PACIFIC INSTITUTE FOR THE MATHEMATICAL SCIENCES VIRTUAL EXPERIMENTAL MATHEMATICS LAB (PIMS VXML) FINAL REPORT: COMBINATORICS AND KNOT THEORY FOR RNA-DNA COMPLEXES II

FACULTY MENTORS: MARGHERITA MARIA FERRARI[1]; CHRIS SOTEROS[2]. POSTDOC MENTOR: MATTHEW SCHMIRLER[2]. TEAM MEMBERS: SHRUTI KAREN KAUR[2]; OLIVIA NUMEDAHL[2]; MINGZE SUN[3], BRYNN WILLIAMS[4]

1. Introduction

The nucleic acids DNA and RNA contain the genetic code of life. This is encoded in each molecule via their nucleotide sequences. DNA and RNA naturally interact with each other during cellular processes such as transcription. These processes modify the geometry and topology of DNA and RNA, thus sometimes yielding multi-stranded structures. An R-loop is a 3-stranded structure formed during transcription and it is composed of an RNA-DNA complex and another single strand of DNA. Experimental studies indicate that R-loops can play either destructive or regulatory roles in cellular processes; see, for instance, [1]. Thus, it is important to determine the factors influencing R-loop formation and stability. It is known that both DNA sequence and geometry/topology affect R-loop formation [1, 2, 3]; however, little is known about their geometric and topological entanglement properties.

In order to begin to model the geometric/topological features of R-loops, in this work we introduce and start to build a simplified 3-d lattice polymer model.

To explain our model further, we begin by giving more background about R-loops, some existing models, and a review of some relevant definitions and theory. We then highlight questions to be addressed in developing the model. Following this, we present the progress made towards developing the 3-d model. Finally, we discuss conclusions and possible future work.

Date: Academic Year 2023–24.

1: University of Manitoba; 2: University of Saskatchewan; 3: University of Alberta; 4: University of Victoria

2. Background and the Initial Problem

Transcription involves an enzyme, called RNA polymerase, acting on double-stranded DNA to create a new RNA molecule using the template strand of the DNA. R-loops are RNA-DNA structures that can occur through transcription when the newly formed RNA binds to the template DNA strand. This results in a 3-stranded RNA-DNA structure consisting of an RNA-DNA complex along with the displaced non-template DNA strand.

Based on the 'thread back model', the newly formed RNA threads back into the double-stranded DNA behind the transcription machinery (the location of the RNA polymerase) to form an R-loop [4, 5]. Figure 1 depicts an R-loop formed during transcription.

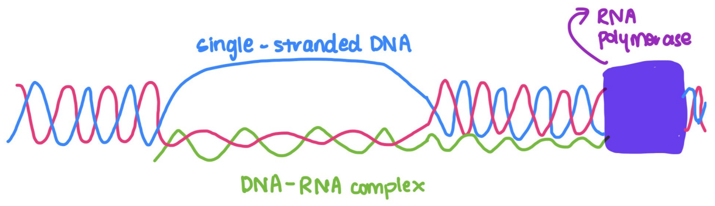

Figure 1. Schematic representation of an R-loop (blue: non-template DNA; red: template DNA; green: RNA transcript; blue box: RNA polymerase). Diagram Credit: S. Kaur.

There is evidence that the formation of an R-loop is favored when a G-rich RNA transcript is created from the template DNA [1, 6]. There is also evidence that the amount of supercoiling in the DNA affects R-loop formation, with more negatively supercoiled DNA promoting R-loop formation [2]. Hence both DNA sequence and DNA topology affect R-loop formation.

Experimental studies of R-loops [2] for specific DNA sequences have yielded site-specific probabilities for R-loop formation under different topological states of the DNA. Existing models [2, 7, 8, 9, 10] can be used to predict where R-loops are more likely to appear in a given DNA sequence; however, these models do not include detailed representations of the R-loop geometry.

The goal of this work is to develop a 3-d lattice polymer model of RNA-DNA structures that incorporates information from site-specific R-loop formation probabilities. Such a model will allow us to address more detailed geometric and topological questions about R-loops.

Towards developing the model, we next present further important background information.

2.1. **Lattice Models of Polymers in Tubes.** A polymer is considered to be any large molecule that is made of repeated molecular units called monomers. Thus DNA and RNA can be thought of as polymers and indeed statistical mechanics models of polymers have proved useful for modelling the average conformational (topological/geometrical) properties of DNA in solution [11]. Such models include lattice models such as self-avoiding walk (SAW) and polygon (SAP) models.

For polymers under confinement conditions, one standard lattice model considers walks or polygons confined to a tubular sublattice of the simple cubic lattice [12]. In this case, the polymer is represented by a set of vertices in $\mathbb{Z}^3$ that are joined by unit edges. For tubular confinement, the vertices are bounded in the y and z directions, with free growth allowed in the positive x-direction. For example, for the (L, M)-tube, the vertex coordinates must satisfy: $x > 0$, $0 < y \leq L$, $0 < z \leq M$. In the case of $M = 0$, the tube is 2-dimensional and called a *slit*. A mathematical advantage of lattice tube models is that they can often be studied exactly using transfer matrix and/or Markov chain methods.

Last year, the team working on this project [13] studied two first-step Markov Chain models. One model was a model for R-loop formation based on the formal grammar model of [9, 10] and the other was a simple 2-d lattice model for R-loop geometry. For the lattice model, the R-loop was represented by two walks in a slit of height 2 ((2,0)-tube); the walks were obtained by dividing a polygon into two walks which cross the slit. One walk was used to represent the double-stranded RNA-DNA complex and the other represented the displaced single-stranded DNA. Since a double-stranded molecule is much less flexible than a single-stranded one, the probability of forming a bend (a right angle) was taken to be different for each of the two walks. Work on this project was continued over the summer [14] to combine the two Markov chain models together. This allowed the RNA-DNA complex to be modelled with different bend probabilities in the two walks before, during and after an R-loop. Properties such as average R-loop length and average bend counts as well as the conditional probability of being in an R-loop at a given step in a fixed-length chain were studied and compared to the experimental results from [2]. Although this Markov chain model was very simplified, the model could take into account different flexibilities between double-stranded and single-stranded polymers and could incorporate different transition probabilities arising from the formal grammar model. However, the Markov chain model assumed that the transition probabilities were time-homogeneous and hence not site-specific. Comparing the model to the results from [2] made it clear that site-specific transition probabilities are needed to reproduce the experimental results. Also, the 2-d nature of the model made it impossible to study entanglements and supercoiling, which are known to be relevant to R-loop formation.

To address the short-comings of the 2-d lattice model, the goal of this work is to develop a 3-d lattice model that can take into account the site-specific probabilities for being in an R-loop that are available from experiments. Important ingredients for such a model include choosing: the lattice structures used to model the RNA and DNA geometry; the number of nucleotides that correspond to a lattice edge; a way to represent different polymer flexibilities; and a way to assign probabilities to different lattice structures.

3-d polygons in small lattice tubes have already been studied [15] and data sets generated from that study have been made available to us. Thus a natural first step is to use the available data to develop a 3-d model and answer the following questions.

Initial Question: How do we use a random sample of lattice tube polygons to model RNA-DNA geometry?

Sub-question 1: How many nucleotides correspond to a lattice edge?

Sub-question 2 How do we incorporate different flexibilities between two parts?

Sub-question 3: How do we assign probabilities to different RNA-DNA configurations?

In order to discuss sub-question 1, we review some relevant background next.

2.2. Measuring Flexibility.

The nature of the monomer units making up a polymer affect the flexibility (or stiffness) of the polymer. One way to measure a polymer's flexibility experimentally is via a geometric quantity called *persistence length*. For DNA, roughly speaking, after two persistence lengths the correlation between starting and ending directions of the DNA will be negligible [16].

Double-stranded DNA (dsDNA) is among the stiffest of known polymers with a persistence length of 50 nm (150 bp) in 0.1M aqueous NaCl [17]. Based on previous experiments, the persistence length of dsDNA is significantly greater than that of single-stranded DNA (ssDNA), 1.48 nm (2 bases) [18], and single-stranded RNA (ssRNA), 1.0 nm (1.5 bases) [19].

Persistence length provides one way to address sub-question 1 (how many nucleotides or base pairs correspond to a lattice edge?). For example, if the persistence length for a lattice polymer model is known to be ℓ lattice edges, then to represent dsDNA with persistence length of 150 bp, 1 edge corresponds to $150/\ell$ base pairs. On the other hand, to model ssRNA/ssDNA, 1 edge would correspond to $\approx 2/\ell$ base pairs. For either case, it is of interest to estimate the persistence length for a lattice model.

The following description is based on the presentation in [20]. To estimate persistence length we use a position-dependent correlation length introduced by Flory [21]. Given a chain consisting of $N+1$ monomers with position vectors $\{\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_N\}$ connected by the bonds $\{\vec{a}_i | \vec{a}_i = \vec{r}_i - \vec{r}_i \ \forall \ i = 1, \ldots, N\}$, we define the orientational correlation locally for each bond k along the chain by

$$c(k) := \frac{\vec{a}_k \cdot \vec{R}_e}{||\vec{a}_k||^2}, \tag{1}$$

where $\vec{R}_e = \vec{r}_N - \vec{r}_0$ and $||\vec{a}_k||$ denotes the length of the vector $\vec{a}_k$. For chains (walks) on the simple cubic lattice, any edge vector has length 1 and the expression in Equation 1 simplifies to $c(k) = \vec{a}_k \cdot \vec{R}_e$.

Note that the definition of $c(k)$ in Equation 1 is for a single chain. We are interested in estimating the expected value of $c(k)$ over all chains at equilibrium (denoted by $E\left[c(k)\right]$). At equilibrium, this expected value is related to a positional persistence length [22]:

$$E\left[c(k)\right] = \frac{l_p(k)}{l_b}, \tag{2}$$

where l_b is the length of the bonds between monomers, and $l_p(k)$ is the persistence length at chain position k. In the simple cubic lattice, $l_b = ||\vec{a}_k|| = 1$, thus Equation 2 simplifies to $l_p(k) = E\left[c(k)\right]$. It is expected that $l_p(k)$ reaches a plateau in the chain interior; this plateau can be used to estimate the persistence length in the model for a fixed chain length N [23]. Since we know the experimental values for the number of bases (nucleotides) in a persistence length of dsDNA, ssDNA and ssRNA, as discussed above, we can divide this quantity by the number of lattice edges per persistence length in order to estimate the number of bases per lattice edge in our model.

3. New directions

To answer our initial questions, for our first step in developing a 3-d lattice model of R-loops in a tube, we used the random polygons with fixed *span* m (the maximum extent in the x-direction) available from [15]. Each polygon is divided into two walks by deleting an edge in the $x = 0$ plane and an edge in the $x = m$ plane. One walk is considered double-stranded and the other single-stranded (see Figure 2). For now these choices have been made arbitrarily but the goal is to develop systematic ways to choose which edges to delete and which walk represents what. We were able to plot these polygons in 3-d, this is described further in Section 4.1. We then take these polygons and use them to model R-loops but we are left with some more questions:

Sub-question 4: How do we decide which edges in the $x = 0$ and $x = m$ plane to delete?

Sub-question 5: How do we decide which walk is considered double-stranded and which is single-stranded?

Sub-question 6: How do we decide which geometries are suitable for the start or end of an R-loop?

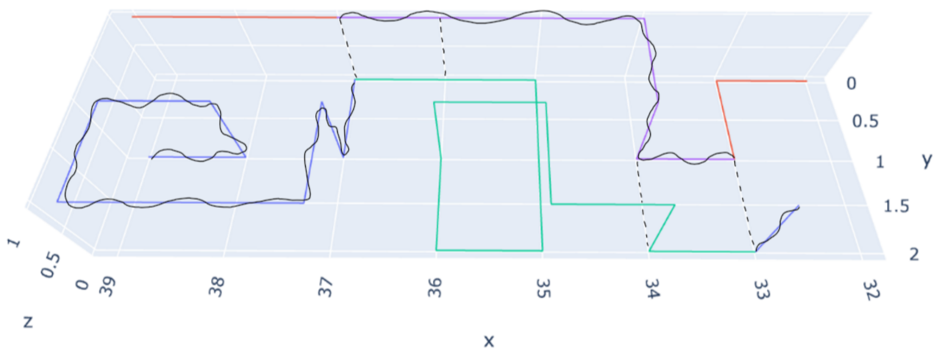

FIGURE 2. Portion of lattice model of an R-loop in a (2, 1)-tube: single-stranded RNA before and after R-loop (red); DNA/DNA before and after R-loop (blue/black); single-stranded DNA during R-loop (green); RNA/DNA during R-loop (purple/black)

Answers to sub-questions 4 and 5 will be left to future work; we discuss one way to address sub-question 6 next.

Using what is known about R-loop formation, we can put constraints on where they would occur in a polygon. We assume that the R-loop beginning or end requires the two walks of the polygon to be close so that the RNA can easily invade. One of the simplest situations where this could happen is at a *2-section*. A 2-section is a half integer x-plane that intersects only two edges of the polygon. At a 2-section, we only have two strands thus we know for certain that one is double-stranded and the other is single-stranded, so we assume that an R-loop will start and end at two different 2-sections. In Figure 2 the black dotted lines represent the 2-sections. As we can see there are only 2 edges that intersect the $x = 36.5$ plane and the $x = 33.5$ plane respectively. To make sure that the two walks are close enough in these 2-sections, we assume further that only the 2-sections where the two edges are only one edge apart in either the y or z direction are suitable for the start or end of an R-loop. We will call these special 2-sections, *2*-sections.* This leads us to some new questions:

Sub-question 7: How can we find 2-sections and 2*-sections in this 3-d model?

Sub-question 8: How do we visualize the model geometry using these 2-sections?

Sub-question 9: How do we decide which 2*-sections are the start and end of the R-loop?

4. Progress

In this section we discuss the work done towards answering the initial question and some of the sub-questions.

The polygons made available to us via [15] were given in files in *UofS format.* A file of this format begins on line 1 with "UofS", then on line 2 a sequence of coordinates for a single vertex, xyz, denoting the beginning of the polygon. On each subsequent line there will be a number from 1 to 6; this indicates what lattice direction the next edge is. A 1 indicates a step in the $+x$ direction, 2 is $-x$, 3 is $+y$, 4 is $-y$, 5 is $+z$, and 6 is $-z$. At the end of all the directions (once you have come back to the starting vertex), there is a -111 indicating the end of the polygon. There can be multiple polygons in a file, in that case after the -111 would come a new vertex xyz denoting the beginning of the new polygon. The file will end with -999 indicating there are no more polygons stored in this file.

4.1. **Progress on a 3-d Lattice Model: Sub-questions 7,8.**

Part of the progress on the 3-d model involved developing the following computer programs:

- `B. Williams' Python Code 1:` This code, written in Python 3, reads in a file of UofS format and translates the data into vertices. It gives a list of polygons with each polygon having three lists with their x, y, and z coordinates respectively, such that `poly[0][0]` gives the list of the first polygon's x coordinates, `poly[1][1]` gives the list of the second polygon's y

coordinates, and `poly[0][2][0]` gives the first polygon's initial z coordinate. The input polygons used for this section were generated from [11].

Algorithm 1 Translating UofS format to Vertices

```
Input: UofS file
Open UofS file and readlines
Intg = empty list, poly = empty list, polys = empty list, x = empty list, y = empty list, z = empty list, pos = empty list, polygons = empty list
for every line in file do
    if the length of the line is >= 5 then
        remove '\n' and ' '
        for every index in the length of the line do
            append each individually as an integer to Intg
        end for
    end if
    if the length of the line is 3 then
        remove '\n' and append as an integer to Intg
    end if
    if the line is '−111\n' then
        append 'end' to Intg
    end if
    if the line is '−999\n' then
        Break loop
    end if
end for
for every value in Intg do
    if the value is 'end' then
        append poly to polys
    else
        append value to poly
    end if
end for
Append final poly to polys and remove extra empty list from polys
for every index i of the number of lists in the list of polys do
    for every index n of the length of the polyon do
        if the index n is 0, 1, and 2 then
            append polys[i][n] to x, y, and z respectively
        end if
        if the index n is 3 or greater then
            depending on the value polys[i][n] append the corresponding change to the last x, y, or z coordinate in their respective lists and append the last x, y and z coordinate as it was before if that coordinate didn't change
        end if
    end for
    Append the lists of the x, y, and z coordinates to pos and then append pos to polygons
end for
Output: list of 3 lists each containing the $x, y, z$ coordinates respectively for the vertices of the polygon
```

- `O. Numedahl's Python Code 1:` This code, written in Python 3, takes an input polygon and looks at the x coordinates $x = i$. As you go along a polygon's edges, when x changes to $x = i + \Delta$ where $\Delta \in \{-1, 1\}$, then the half integer plane being crossed by this edge becomes $x = i + \Delta/2$. The code goes through the entire polygon and records every time any half integer plane is crossed. Then it keeps the ones that only have 2 edges through them, thus the 2-sections. It looks at all these 2-sections and only picks out the ones which are one edge apart in either the y coordinate or the z coordinate, thus the 2*-sections. It then records which half integer planes contain these 2*-sections and which edges are going through them.

Algorithm 2 Finding 2*-sections

```
1: Input: list of 3 lists each containing the $x, y, z$ coordinates respectively for the vertices of the polygon
2: $x_0$ = previous $x$ coordinate and $x_1$ = current $x$ coordinate
3: for every vertex in the polygon do
4:     if $x$ coordinate changes then
5:         Record half integer plane $|x_1 - x_0|/2$ as having $+1$ edge going through
6:         Record coordinates of the current and previous vertices
7:     end if
8: end for
9: for Every half integer plane in the span of the polygon do
10:     if plane has only 2 edges going through then
11:         if 2 edges crossing the plane are one edge apart then
12:             Add this plane to a list 2*-sections
13:         end if
14:     end if
15: end for
16: Output: 2*-sections
```

- `B. Williams' Python Code 2:` This code, written in Python 3, takes the list of polygons from the first code and creates a dataframe of the first polygon in the list, such that there are four columns one for the x, y, and z coordinates respectively and another listing their position in the polygon. This code then creates an empty 3-d figure and two traces are added, one for each walk with the edge connecting them removed. This gives us Figure 3. Then using the results of the code above we create two lists of the edges that pass through the 2-sections, one for the first walk and one for the second, this gives the positions in the polygon where the 2-sections occur. We can filter these two lists and find ones that only have one edge apart in either the y or z direction. Then we use the fourth column to select the vertices on either side of a 2-section. Using this we can add traces for each 2-section that are only one edge apart. The trace starts at the first 2-section and ends at the next closest 2-section, this is graphed using a for-loop. This graph is shown in Figure 4. All of the code is done so that it can graph the same for any given polygon of this format.

Algorithm 3 Graphing basic 3-d Model

```
1: Input: list of 3 lists each containing the $x, y, z$ coordinates respectively for the vertices of the polygon
2: choose one polygon and zip the x, y, and z coordinates and a list of numbers 1 to the length of the polygon removing the last coordinates, turn this into a list and assign it to a variable ver
3: create dataframe of ver with columns x, y, z, and w call it df
4: create an empty plotly express 3-d line figure call it fig
5: add scatter 3-d to fig with x = df[df['w']<=walklength]['x'] for x, y, and z repectively setting mode = 'lines' and line = dict(width = 4, color = 'blue')
6: add another scatter 3-d to fig with x = df[df['w']>walklength]['x'] for x, y, and z repectively setting mode = 'lines' and line = dict(width = 4, color = 'red')
7: update layout of fig to have scene = dict(aspectmode = 'data')
8: Output: Figure 3
```

Algorithm 4 Graphing 2*-sections in 3-d

```
1: Input: fig from algorithm 3
2: create a list of the edges found in the output of algorithm 2 from the first vertex to the end of the walk call it walk1
3: create another list of edges found from algorithm 2 from the end of the walk to the last vertex call it walk2
4: for index i in the range 0 to length of walk1 do
5:     if the index i is divisible by 2 then
6:         m=(df['w']>=walk1[i]) and (df['w']<=walk1[i+1]+1)
7:         add scatter 3-d for x=df[m]['x'] for x, y, and z respectively setting mode = 'lines' and line = dict(width = 4, color = 'violet')
8:         m=(df['w']>=walk2[i+1]) and (df['w']<=walk2[i]+1)
9:         add scatter 3-d for x=df[m]['x'] for x, y, and z respectively setting mode = 'lines' and line = dict(width = 4, color = 'mediumseagreen')
10:    end if
11: end for
12: Output: Figure 4
```

4.1.1. *Results:* Regarding Sub-question 7: we found from O. Numedahl's Code 1 that by looking at only the x coordinate we can find the 2-sections and by looking at the y and z coordinates we can easily find the 2-sections that are one edge apart, the 2*-sections. O. Numedahl's Code 1 was only used on a few polygons of span 50 for the results to be used to visualize via B. Williams' Code 2. It can be used on polygons of any span and we could use this to find the average number of 2*-sections for polygons of a given span. Given the theory [24, Theorem 2.6.2] we would expect to find an $\alpha_T > 0$ such that the number of 2*-sections per span for a span m polygon goes to α_T as $m \to \infty$, that is, $\lim_{m\to\infty} E[\#2*-\text{sections}]/m = \alpha_T$, where $E[\#2*-\text{sections}]$ is the expected number of 2*-sections in a random span m polygon in a given tube T.

Results for Sub-question 8: B. Williams' Code 1 was tested for polygons up to span 250; it should work for polygons of any span. Algorithm 3 and 4 were written to take as input any polygon with any given span. The two figures below illustrate the output from these two algorithms for one specific polygon with span 50 in the $(2,1)$-tube.

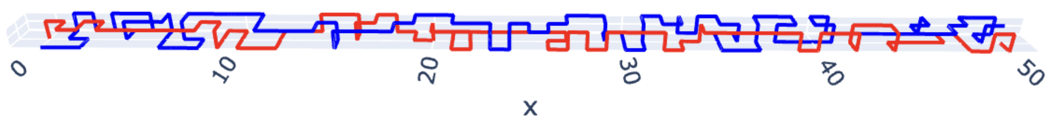

FIGURE 3. Output from Algorithm 3 with input a span 50 polygon in the $(2,1)$-tube: the output shows two walks with the same span; one walk (red, say) can be used to model a single-stranded molecule and the other a double-stranded molecule (blue).

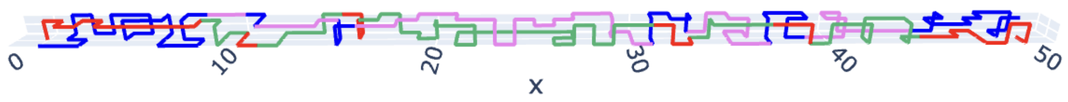

FIGURE 4. Output from Algorithm 4 with input a span 50 polygon in the $(2,1)$-tube: the output is two walks where colours on the walks change at 2*-sections. One assignment of colours could be as follows: single-stranded RNA (red); double-stranded DNA (blue); RNA/DNA during R-loop (green); single-stranded DNA during R-loop (purple).

4.2. **Progress on Persistence Length: Sub-question 1.** For the purpose of estimating persistence length, we used $(2,1)$-tube polygon data consisting of 10,000 independently sampled polygons for each span $m = 50, 100, 150, 200, 250$. Given a span m, the 10,000 independent samples were generated with each polygon of span m being equally likely. All sample polygon files were in the UofS file format.

- `M. Sun's Python Code 1:` Given a UofS format file containing multiple polygons with a specific span m, M. Sun wrote an algorithm to calculate the persistence length using Python 3, shown in Algorithm 5. The first step is to take the first $m+1$ vertices in the polygon to get a walk of length m from the polygon. The reason for this choice of walk length stems from the fact that our RNA-DNA model involves dividing a span m polygon into two walks, each of which starts in the plane $x = 0$ and ends in the plane $x = m$. Choosing walk length m for the persistence length calculation ensures that the chosen walk exists and is not long enough to turn back after reaching the $x = m$ plane.

Figure 5 may help with understanding the algorithm. For example, given a lattice walk with 9 vertices *i.e.* $walklength = 8$, $\vec{R}_e$ is defined as an end-to-end vector. We aim to get the dot product of $\vec{R}_e$ and $\vec{a}_i$ where i is in range [1, *walklength*]. Repeat this process for every walk in the sample, and compute the average of these dot products to get a position dependent persistence length.

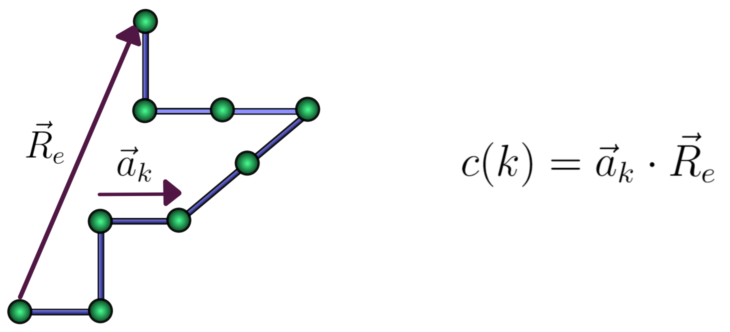

FIGURE 5. For a given walk, the positional correlation $c(k)$ is equal to the dot product of the k^{th} edge vector, $\vec{a}_k$, with the end-to-end vector, $\vec{R}_e$.

Algorithm 5 Computing Persistence Length

```
1: Input: Arrays containing polygon vertices and polygon-span
2: $walklength$ = polygon-span and initialize array $dotprods$ as 0
3: for each polygon $p$ do
4:     Extract the first ($walklength$ + 1) vertices from $p$
5:     Compute $\vec{R}_e$ as $p[-1]$ - $p[0]$
6:     for $j = 1$ to $walklength$ do
7:         Compute the $i$th edge direction as $p[i+1]$ - $p[i]$ ($edgedir$)
8:         Compute the dot product of $\vec{R}_e$ and $edgedir$ ($dotprod$)
9:         Update $dotprods[i]$ += $dotprod$
10:    end for
11: end for
12: for $i = 1$ to $walklength$ do
13:     Compute the average of $dotprods[i]$ over $numpolys$
14: end for
15: Output $dotprods$
```

4.2.1. *Results:* Regarding Sub-question 1:

Using Algorithm 5, we obtained graphs of persistence length with respect to the chain position. Using span 250 as an example, we estimated the peak of the curve that covers 95% points, shown in Figure 6. Repeating the process for other spans yielded Figure 7, which shows an increase in persistence length with larger spans. For the given DNA polygons with span 250, the estimated persistence length is approximately 42 lattice edges. If we assume the persistence length of DNA is 150 bp, then there are approximately $150/42 = 3.57$ base pairs per edge.

The lattice persistence length calculated here is for the distribution of lattice tube polygons where each polygon of a given span is equally likely. For a distribution where polygons are weighted according to the number of bends (right angles), the measured persistence length is expected to, for example, decrease if bends are favoured. This suggests a way to impose different flexibilities on different parts of the polygon and thus address sub-question 2.

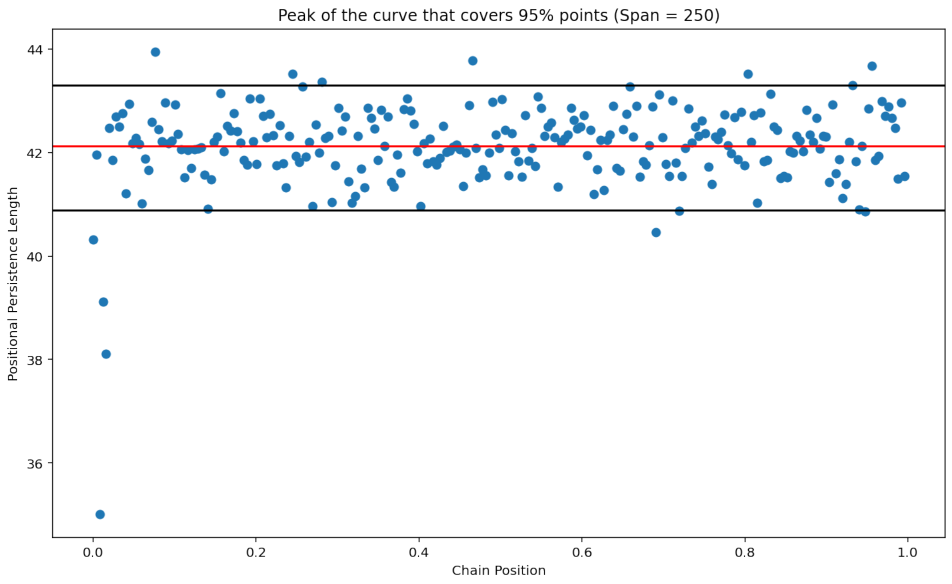

Figure 6. For walk-length $m = 250$, estimates for $E[c(k)]$ plotted versus k/m. The red line indicates the plateau which is used to estimate persistence length. Black horizontal lines indicate an estimated 95% confidence interval for the persistence length.

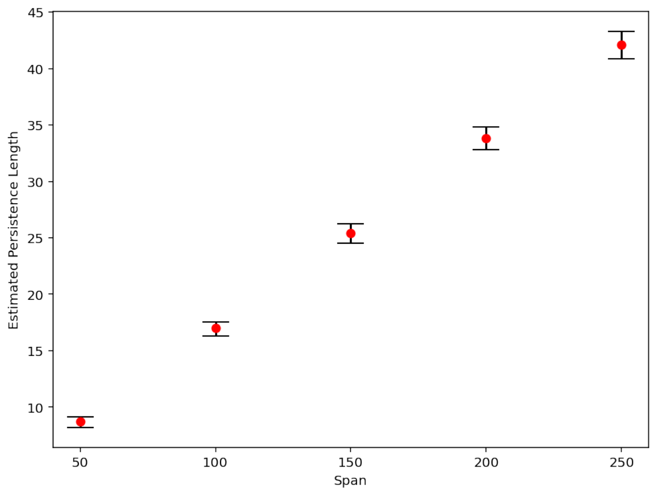

Figure 7. Estimates for persistence length for length-m walks in the $(2, 1)$-tube, $m = 50, 100, 150, 200, 250$. Error bars indicate an estimated 95% confidence interval

5. Conclusions and Future directions

In summary, we made a first step towards developing a 3-d lattice tube model for studying the properties of R-loops in 3-d.

Future directions will begin with completing an answer to our initial question, this will include answering sub-questions 2-5. It would start with finishing the 3-d model and assigning probabilities to each possible model configuration. From this, we can get average geometric and topological properties of the RNA-DNA complexes. For this purpose, we can upload the lattice walks and polygons from the model into Knotplot [25] and measure entanglement complexity using the knotplot toolbox.

Acknowledgments

The team acknowledges helpful discussions with M. Vazquez and J. Lusk. We also acknowledge the usefulness of the CoCalc account provided by PIMS and Overleaf accounts provided by MMF and CES. CES acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number RGPIN-2020-06339].

References

[1] Y. A. Hegazy, C. M. Fernando, and E. J. Tran. "The balancing act of R-loop biology: The good, the bad, and the ugly". *Journal of Biological Chemistry* **295**.4 (2020), pp. 905–913.

[2] R. Stolz, S. Sulthana, S. R. Hartono, M. Malig, C. J. Benham, and F. Chédin. "Interplay between DNA sequence and negative superhelicity drives R-loop structures". *Proceedings of the National Academy of Sciences* **116**.13 (2019), pp. 6260–6269.

[3] F. Chédin and C. J. Benham. "Emerging roles for R-loop structures in the management of topological stress". *Journal of Biological Chemistry* **295**.14 (2020), pp. 4684–4695.

[4] D. Roy, K. Yu, and M. R. Lieber. "Mechanism of R-loop formation at immunoglobulin class switch sequences". *Molecular and Cellular Biology* **28**.1 (2008), pp. 50–60.

[5] B. P. Belotserkovskii, S. Tornaletti, A. D. D'Souza, and P. C. Hanawalt. "R-loop generation during transcription: Formation, processing and cellular outcomes". *DNA repair* **71** (2018), pp. 69–81.

[6] D. Roy and M. R. Lieber. "G clustering is important for the initiation of transcription-induced R-loops in vitro, whereas high G density without clustering is sufficient thereafter". *Molecular and Cellular Biology* **29**.11 (2009), pp. 3124–3133.

[7] T. Wongsurawat, P. Jenjaroenpun, C. K. Kwoh, and V. Kuznetsov. "Quantitative model of R-loop forming structures reveals a novel level of RNA–DNA interactome complexity". *Nucleic Acids Research* **40**.2 (2012), e16–e16.

[8] P. Jenjaroenpun, T. Wongsurawat, S. P. Yenamandra, and V. A. Kuznetsov. "QmRLFS-finder: a model, web server and stand-alone tool for prediction and analysis of R-loop forming sequences". *Nucleic Acids Research* **43**.W1 (2015), W527–W534.

[9] N. Jonoska, N. Obatake, S. Poznanović, C. Price, M. Riehl, and M. Vazquez. "Modeling RNA:DNA Hybrids with Formal Grammars". *Using Mathematics to Understand Biological Complexity: From Cells to Populations*. Ed. by R. Segal, B. Shtylla, and S. Sindi. Cham: Springer International Publishing, 2021, pp. 35–54. DOI: 10.1007/978-3-030-57129-0_3.

[10] M. M. Ferrari et al. "The R-loop Grammar predicts R-loop formation under different topological constraints". *In preparation* (2024).

[11] E. Orlandini and S. G. Whittington. "Statistical topology of closed curves: Some applications in polymer physics". *Reviews of Modern Physics* **79** (2007).

[12] D. J. Klein. "Asymptotic distributions for self-avoiding walks constrained to strips, cylinders, and tubes". *Journal of Statistical Physics* **23**.5 (1980).

[13] C. Soteros, M. M. Ferrari, M. Schmirler, M. Sun, M. Shvets, and J. Li. “PIMS - VXML Project 2022-2023: Combinatorics and Knot Theory for RNA-DNA Complexes”. https://vxml.pims.math.ca/projects/2022-2023/combinatorics/.

[14] O. Numedahl, C. Soteros, M. Schmirler, and M. M. Ferrari. “Lattice Models of RNA-DNA R-loop Complexes”. https://harvest.usask.ca/items/10faea2d-14fa-4e85-aa2b-9e04daf562b4/.

[15] J. W. Eng. “A transfer matrix approach to studying the entanglement complexity of self-avoiding polygons in lattice tubes”. PhD thesis. University of Saskatchewan, 2020.

[16] A. D. Bates and A. Maxwell. *DNA Topology.* Second. New York: Oxford University Press, 2005.

[17] G. S. Manning. “The persistence length of DNA is reached from the persistence length of its null isomer through an internal electrostatic stretching force”. *Biophysical Journal* **91**.10 (2006), pp. 3607–3616. DOI: `10.1529/biophysj.106.089029`.

[18] B. Tinland, A. Pluen, J. Sturm, and G. Weill. “Persistence length of single-stranded DNA”. *Macromolecules* **30**.19 (1997), pp. 5763–5765.

[19] K. Hayashi et al. “Influence of RNA strand rigidity on polyion complex formation with block catiomers”. *Macromolecular Rapid Communications* **37**.6 (2016), pp. 486–493.

[20] M. Schmirler. “Modelling DNA Knotting Using Interacting Lattice Self-Avoiding Polygon Models”. PhD thesis. University of Saskatchewan, 2022.

[21] P. J. Flory. “Statistical thermodynamics of semi-flexible chain molecules”. *Proc Roy. Soc. A* **234** (1956).

[22] P. J. Flory. *Statistical Mechanics of Chain Molecules.* New York: Interscience, 1969.

[23] H. P. Hsu, W. Paul, and K. Binder. “Standard Definitions of Persistence Length Do Not Describe the Local “Intrinsic” Stiffness of Real Polymer Chains”. *Macromolecules* **43** (2010).

[24] M. Atapour. “Topological entanglement complexity of systems of polygons and walks in tubes”. PhD thesis. University of Saskatchewan, 2008.

[25] R. Scharein. “The KnotPlot Site”. http://knotplot.com.