use a sequence with a flat energy landscape we should see that the defect lifetimes remain constant. In the appendix we have added a defect lifetime plot for an all AAA sequence. More interesting however is to use energy landscapes for positional dependant sequences. We will be using the energy landscape as seen in figure A.4

Version of October 10, 2022– Created October 10, 2022 - 09:08

Secondly we can choose to simulate or neglect the other side of the remodeler. When introducing a new defect in the simulation it is important that there is no defect present on the SHL on which the remodeler will insert one. If we only look at one side we are neglecting that there could be an opposing defect present on the other side. In this case it would not be possible to introduce a new defect. When only simulating one side we would not know this and introduce a defect nevertheless. As a test we decided to compare if including this effect is significant.

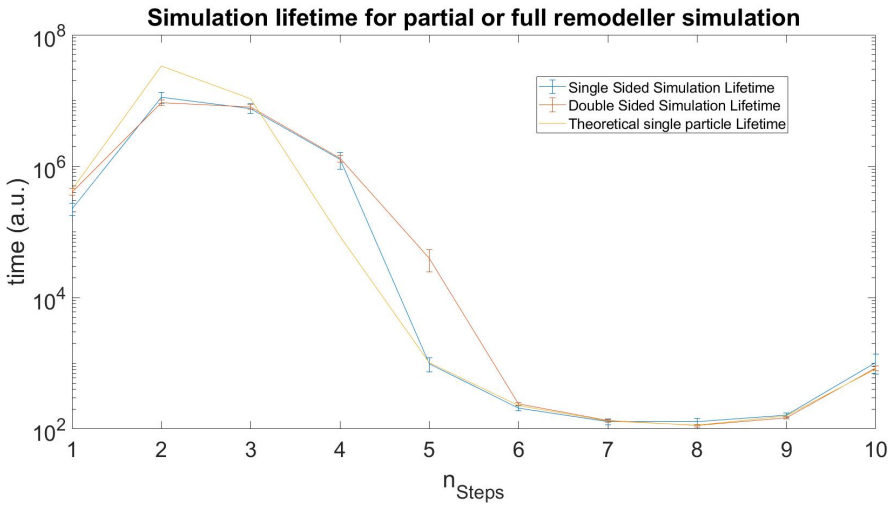

Figure 4.5: *Difference between partial and full remodeler simulations using intro rate of* 10^{-6}*, this gives an almost identical lifetime for the defects.*

In figure 4.5 we find that this effect is mostly negligible on the actual lifetime itself. There seems to be a slight difference on the lifetime at position 5 though however. We decided to continue simulating on one side for both simulations. This was done due to simulations taking too long when simulating two sides. The result thus allows us to speed up the simulation without affecting the system too much.

Finally in figure 4.6 we changed the introduction rate of defects by our remodeler. We wanted to know how big the effect would be on the lifetimes of the defects. Changing the introduction rate, we were able to tweak the amount of defects present around our nucleosome. We let our introduction rate range from really low such that we have mostly single defect movement, all the way to really high such that we end up with a *conga line* of defects.

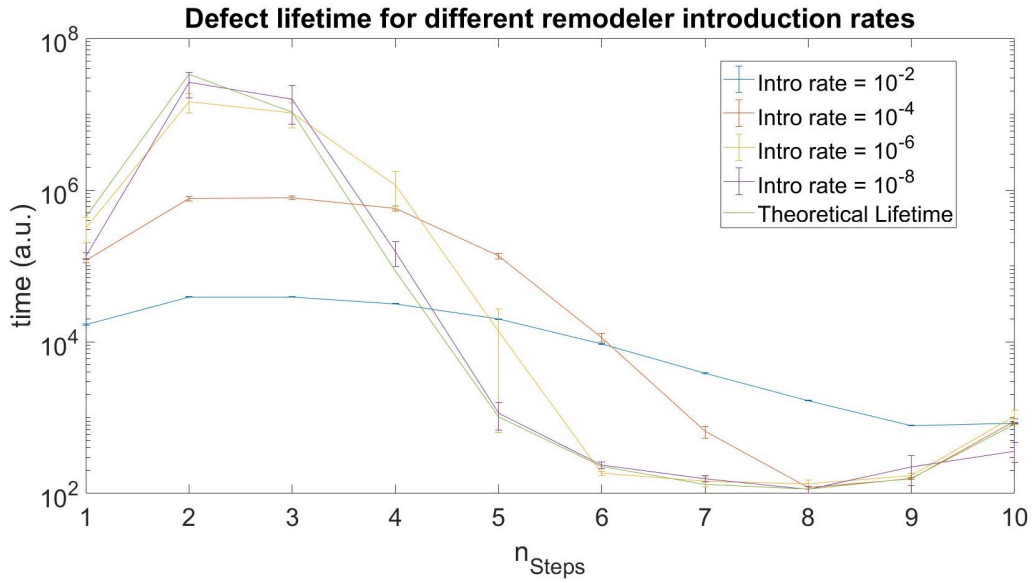

Figure 4.6: *Averaged defect lifetimes for different introduction rates. Preferred position of the sequence is in position 10 (mod 10). Remember that $n_{step} + 1$ is the defect that would be to the left of n_{step} if there are two defects present. The system also loops around at modulo 10 so after 11 steps we are in position 1 again.*

We observe that as our introduction rate increases, we have both an increase and decrease in defect lifetimes. We also observe that defect lifetimes do not seem to change much until the inverse of the introduction rate is equal or smaller than the defect lifetimes. The reason for this is trivial since this inverse corresponds with the expected time a defect would be introduced. If it is much larger than the defect lifetimes we will mostly see single defect motion.

We made an attempt at using the derived two-state theoretical lifetime which can be seen in figure 4.7. However when using it the approximation actually gets worse. The reason why this happens is not exactly clear yet, it might be caused by a misplaced index or there could still be a mistake. Nevertheless we show this result and see that the values are still close to what happens in the simulation.

The observed increase and decrease of defect lifetimes in figure 4.6 is caused by interactions when there are multiple defects present. While all defects

Version of October 10, 2022– Created October 10, 2022 - 09:08

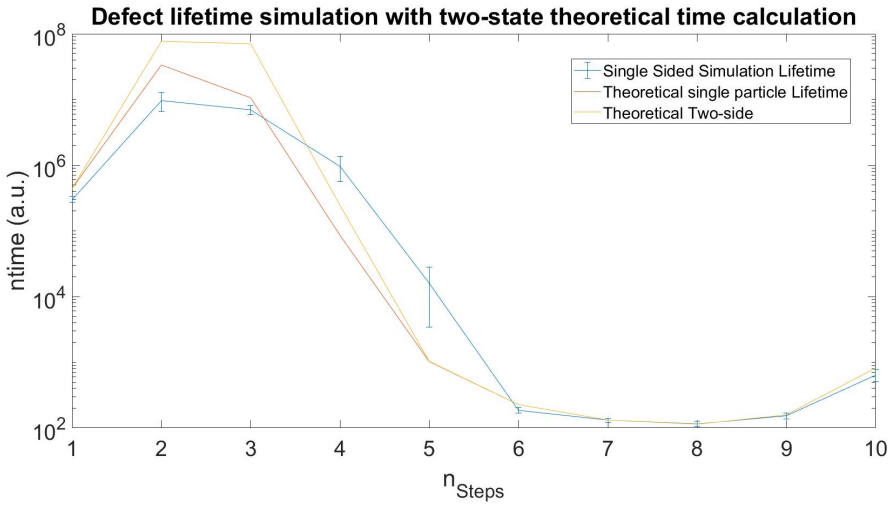

Figure 4.7: *Averaged defect lifetimes for an introduction rate of* 10^{-6} *plotted with the single defect lifetime and two-state defect lifetime calculations.*

are identical, we can categorize our defects in two groups, pushers and fallers, according to their behavior. The pusher defects are defects which have a shorter lifespan than the preceding defect. When they get introduced in the system they act as a moving wall preventing the previous defect from falling down too much. Fallers are the exact opposite, they are the defects that are being pushed. They have very long lifespans and tend to fall down to their start position.

When a faller is followed by a pusher their lifetimes change. The falling defect will be pushed out which will reduce its lifespan in the system. Meanwhile the pushing defect cannot freely move and has to wait for the falling defect to leave. All defects can act like fallers and pushers however their effectiveness can be different. For example the defect in position 1 is a faller however it very slightly is able to push the preceding defect in position 10 to decrease its lifetime.

We observe that the defects in position 10 and 1 greatly reduce their lifetimes as they get pushed. We also observe that the defects that push them in positions 2 to 5 increase their lifespan as they have to make sure these earlier defects leave. Interesting is how the role of the defect in position 2 switches from pushing to falling as the introduction rate gets larger. This mechanism of defects being present simultaneously and pushing each other also gives us the first insight into duplet and triplet states.

Their role might be more than just stabilising the ATP-remodeling. Instead they could be able to greatly reduce the twist defect motion timescale by reducing the defect lifetimes.

4.3 Nucleosome occupancy

In the final simulation we will be using longer DNA sequences and analyse its positioning around a nucleosome. We used periodic boundary conditions for a DNA sequence of length 350, the main sequence we used is listed in the appendix. When other sequences are used we will refer to them. Due to the periodic boundary conditions it becomes important to have a sequence length which is a multiple of 10. If not, we would have a discontinuity in the preferred position. Each simulation starts with the nucleosome in a random position. We then choose a direction of movement at random and attach the remodeler. The remodeler stays attached for a random Poisson distributed time with parameter t_{Twist}. It will run our first simulation as a sub-simulation but with the full energy landscape of the sequence. This energy landscape is pre-calculated for the entire sequence and both directions to speed up the simulation. The simulation will keep track of introduction time and leave time of the defects and their positions. Once the sub-simulation is over all the present defects are undone (see 3.4) and we return to the main simulation. The duration spent in each position of the exit side of the nucleosome and the new position is saved. The process then repeats by choosing a random direction and duration again. This continues until the end of the simulation t_{Total}. All simulations are done using the following parameters unless specified:

Simulation Parameters	
t_{Total}	10^{11}
t_{Twist}	10^{8}
Intro Rate	10^{-6}
Damp Factor	0.8
Direction	50% for left and right
Passable Chromatin	False
Variable Introduction	False

We keep track of the duration spent in each position to calculate the fraction of time spent per location. By doing this we can get an estimate for the occupancy probability along the sequence. We have compared these occupancies with the Boltzmann distribution obtained from the *Nucleosome Wrapping Energy* for the different positions as seen in Figure 4.3. We

also performed parameter sweeps to see how they affect the occupancy. Finally we made our simulation only move in a single direction which allowed us to observe a quick ejection of defects as seen in [1].

Our first result in figure 4.8 compares the occupancy of the simulation

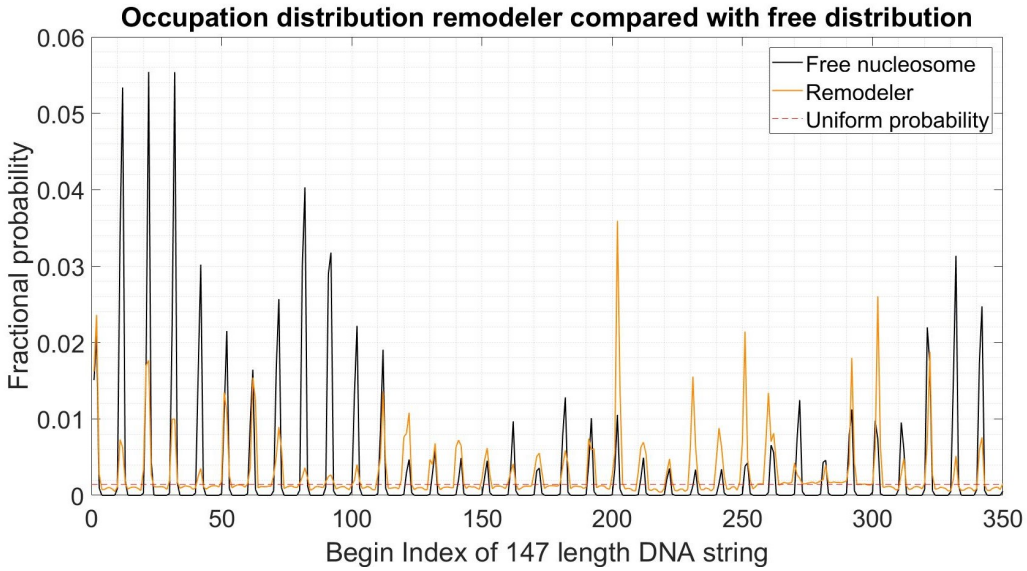

Figure 4.8: *Occupancy of the remodeler simulation (black) compared with the Boltzmann distribution (orange) and the uniform probability (red). The Boltzmann distribution was calculated with a damped wrapping energy such that the energy hill is around the order of 10 kT. This corresponds with a damp factor of 0.18. For the remodeler simulation we used a damp factor of 0.8 and* $t_{Total} = 10^{12}$.

with that of the free nucleosome. We observe that both have very strong peaks which are located in multiples of 10 steps. The free nucleosome shows peaks that are more strongly grouped together as we can see more towards the left, while the remodeler peaks are more distributed at random. These peaks correspond with the preferred positions of the nucleosome and the way they are grouped comes from how sensitive they are to the global energy landscape. For the free nucleosome, the partition function of the distribution takes into account the entire landscape. Due to this, all positions feel the effect of the low energy position which attracts the nucleosome. For the remodeler simulation, the peaks are caused by defects having to go up an energy hill. Having the remodeler present prevents moving back to the lower energy configuration after taking a step away from it. While the defects still have to fight the energy hill, if this

barrier is much larger than the energy difference between preferred positions, then this will dominate the dynamics. The former is given by the energy difference of the preferred and least preferred position of the nucleosome (while staying at the same SHL's of course). The latter is given by the energy difference of two preferred positions (while moving a single SHL in total). We can see now that it makes sense that these energy hills dominate the dynamics at play here. Due to this the occupancy in the remodeler simulation is dominated by the local energies.

One may ask themselves what where to happen if we lower these energy hills. Would the globally preferred position have a significant effect on the peaks or will the local energies continue to be the main influence. We decided to run another simulation with a lower damp factor to observe how the landscape changes.

In figure 4.9 we see that the strong initial peaks we observed before have

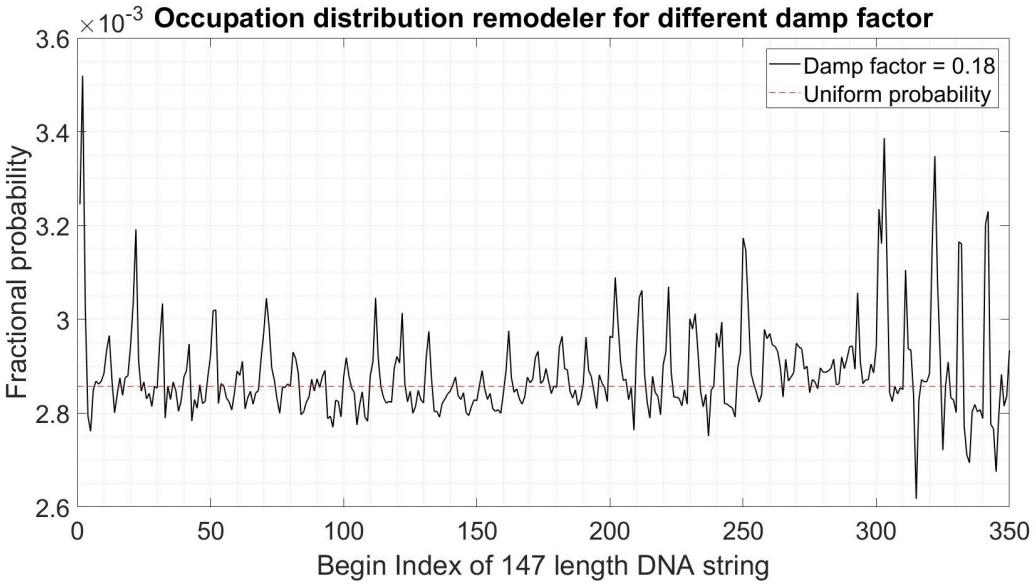

Figure 4.9: *Occupancy of the heavily damped remodeler. The occupancy now almost approaches a uniform distribution. We used a damp factor of 0.18 and* $t_{Total} = 10^{13}$.

significantly decreased. It also seems which peaks stand out are slightly different as the peak at position 200 is now no longer the biggest. This unequal peak reduction could be a real effect due to the damp factor changing the energy costs of moving the sequence but not that of the defect itself.

It could however also be attributed to statistical noise. We also observe that there does not seem to be a strong rise or fall in the occupancy rates which we would expect to see for the preferred global position. A possible explanation for this could be that the forced single-direction movement by the remodeler can easily overcome the rise and fall in energy towards the globally preferred position. Considering the defects are able to overcome these huge energy barriers in between preferred positions this should not be too surprising.

Next we would like to see how else we could change the occupancy if we adjust our parameters. We already saw the effect of a damp factor and we have added another plot with more factors in appendix A.2. We can start by adjusting the total duration of the simulation.

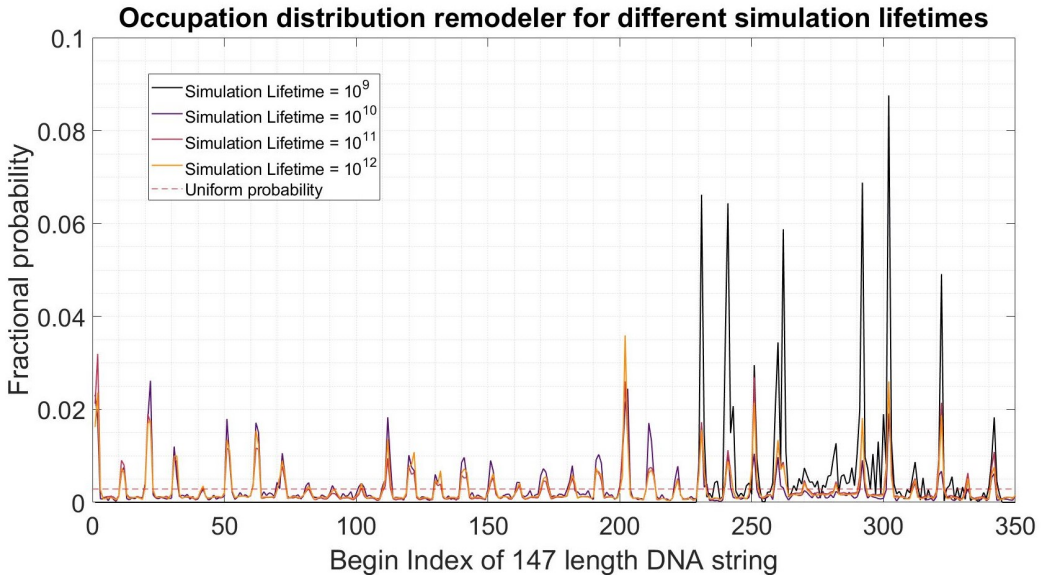

Figure 4.10: *Occupancy parameter sweep for* t_{Total}.

We observe in figure 4.10 that the shortest simulation lifetime is only present on the right side of the sequence. This is caused due to insufficient sampling time for our Markov process. Our remodeler stays stuck in these positions and does not have enough time to explore the entire phase space. The longer simulations seem to sample the entire space and have their peaks closer together. For most peaks the two longest simulations overlap very well however there are a few outliers. We can estimate that for this duration these longest simulations sample on average every point 100

and 1000 times. Sampling every point 100 times is accurate enough for our purposes since we are mainly interested in seeing how our peaks change.

Another very interesting parameter to change is the Intro rate. In Section 4.2 we saw how changing this has a significant impact on the lifetimes of the defect present in the system. Because of this we would assume to see a similar effect here where defects are able to push each other out more quickly.

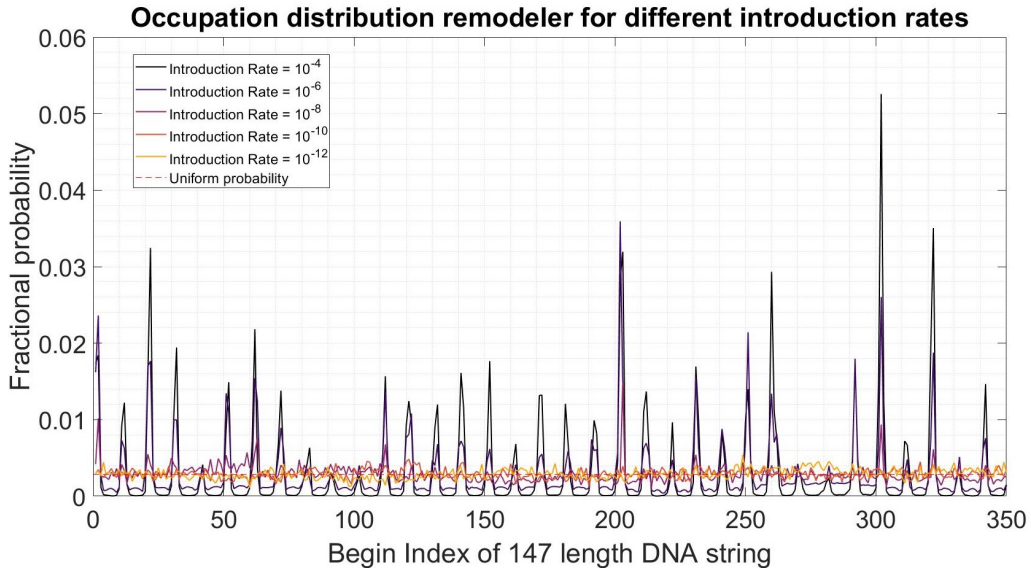

Figure 4.11: *Occupancy parameter sweep for the intro rate. To simulate the system when changing the intro rate we have to make sure we multiply t_{Twist} and t_{Total} by a factor of 10^2, 10^4 and 10^6 when we have an intro rate of 10^{-8}, 10^{-10} and 10^{-12} respectively. This is due to the time to introduce a defect is being increased here. We do not do this for the high intro rate since the defect movement still takes a lot of time and we receive a better statistical sample if it runs longer.*

We observe in figure 4.11 that for large intro rates we get strong peaks, while the really low intro rates are almost close to being uniform and more comparable with statistical noise. This might seem confusing at first if we compare it with our results from Section 4.2. There we observed that for high intro rates we do not observe much time difference anymore while for really low intro rates it takes much longer to move out of the preferred position. Due to this we may expect to not see much difference between positions for high intro rates. While this change in defect lifetime still happens, there are two reasons we do not see this here. First, we have to wait for a defect to be introduced. Since we need to wait in every position for

the defect to be introduced we would get the sum of the lifetime in the nucleosome and the time until the defects gets introduced. If this time to introduce a defect is much larger than the lifetime near the preferred position. We would have that it spends almost the same amount of time everywhere which then gives an equal density distribution for the nucleosome along the DNA. Secondly we do not measure the full defect lifetime but only its lifetime after the other present defect has left the system. Suppose we have a high intro rate and a defect moving away from the preferred position followed by some defects. We would still see it take more time before the first defect leave when compared to the others. However the moment this defect leaves we do not need to wait for another defect to be introduced and we are already very likely to find the next defect close to the end. This defect then leaves again quickly and the following defects are also more likely to fall of soon. So even though we may have many defects present, we will observe a quick succession of defects leaving the system. This is already a small teaser for our main result which we obtain from the occupancy plots from a single direction in the next section.

There are a few more occupancy plots that we could make, for example for t_{Twist}, different directions, different DNA, Variable introduction and Passable Chromatin with different barriers. The different direction occupancy plot is saved for the next section while the other plots and a short discussion can be found in Appendix A.2.

4.4 Duplet/Triplet state ejection

Finally we come to our main result, the simultaneous ejection of multiple defects. We analysed the effect of having multiple defects present at the same time for the defect lifetime. We theorised and found that a secondary defect can push out a stubborn defect that does not want to leave. However doing this we mostly focused on the fate of the first defect. Now instead we will be looking at what happens to the secondary defect present in the system.

First we tried to calculate the time delay between defects leaving the nucleosome. This was done for duplet and triplet states which we did in our first simulation. We expected to see a disparity between the difference in leaving time for these defects and the defect lifetimes. If we see this difference being much smaller than the expected introduction time, we can be certain the secondary defect was pushing out our primary defect and

Version of October 10, 2022– Created October 10, 2022 - 09:08

leaving soon after it as well.

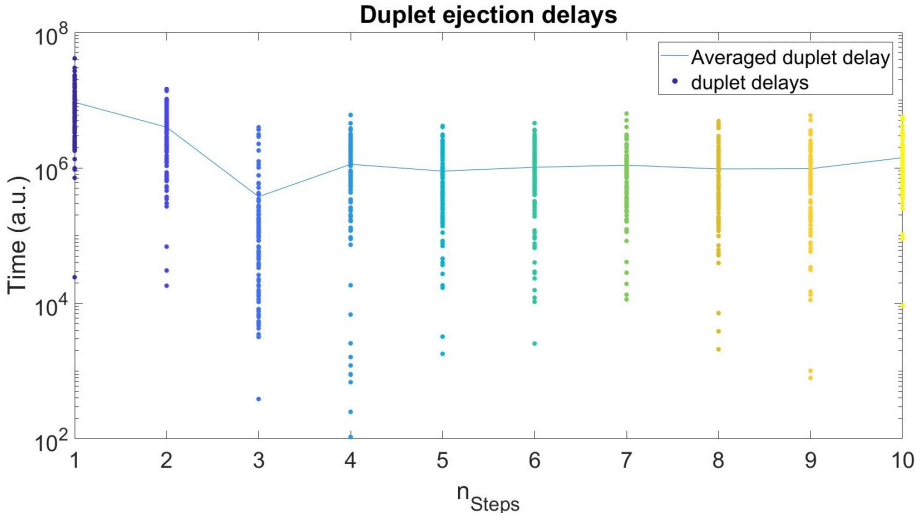

Figure 4.12: *Ejection delays for the Nucleosome Transition simulation with intro rate* 10^{-6} *compared with the simulation lifetimes. Our x-axis is the index of our first defect. This means the duplet delay is how long it would take for the next defect to leave the system.*

If we were to have a system of defects that never interact with each other and have short lifetimes, we would expect the delay between defect ejections to be equal to the expected time for a defect introduction. However when defects interact or are present for long durations, this will no longer be the case and give an increase or decrease in delays. We observe in figure 4.12 that the averaged duplet delay stays near the expected time of an introduction for positions 4 to 10. We also observe a slight peak at the start and a slight valley on position 3. Since these delays look at how long it takes before the next defect leaves we should take into account the lifetime of the following defect. This is done since the delay is dominated by either the defect lifetime or the expected time for an introduction. Because for position 2 and 3 this lifetime is longer than the expected time for an introduction, we see these peaks in the delay which are caused by defect having a hard time to leave the system.

We would however like to detect a duplet state using 4.12. As a quick refresher, these duplet states would consist of two defects being present together. The first introduced defect would tend to fall down, while the second defect would tend to push and act as a wall. If this were to be successful we would observe a very short delay between the ejection of two

defects. We do not seem to observe a duplet state for position 1, its delay is almost exactly the same as for the next defects lifetime which means the second defect just fell down. For position 2 we see a delay that is slightly lower than the next lifetime which could be the first duplet state we observe. However this delay is still higher than the expected time for an introduction so there is no quick successions of defects leaving.

In position 3 it becomes very clear that the delay is much shorter than that of the expected time of an introduction. We also observe for position 3, that when we scatter the delays they tend to be much lower than the rest of the delays. These lower delays indicate that the second defect left the system very quickly after the first defect did. This quick succesion of ejections would then serve as an indication that we have a duplet state for the defects on position 3 and 4. There are also scatters for position 4 which seem to indicate something similar but this could be attributed to randomness. If it is not noise but instead an actual observation there are two explanations. Either there was no duplet for position 3 and 4 but instead 4 and 5 formed a duplet state. Or the duplet state of 3 and 4 became a triplet state when another defect got introduced. However it seems that both of these occurrences are quite rare if they even happen at all.

This plot seems to indicate to us that there is a slight probability for a duplet state to be present. For position 2 which gets pushed by position 3, but also a much more probable duplet state for position 3 which gets pushed by 4. These duplet states cause the delays to be smaller than the expected intro time or the next defect lifetime. Because of this we should be able to observe a quick succession of defects leaving the system.

Another way how we can find these duplet states leaving is through the occupancy plots of the second simulation. If we were to only allow for a single direction and run our simulation we could see a duplet state as a large peak followed by a smaller valley. This valley should be located to the left or right depending on the direction the nucleosome is travelling in.

In the occupancy plot of figure 4.13 we see our strong peaks like before but now we also see a slight valley to the left of these peaks. These valleys could in theory be caused by the defect having an extremely short lifetime here. However if we were to look at defect lifetime plots such as figure 4.6, we find that there are 3 to 4 positions on which the lifetime of the defect is extremely short while we only see very narrow valleys of 1 to 2 base pair

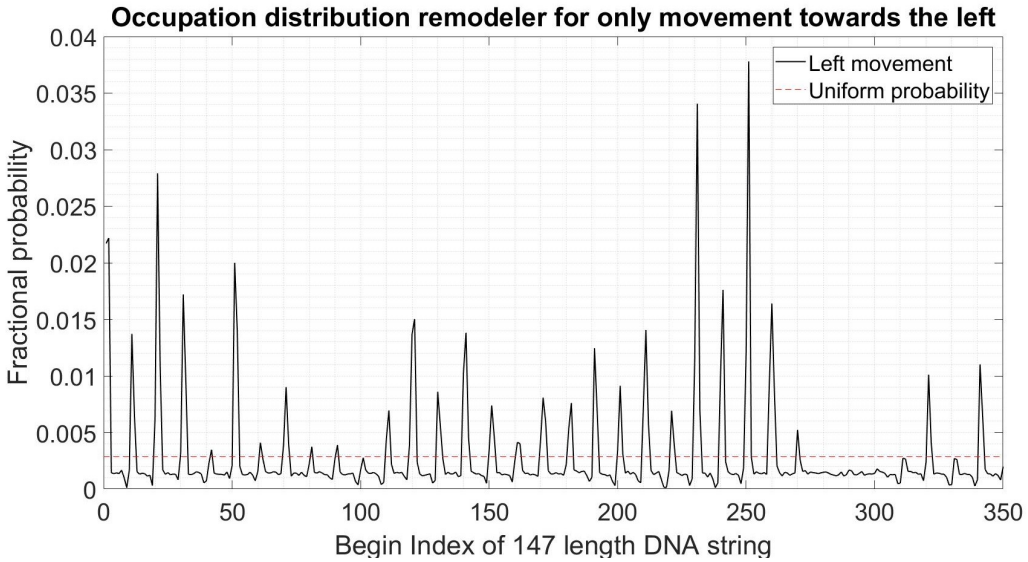

Figure 4.13: *Occupancy of single direction nucleosome. The remodeler is only able to connect for movement to the left. See appendix A.2 for the movement to the right.*

steps. The position of these short defect lifetime plots would also be more centered instead of so close to the peak. Instead we theorise these valleys are caused by duplet or triplet states. The long lifetime of the defects at the peak positions cause a build-up of other defects. These defects can then help push our primary defect out of the system. After this they have a very high chance of falling of quickly themselves too due to either another defect that has built up or due to them likely being near the exit side of the nucleosome already.

Chapter 5

Discussion

The discovery of these duplet states is quite interesting as they might be able to give more insight into the working of single base pair reposition of nucleosomes. While the simulation is a very basic model it is able to show how these duplets are able to reduce the movement time scales. Our attempt at analytically calculating the lifetimes of the defects was not successful. While the idea might work we would have to define the triplet state dynamics as well. On top of this the probability calculations of having duplet states present would need to be changed to accommodate. We also found that the distribution of the nucleosome with remodeler feels very local effects. This local sensitivity means a remodeler would also be able to more easily traverse a higher energy region which the free nucleosome would normally not be able to. We will use the rest of this chapter as an outlook as to how this project may be expanded in the future to improve the results we obtained.

5.1 Future development of the code

While the code works very well, there are still some improvements to be made and features that can be added for future analysis.

Double-sided simulation

A big issue in the current simulation is that we only simulate on one side of the remodeler. This was done due to an earlier computational limitation. However at this point the code is much better optimized and could

easily support this feature. Implementing this would be the next step to be taken since it would allow for two major improvements.

First of all, once our remodeler releases we are able to continue simulating our model and see where our defects actually ends up instead of undoing the defect.

However secondly and more importantly, there is a much better understanding of the time delay for the defects escaping on both ends. For now it is assumed that defects on the other end of the remodeler escape much more easily since for SHL = -2 they only have to go through 4 bindings instead of 9 bindings on the other side. Due to the exponential suppression of going uphill on the energy landscape it is a fair assumption that the other side escapes much more quickly. In fact this is even seen in the theoretical calculation of the expected lifetime of the twist defect. Calculating the ratio between having 4 binding sites or 9 binding sites using equation 2.12 and the rates of our simulation, we find it to be slower by an order of 100.

Nevertheless it is still an important fact that needs to be considered and implementing it would allow us to better model the observations in the experiment and calculate an actual time delay between the escapes on both ends and the undoing of earlier defects after the remodeler releases.

Rest periods and secondary movement method

The current iteration of the code constantly tries to re-attach the remodeler. This is quite unrealistic since it can take a while before this happens again. There is some code written to allow for these rest periods it is not currently in use since the main goal of these rest periods is not implemented either.

During these rest periods it would be possible for the nucleosome to undergo different repositioning such as through spontaneous twists or loop formation [24]. This was not however the main goal so we left this open for possible future development to combine repositioning methods.

Implementation of overtwist

While most of the code is written to use both under- and overtwist, this is not the case for the final nucleosome repositioning sequence for two reasons. First of all, the energy values are much higher for the overtwist

defects which exaggerate what the actual time durations would be. Secondly, the overtwist makes all movement work in the other way and this would require very carefully keeping track of every place where we take displacement into account. This is done almost constantly and would require a lot of rewriting the code. Since a simple mistake is quickly made but hard to find and there is not much different dynamics to see for it, we decided to stick to undertwists and analyse these more deeply instead.

Non-periodic DNA sequence and linker DNA

Right now the simulation uses periodic boundaries, the nucleosome can move from one side to the other and it wraps using a periodic sequence. We would like to give an option to prevent this from happening by putting either a hard or soft boundary on the nucleosome position to stay in one spot and adding linker DNA on both ends. This would also allow us to study how different sequences of linker DNA affect our nucleosome repositioning.

Tracking of position

We need to improve the method the position is being tracked. Currently we just have a nucleosome position which corresponds with the first base pair position. We use this position then to track the location of the nucleosome and to update the energy barriers felt by the defect. In reality we should split these from each other and introduce a third position. We should use the original position as the defect energy landscape index. We then keep track of the position at the beginning and the end of the sequence such that we now have 3 positions in total. This does require the usage of the double-sided simulation though. Doing this however allows us to much better keep track of where and when the nucleosome moves. Currently the tracking of the position is just for one side. When using the reversed direction we are looking at the other end however we neglect this and just try to focus on how the movement looks like and how the occupancy changes.

Energy exaggeration

A big issue with the current simulation is that some of these energy values get blown out of proportion due to the nature of the rigid body model. While this is fine to see actual effects happening due to the nature of the system, it does not give accurate predictions for what we would observe.

Version of October 10, 2022– Created October 10, 2022 - 09:08

Instead we would have to change our values such as introduction rate, total duration etc. to sometimes unrealistic values to have a more realistic proportionality of parameters. For example the introduction rate when compared to the actual introduction landscape is very different. This may cause some of the observations that we see in this project to be unrealistic. Due to this a deeper and more accurate analysis should be done to confirm these results.

Remodeler release

Currently when the remodeler is released, all the present defects in the systems are undone. The logic behind it is that if a defect is still stuck by the time the remodeler is released, it is very likely that this was a defect that had to go up an energy hill. Due to the remodeler releasing it would make sense it now just goes down hill and falls off on the other end. This is not always the case though, other defects can be present to preventing this from happening or a defect that goes down hill might just happen to be present. Due to this the remodeler simulation is not as accurate as we would hope it to be and this should be corrected in the future. (see also double-sided simulation for how)

Remodeler position

We used SHL=-2 as the remodeler position throughout this simulation. While this corresponds with the literature [6, 12], we also introduce the defect on this position which is not what happens. Instead the defect should be introduced on SHL=-1. In the future the code could either be rewritten such that we position the defect one SHL away from the remodeler position or we use SHL=-1 and use this parameter as the defect introduction position.

5.2 Semi-Analytical lifetime model

To continue improving the analytical model into a more accurate representation we should consider expanding the range of situations that we may consider. For example we currently neglect the potential presence of having more than two defects present at the same time. This is quite unrealistic as we have observed defects with an initial lifetime of 10^8 which gets reduced when we have an introduction rate of 10^{-6}. This reduction could be due to having multiple defects build up which should speed up the

ejection process even more. Adding this effect we would first of all have to create a calculation to have three simultaneous defects present. Then we would need to adjust the probabilities of having two defects present since this probability will start going down after a long duration.

Once we have new probabilities for these scenarios, we have to repeat all motion of particles and find simple models. These simple models should be able to give a recursive method to get an estimate for the expected time to take a step to the right.

Writing this all out by hand would be a complete nightmare, even just going a few steps for two particles gives giant equations. If no simple recursive models to reduce the complexity of the motion can be found there is still another solution. We could simply let an algorithm go through as many scenarios as possible. This way it can explore paths with different probabilities and durations before a step to the right is taken. Once a certain threshold of remaining probability is reached (ex. 1% of paths have not yet resolved) we could terminate the path exploration at this point and use it as a lower bound.

While this sadly loses the beauty of having an actual analytical expression, an algorithm like this should be able to quickly go through these paths as long as it recombines converging paths. Using this in combination with a bottom-up approach of calculating the lifetimes allows us to use the recursion in the system to our advantage.

Chapter 6

Conclusion

We successfully implemented the RBP model as described in [5, 11]. We defined a simple super-helical geometry for the DNA and used positional preferences for the base pair steps to create sequences. By then defining a defected geometry with an extra base pair we were able to introduce a defect in the DNA. We made this for K10 and K20 undertwists and overlapped the defected and undefected geometry to verify. These defected geometries are used with the RBP model to calculate the energy costs of having a defect present somewhere. We use unbinding costs for the K20 defect around the order of 12kT and find that our energy barriers are around the order of 3kT. These defect energies were used to create an energy landscape, this was repeated for multiple steps such that we get landscapes which include all intermediate steps between preferred positions. For a sequence whose base pairs have equal probabilities everywhere we did not observe any difference in the landscapes, but for a sequence with positional preference we found a rise and fall in the landscape. This rise and fall in energy was compared with full nucleosome base pair steps and overlapped very well. We then made a simulation to analyse the lifetime of defects present around the nucleosome. This was done by adding a defect introducing remodeler at SHL=-2 [6] and evolving the system with a Gillespie algorithm. Increasing the introduction rate allowed us to see the effect of having multiple defects present simultaneously. These multiple defects were able to push defects near the preferred position and reduce their lifetime drastically. We also observe that simulating the other side of the nucleosome is mostly irrelevant for the dynamics on our side. We continued with making another simulation which made a nucleosome move

over a sequence. Calculating the occupancy of the nucleosome, we found that the nucleosome has very strong peaks at the preferred position in the sequence. We found that that these peaks of the remodeler are distributed at random unlike the peaks of the Boltzmann distribution of a free nucleosome which are grouped. This means our remodeler mostly feels local energy values. Finally we observed duplet state ejections for both of our simulations. In the first simulation we found a reduction in the delay between defect ejection near $n_{Steps} = 3$. In our second simulation when only moving in one direction we observe an extremely low occupancy right after the nucleosome passes a peak which is causes by a quick succession of defect ejections.

We found that these multiple defects present at the same time and ejection of duplet states could give a possible explanation for the observations of Sabantsev et al. [1]. One possible next step would be to use a better model [14] to repeat a similar simulation. The question would be if these duplet states are still present or are just an artifact of the exaggerated energy values.

While this RBP model is a simple toy model and it tends to exaggerate energy values, we are still able to recover some interesting properties of the nucleosome. We attempted to mitigate this issue by damping the energy cost of the undefected DNA. While this damping allows us to adjust the energy barrier between preferred positions, it is still unclear if this damp effect gives realistic dynamics. We have also found some interesting properties such as the local energy dependence of the remodeler occupancy and the presence and ejection of duplet states. This toy model serves well in finding and discovering these potential dynamics of the system and can thus serve as a stepping stone for more detailed and advanced simulations.

Version of October 10, 2022– Created October 10, 2022 - 09:08

Bibliography

[1] Anton Sabantsev, Robert F. Levendosky, Xiaowei Zhuang, Gregory D. Bowman, and Sebastian Deindl. Direct observation of coordinated DNA movements on the nucleosome during chromatin remodelling. *Nature Communications*, 10(1):1720, 2019-12.

[2] Amber Cutter and Jeffrey J. Hayes. A brief review of nucleosome structure. *FEBS letters*, 589(20):2914–2922, 2015-10-07.

[3] Yuri Moshkin. Eukaryotic gene expression: a hurdle-race through chromatin. *Adjacent Government*, 10 2015.

[4] Jiannan Guo and D. Price. RNA polymerase II transcription elongation control. *Chemical reviews*, 2013.

[5] I. M. Kuliç and H. Schiessel. Chromatin dynamics: Nucleosomes go mobile through twist defects. *Physical Review Letters*, 91(14):148103, 2003-10-01.

[6] Giovanni B. Brandani and Shoji Takada. Chromatin remodelers couple inchworm motion with twist-defect formation to slide nucleosomal DNA. *PLOS Computational Biology*, 14(11):e1006512, 2018-11-05.

[7] Giovanni B Brandani, Toru Niina, Cheng Tan, and Shoji Takada. DNA sliding in nucleosomes via twist defect propagation revealed by molecular simulations. *Nucleic Acids Research*, 46(6):2788–2801, 2018-04-06.

[8] Curt A. Davey, David F. Sargent, Karolin Luger, Armin W. Maeder, and Timothy J. Richmond. Solvent mediated interactions in the structure of the nucleosome core particle at 1.9 a resolution. *Journal of Molecular Biology*, 319(5):1097–1113, 2002-06-21.

[9] K. Luger, A. W. Mäder, R. K. Richmond, D. F. Sargent, and T. J. Richmond. Crystal structure of the nucleosome core particle at 2.8 a resolution. *Nature*, 389(6648):251–260, 1997-09-18.

[10] Wilma Olson, Andrey Gorin, Xiang-Jun Lu, Lynette Hock, and Victor Zhurkin. DNA sequence-dependent deformability deduced from protein-DNA crystal complexes. *Proceedings of the National Academy of Sciences of the United States of America*, 95:11163–8, 1998-10-01.

[11] Helmut Schiessel. *Biophysics for Beginners: a Journey through the Cell Nucleus*. 2014-01-01.

[12] Gregory D. Bowman. Mechanisms of ATP-dependent nucleosome sliding. *Current opinion in structural biology*, 20(1):73–81, 2010-02.

[13] Philippe A. Bopp, Jörn B. Buhn, Holger A. Maier, and Manfred J. Hampe. Scope and limits of molecular simulations. *Chemical Engineering Communications*, 195(11):1437–1456, 2008.

[14] Behrouz Eslami-Mossallam, Raoul D. Schram, Marco Tompitak, John van Noort, and Helmut Schiessel. Multiplexing genetic and nucleosome positioning codes: A computational approach. *PLOS ONE*, 11(6):e0156905, 2016-06-07.

[15] Noam Kaplan, Irene K. Moore, Yvonne Fondufe-Mittendorf, Andrea J. Gossett, Desiree Tillo, Yair Field, Emily M. LeProust, Timothy R. Hughes, Jason D. Lieb, Jonathan Widom, and Eran Segal. The DNA-encoded nucleosome organization of a eukaryotic genome. *Nature*, 458(7236):362–366, 2009-03-19.

[16] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[17] Arman Fathizadeh, Azim Besya, Mohammad Reza Ejtehadi, and Helmut Schiessel. Rigid-body molecular dynamics of dna inside a nucleosome. *The European physical journal. E, Soft matter*, 36:9835, 03 2013.

[18] Andres E Leschziner. Electron microscopy studies of nucleosome remodelers. *Current opinion in structural biology*, 21(6):709–718, 2011-12.

[19] Anastasiia L. Sivkina, Maria G. Karlova, Maria E. Valieva, Laura L. McCullough, Timothy Formosa, Alexey K. Shaytan, Alexey V. Feofanov, Mikhail P. Kirpichnikov, Olga S. Sokolova, and Vasily M. Studitsky. Electron microscopy analysis of ATP-independent nucleo-

some unfolding by FACT. *Communications Biology*, 5(1):1–9, 2022-01-10. Publisher: Nature Publishing Group.

[20] Geert Meersseman, Sari Pennings, and E.Morton Bradbury. Chromatosome positioning on assembled long chromatin: Linker histones affect nucleosome placement on 5 s rdna. *Journal of Molecular Biology*, 220(1):89–100, 1991.

[21] Sreekala Balasubramanian, Fei Xu, and Wilma K. Olson. DNA sequence-directed organization of chromatin: Structure-based computational analysis of nucleosome-binding sequences. *Biophysical Journal*, 96(6):2245–2260, 2009-03-18.

[22] Dario Camuffo. Chapter 9 - consequences of the maxwell boltzmann distribution. In Dario Camuffo, editor, *Microclimate for Cultural Heritage (Second Edition)*, pages 347–366. Elsevier, Boston, second edition edition, 2014.

[23] Hidetoshi Kono and Hisashi Ishida. Nucleosome unwrapping and unstacking. *Current Opinion in Structural Biology*, 64:119–125, 2020. Biophysical and Computational Methods, Cryo EM.

[24] I.M. Kuliç and H. Schiessel. Nucleosome repositioning via loop formation. *Biophysical Journal*, 84(5):3197–3211, 2003-05.

Chapter A

Appendix

A.1 Validity testing

A.1.1 base pair probabilities

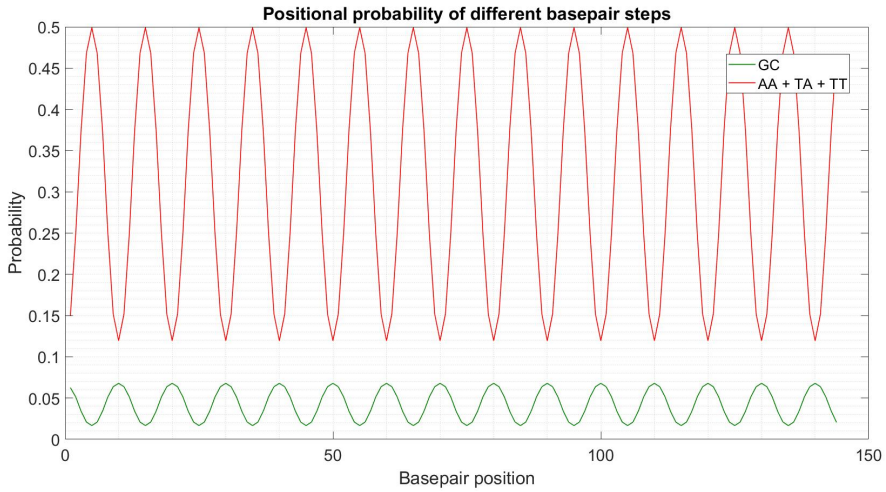

Figure A.1: *Positional probability for some of the base pairs to be found. We find GC has a higher probability to occur near the binding site while AA, TA and TT are commonly found in between the sites.*

Using the RBP energy model we recreated the positional probabilities of the base pair steps. This is done through multiplying transfer matrices. For a deeper description of how this technique works see [11]. We end up recovering the probabilities as seen in other works [14].

A.1.2 Defect geometry

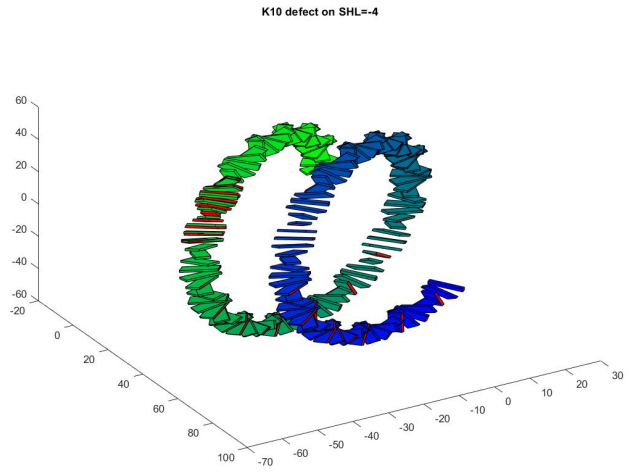

Figure A.2: *Regular DNA (green) is overlapped with the K10 defect DNA (red). They perfectly overlap everywhere except for the SHL containing the defect.*

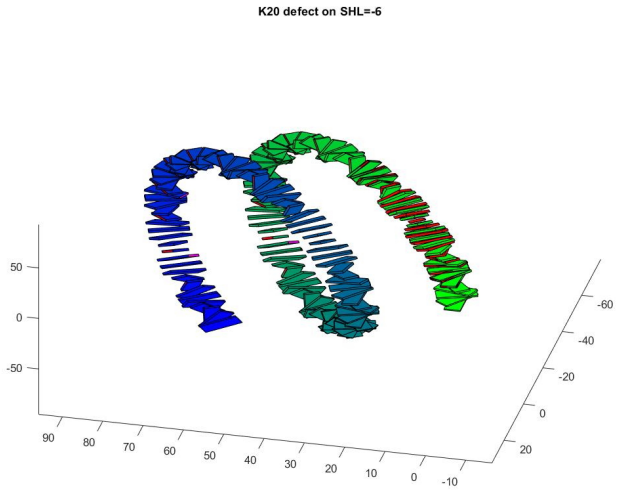

Figure A.3: *Regular DNA (green) is overlapped with the K20 defect DNA (red). They perfectly overlap everywhere except for the SHL containing the defect.*

We find that plotting the undefected geometry we recover the superhelical structure we desire to have. We also find that plotting the defected geometry over it that we see them perfectly overlap except for the defected SHL. See also Fig. 4) d from [9] and 7) c from [7]

A.1.3 Energy plots

Additional Energy landscapes

Here we show some additional energy landscapes that we made.

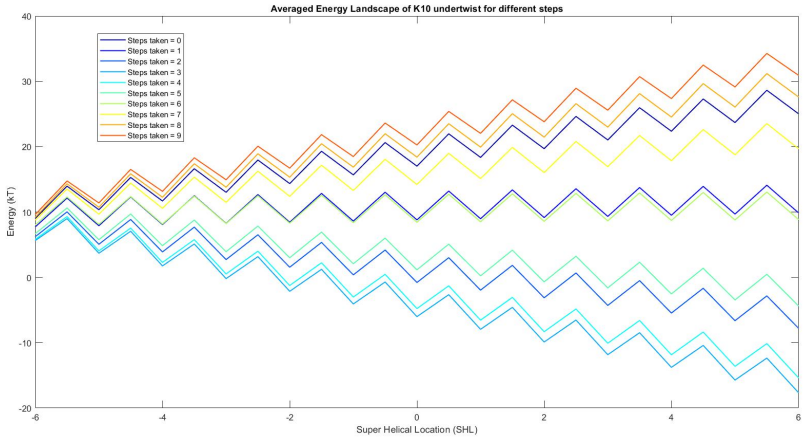

Figure A.4: *Averaged energy landscape of a positional dependent sequence.*

We can see how the saw-tooth pattern from earlier energy landscapes is recovered in this plot. The energy however simply goes up or down a hill at the same time. The same can be repeated for a damp factor of 0.18

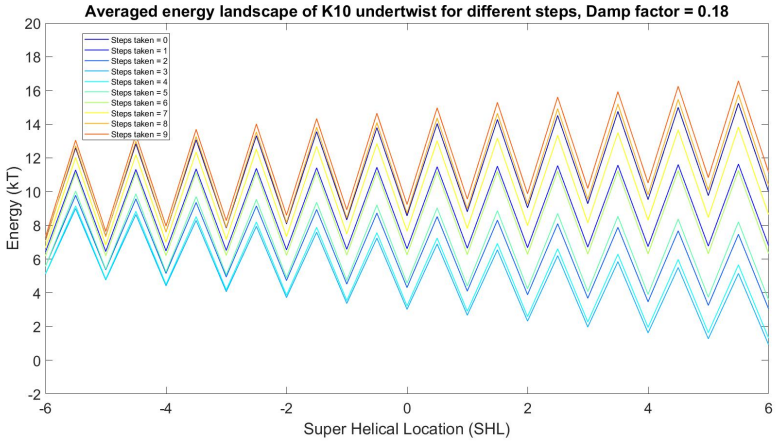

Figure A.5: *Averaged energy landscape for a damp factor of 0.18. We see that the slight rise and fall in energy is still present in the landscape.*

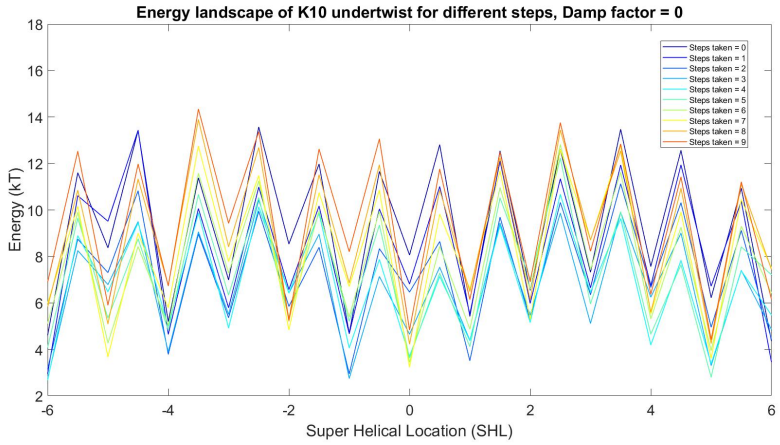

Figure A.6: *Energy landscape with a damp factor of 0, the rise and fall in energy is no longer present.*

When using a damp factor of 0, we found that we are able to suppress the rise and fall in energy. This was done without losing any of the defect energy. Doing this allowed us to recover a flat energy landscape. However in reality we did not suppress every contribution which can be seen if we average it out.

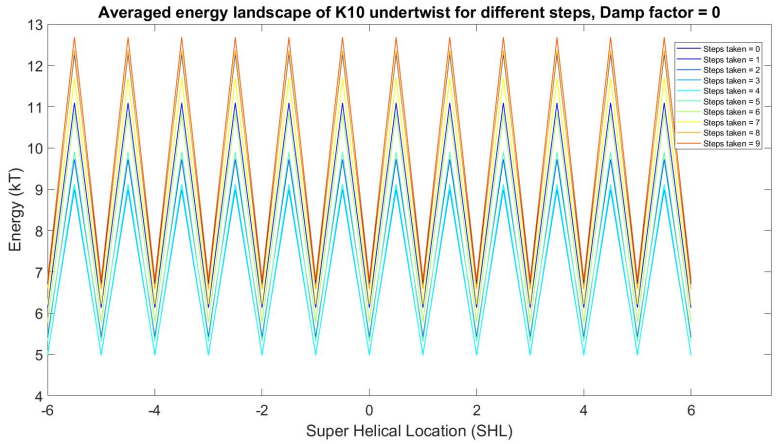

Figure A.7: *Averaged energy landscape of a positional dependent sequence with a damp factor of 0. We are left with a slight shift along the vertical axis.*

There is a positional energy contribution due to our defect itself having

base pairs that have moved out of their preferred position. So this causes a slight change in the energy values between the different steps taken. This is not an issue however for the first simulation since we do not care about the energy levels between two different energy landscapes, we only look at the energy difference between two neighbouring states in a single landscape. We found that we recovered a perfect flat and periodic landscape which is what we hoped to gain from our damp factor.

We could also plot the energy landscape for positional independent sequences. These sequences have equal probability for all base pairs at any position.

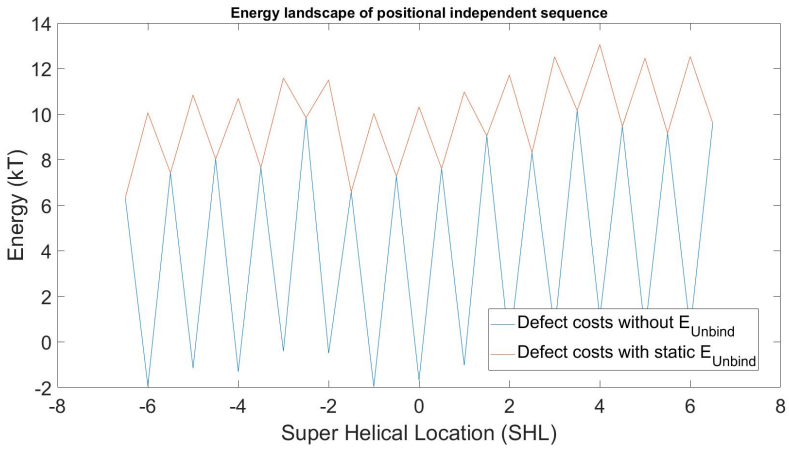

Figure A.8: *Energy landscape of undertwist for a positional independent sequence.*

If we were to now average this out, we would expect not to see a rise or fall in energy as we no longer have a preferred position present. Taking the average over 100 sequences in figure A.9 then generates a perfect flat landscape which is what we were expecting to see. Changing n_{Steps} is not important for this plot as there is no positional dependence anyways.

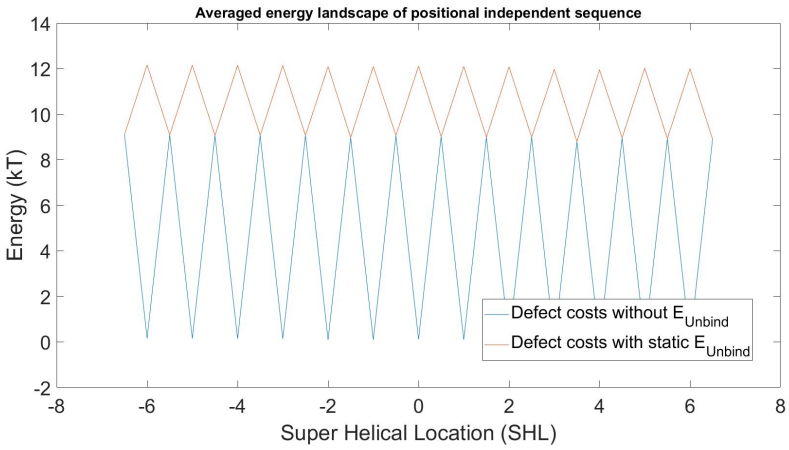

Figure A.9: *Averaged energy landscape of an undertwist for a positional independent sequence.*

A.1.4 Occupancy tests

We can try to test the occupancy plots by verifying some simple results. First of all we can try to use an all AAA sequence to see if we recover a flat occupancy landscape.

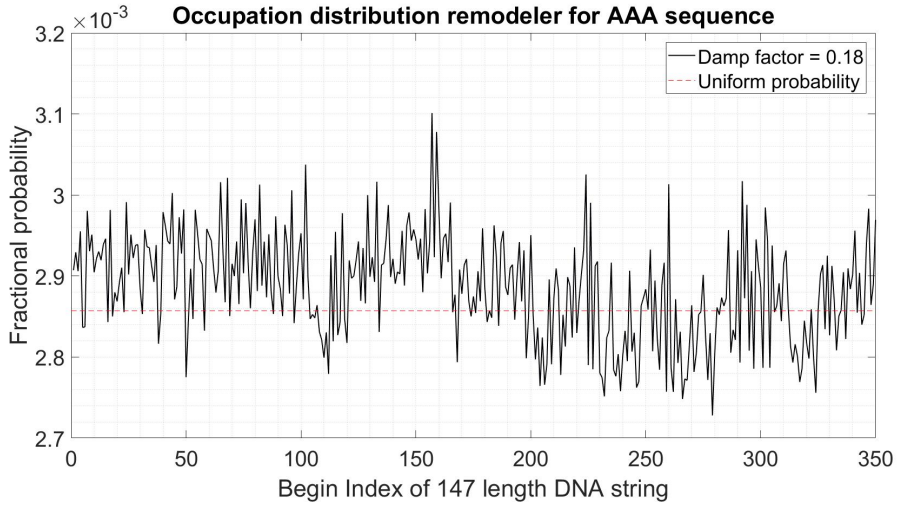

Figure A.10: *Occupancy for an all AAA DNA sequence. We find that the occupancy is very close to flat except for statistical noise.*

A.2 Additional figures and plots

A.2.1 Occupancy plots

Here we shall show some additional occupancy plots that we have created.

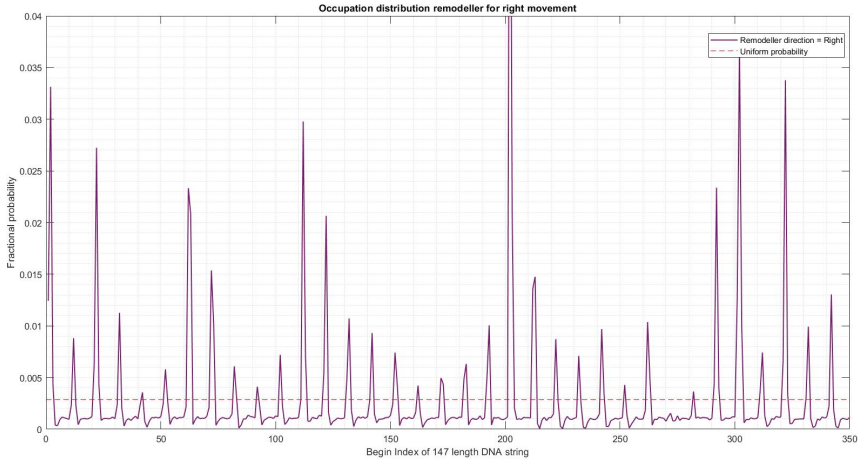

Figure A.11: *Occupancy plot for single direction nucleosome. The valleys have now switched to the other side of the peak due to us now only moving to the right.*

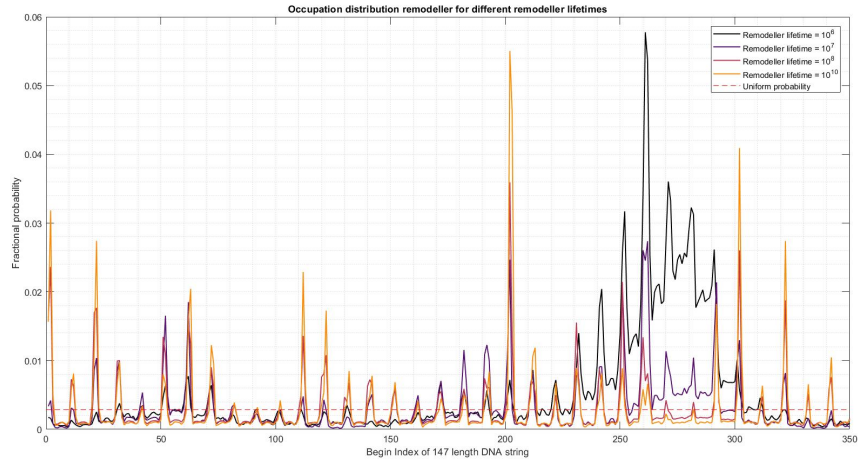

Figure A.12: *Parameter sweep of* t_{Twist}

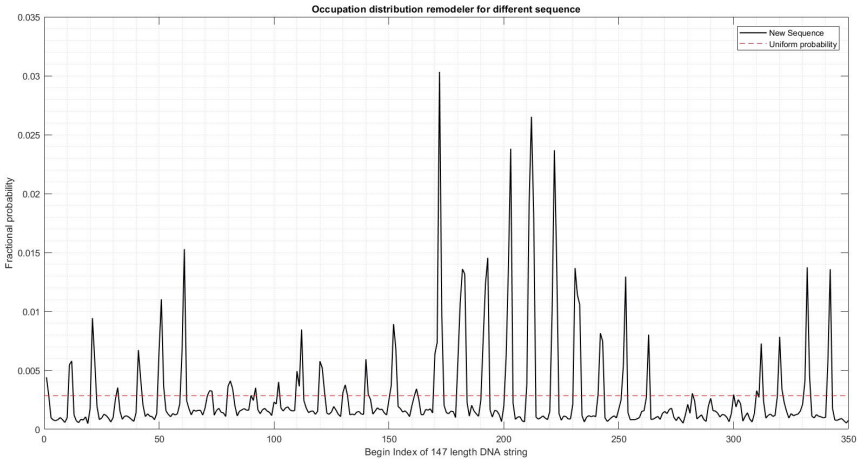

Figure A.13: *Occupancy for a new positional dependant sequence. We again observe peaks distributed at random*

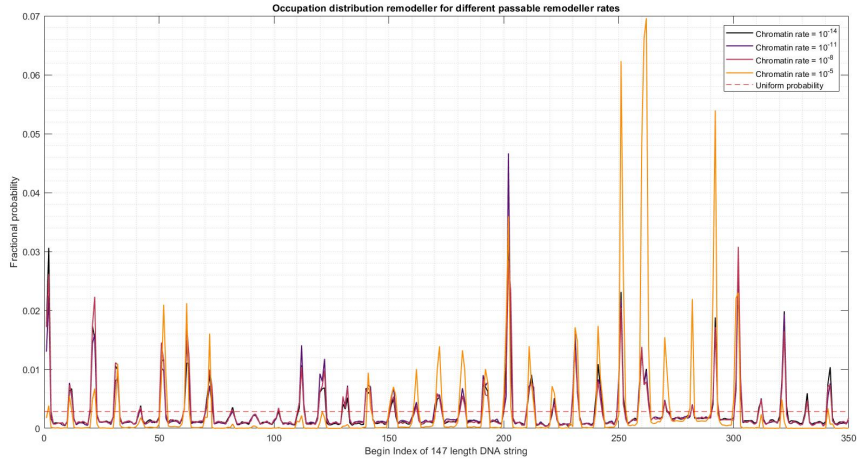

Figure A.14: *Parameter sweep for different remodeler passage rates*

A.2.2 Other lifetime plots

We have also calculated the defect lifetimes for some very simple sequences. First of an all AAA sequence And finally we used a 5 base pair periodic

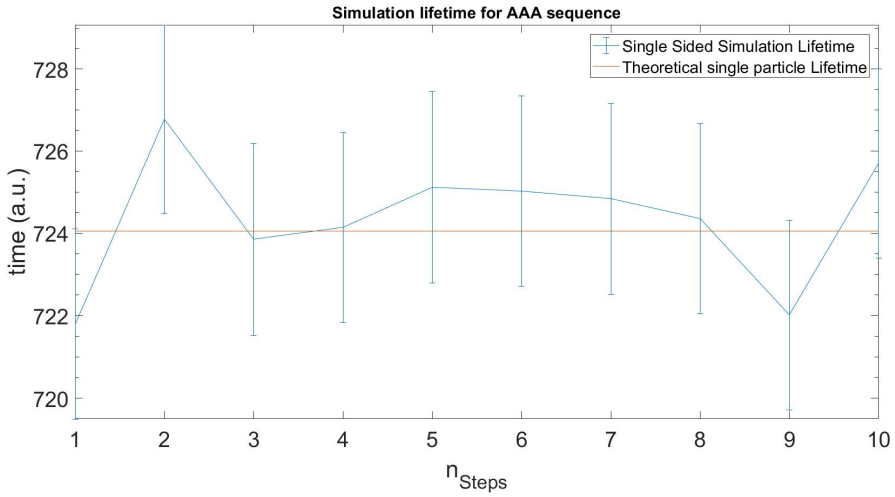

Figure A.15: *Lifetime of the defects on an all AAA sequence. We recover that the lifetime is the same all over the simulation except for statistical noise.*

sequence which should give a landscape that repeats itself once.

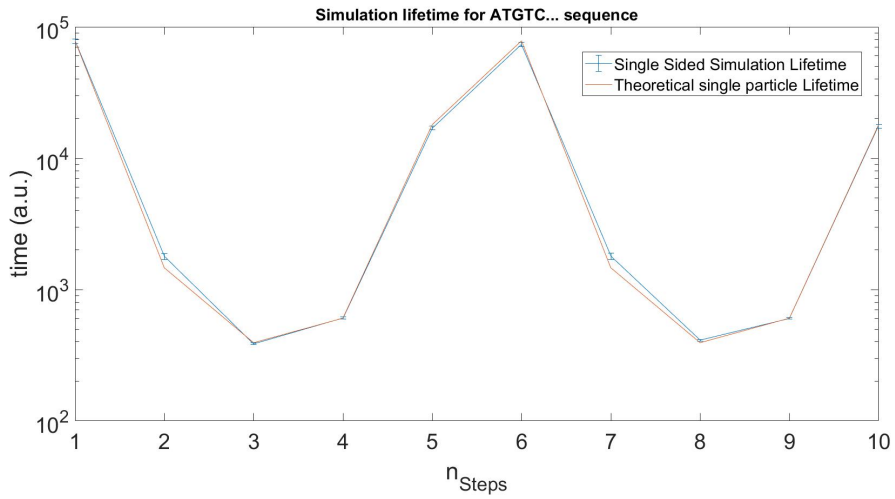

Figure A.16: *Lifetime of the defects on a 5 base pair periodic ATGTC sequence. We recover that the lifetime repeats after 5 steps.*

Version of October 10, 2022– Created October 10, 2022 - 09:08

A.3 Sequences used

We give the two most commonly used sequences in this thesis here, the full list of used sequences is available in the code too.

Energy landscape sequence:

Here we used a series of 100 randomly generated positional dependent sequences and average them out to get the positional dependent energy landscape. However sometimes we will have a landscape done with just a single sequence which will be:
ACATACATTCTTTTAAATTTCGAAGACCTCGTGATTCTTCCCGCGTT
TACTGGCAAAATCTTAGAAATCGCGGCAAAGTCAAAAAAATTTTACA
AATTGTCGTCGGGTCCTTAGAATTCGGCAGTCGTTTCTGGGTTCACG
CACGACTTCTTTGAAAGGTCAAAAGCCGCCGAGTTTTTTCGGCGTAA
CTCCCGATCAGACCGAAACATCTGAAAACTTTCCTGTGACTACGCAG
ACGTCCGGGAAGTTCAAATCGTTTCGGGAGTTATCTTGAGATTTCTT
CAAAACACACCAGCGTTTTCGAAAAATCTCCGTAATCACGCAAAACA
CGCTACGAATCAGAAAACGAA

Occupancy sequence:

GGTGGGCTCCGAAAAATTCGCAGGGCGACCGGCGAAATGCTCAAAAAA
TCAAAAAATATTCCCTGAAACAAAAGCAACTCTTGCGAAACGGGCGCA
TTTTTAAAAAATTCGGAGAAAATTTTTGAAATCGTGACGCATCTCTTG
CCGTTCGCCAAACAATCCAGAAATTTTATTGTTCAGGCAAAATTCACT
AGGTTTTATGGTGAAACGCAAAAAATTCTGACGTTTTCATGAACGTCT
TTTAGGATTTTCAGGTTAATGCGGTTGGGCTCCAAAATCTCAAGCAGT
CTCGTAGAAATTTCAAAAATTTCGGCATTCTTGGAGAATCACGGGAAG
ATACTCGCCGGTTT

Version of October 10, 2022– Created October 10, 2022 - 09:08