A Framework of Multi-Dimensional and Multi-Scale Modeling with Applications

Zilong Li

A FRAMEWORK FOR MULTI-DIMENSIONAL AND MULTI-SCALE MODELING WITH APPLICATIONS

by

Zilong Li, B.S., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy of Science

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

March 2020

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

November 14, 2019
Date of dissertation defense

We hereby recommend that the dissertation prepared by

Zilong Li, B.S. , M. S.

entitled **A FRAMEWORK OF MULTI-DIMENSIONAL AND MULTI-SCALE MODELING WITH APPLICATIONS**

be accepted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computational Analysis & Modeling

Thomas C. Bishop, Supervisor of Dissertation Research

Weizhong Dai,
Head of Computational Analysis & Modeling

Members of the Doctoral Committee:
Pradeep Chowriappa
Songming Hou
Galen Turner
Gergana G. Nestorova

Approved:

Hisham Hegab
Dean of Engineering & Science

Approved:

Ramu Ramachandran
Dean of the Graduate School

GS Form 13a
(10/18)

ABSTRACT

In this dissertation, a framework for multi-dimensional and multi-scale modeling is proposed. The essential idea is based on oriented space curves, which can be represented as a 3D slender object or 1D step parameters. SMILES and Masks provide functionalities that extend slender objects into branched and other objects. We treat the conversion between 1D, 2D, 3D, and 4D representations as data unification. A mathematical analysis of different methods applied to helices (a special type of space curves) is also provided. Computational implementation utilizes Model-View-Controller design principles to integrate data unification with graphical visualizations to create a dashboard. Applications of multi-dimensional and multi-scale modeling are provided to study “Magic Snake”, “Nanocar” and “Genome Dashboard”.

APPROVAL FOR SCHOLARLY DISSEMINATION

Author Zilong Li

Date 01/10/2020

DEDICATION

I dedicate this dissertation to my parents for their unconditional love and support, and to Aobo Jiang for going through thick and thin with me the entire Ph.D. journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

We thank Joohyun Kim and Jinghua Ge of the Center for Computation and Technology, Louisiana State University, and John Gentle of Science Gateways Community Institute (SGCI) for assistance and advice. We thank the Schlick Lab at NYU for sharing coordinate data for the HOXC models. We thank Professors Gergana G. Nestorova, Pradeep Chowriappa and Galen Turner as committee members for help with the thesis. We thank Professor Songming Hou for the supervision of the Magic Snake research. We thank Professor Thomas C. Bishop as the advisor for overall guidance. This work was supported by the National Institute of General Medical Sciences of the National Institutes of Health [P20 GM103424-17]; National Science Foundation [OIA-1541079]; and the Louisiana Board of Regents.

CHAPTER 1

INTRODUCTION

1.1 Overview

With computational power increasing exponentially, larger systems can be simulated (e.g. scientists have created the first billion-atom biomolecular simulation in April, 2019 [1]). These simulations may represent the movement of stars in a galaxy or the movement of atoms in a living cell. At the same time, a lot of data are acquired and processed with less costs (e.g. in 2003, it costs $3 billion dollars to sequence the first Human Genome Project, and by 2017, the first $100 dollar genome appears [2]). Today, abundant information such as DNA sequencing and chemical informatics are provided in a variety of databases. Automated experimental workflows are maturing for the production of this informatics data. However, physical modeling and informatics are typically not well integrated. Informatics data, often available as a 1D representation, may not necessarily be associated with a 3D geometric structure, even if it describes a physical object. Meanwhile, physical modeling requires geometric structures, which may not necessarily be related to 1D informatics data. Thus, in many cases, informatics data sets are missing geometric constraints and physical modeling is not fully utilizing the wealth of available informatics data. Here, we propose a multi-dimensional and multi-scale modeling strategy that unifies 1D informatics, 2D analyses, 3D geometries, and 4D simulations.

1.2 Dissertation Outline

In this dissertation, we first describe the mathematical and computational background and previous work used to achieve data unification in Chapter 2. Chapter 3 provides detailed analyses of oriented space curves described in Chapter 2. Then, three applications demonstrate the general utility of the method and include tools for modeling and analysis: Magic Snake toys, Nanocars, and genomics in Chapter 4, 5, and 6, respectively.

CHAPTER 2

BACKGROUND AND PREVIOUS WORK

In order to achieve the desired multi-dimensional and multi-scale modeling, both mathematical and computational methods are needed. This chapter provides the background on previous work of the method utilized in this dissertation. For multi-dimensional modeling, we describe “Space Curves” and “SMILES” for data unification of slender and branched entities, respectively. For multi-scale modeling, we describe “Masks” for data unification through coarse-grained to all-atom to continuous models. Adoption of methods with specific purposes for individual projects are described in later chapters.

2.1 Multi-dimensional Modeling

There are many ways to represent a 3D object. For example, we can identify a sphere by providing the center location and the radius, or draw on a piece of paper, or display it in 3D with a graphical viewer. A mathematical foundation of representing and converting data within different representations becomes essential. Here, we propose a framework for multi-dimensional and multi-scale modeling strategies.

2.1.1 Space Curve

Mathematical Description

A space curve, Figure 2.1, is a three-dimensional curve with center line, $\vec{r}(s) = (r_x\hat{e}_x, r_y\hat{e}_y, r_z\hat{e}_z) = (\vec{r}_x, \vec{r}_y, \vec{r}_z)$, and director frames $\mathbf{D}(s) = \hat{d}_i(s), i = 1, 2, 3$, that track the local orientation of the space curve [3].

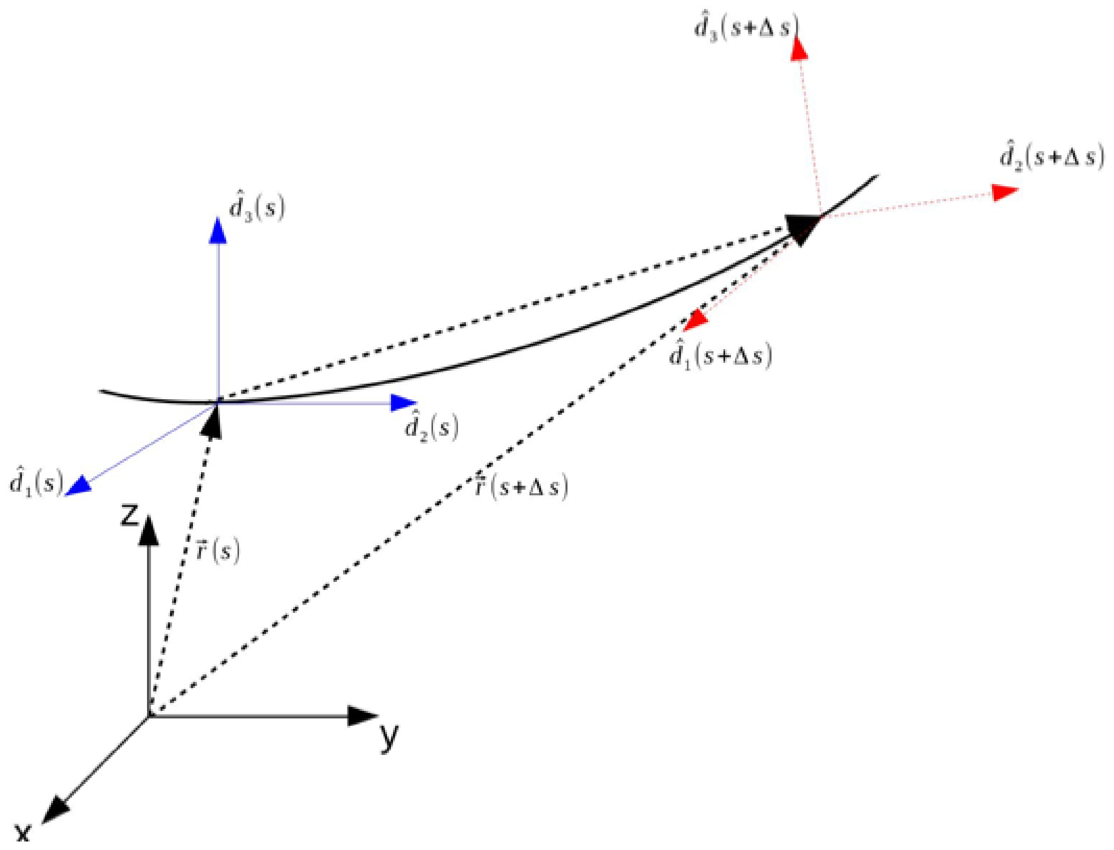


Figure 2.1: Space Curve, where $\vec{r}(s)$ represents the center line, $\mathbf{D}(s)$ $(\hat{d}_i(s), i = 1, 2, 3)$ are director frames.

The combined $[\vec{r}(s),\hat{d}_i(s)]$ representation is labeled $\mathbf{RD}(s)$. An equivalent description of the space curve is based on the director frames themselves. This description is a material reference frame description that captures the translations, $\vec{\Gamma}(s)$, and rotations, $\vec{\Omega}(s)$, connecting adjacent director frames. $\vec{\Gamma}$ is a proper vector.

$\vec{\Omega}$ is a pseudovector. They can be expressed in the local reference frame as $\vec{\Gamma}(s) = (\Gamma_1\hat{d}_1, \Gamma_2\hat{d}_2, \Gamma_3\hat{d}_3)$, and $\vec{\Omega}(s) = (\Omega_1\hat{d}_1, \Omega_2\hat{d}_2, \Omega_3\hat{d}_3)$. The $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ representation is labeled $\mathbf{SP}(s)$ and is a continuous analog of the discrete step parameters (Tilt, Roll, Twist, Shift, Slide, Rise) used to describe DNA [4] or (Pitch, Yaw, Roll, Heave, Sway, Surge) used by navigators [5]. The $\mathbf{RD}(s)$ and $\mathbf{SP}(s)$ representations are mathematically related by the expressions:

$$\frac{d\vec{r}}{ds} = \vec{t} = \mathbf{D}\vec{\Gamma} \tag{2.1}$$

$$\frac{d\hat{d}_i}{ds} = \mathbf{D}\vec{\Omega} \times \hat{d}_i \tag{2.2}$$

$$\vec{r}(s_1, s_2) = \int_{s_1}^{s_2} \vec{t} ds = \int_{s_1}^{s_2} \mathbf{D}\vec{\Gamma} ds \tag{2.3}$$

$$\hat{d}_i(s_1, s_2) = \int_{s_1}^{s_2} \mathbf{D}\vec{\Omega} \times \hat{d}_i ds \tag{2.4}$$

Here, $\vec{t}(s)$ is the non-normalized tangent, and $\mathbf{D}(s)$ is the matrix whose columns are the directors at position s. The Frenet-Serret Tangent-Normal-Binormal, $\mathbf{TNB}(s)$, description of a space curve can be obtained from just $\vec{r}(s)$ [6]. Setting $\mathbf{D} = \mathbf{I}$ in Equation 2.1 equates $\vec{\Gamma}(s)$ to $\vec{t}(s)$. Moreover, the $\mathbf{TNB}(s)$ model is a shear free model with $\vec{t}(s) = \vec{\Gamma}(s)$. The normal $\hat{n}(s)$ is obtained from the second derivative of $\vec{r}(s)$, and the bi-normal $\hat{b}(s)$ is the cross product of $\hat{t}(s)$ and $\hat{n}(s)$, but the $\mathbf{TNB}(s)$ directors are not necessarily the desired director frames for a particular space curve. For example, DNA is highly twisted if the director frames are aligned with base pair frames. Hence, $\mathbf{D}(s)$ obtained from DNA and from $\mathbf{TNB}(s)$ are not the same.

In order to obtain Equations 2.1-2.4, we can study the region from s to $s + \Delta s$ on a space curve, as shown in Figures 2.2 and 2.3. In Figure 2.2, we can treat this point s on the space curve located at the origin of an "imaginary" coordinate system, such that $(\hat{e}_x, \hat{e}_y, \hat{e}_z) = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$. Then, the vector $\vec{r}(s) = (0, 0, 0)$ is in this coordinate system. By taking the derivative of a space curve at position s (or at the origin of this "imaginary" coordinate system), $\frac{d\vec{r}(s)}{ds}$ is $\Delta\vec{r}(s)$ in Figure 2.2, where $\Delta\vec{r}(s)$ has three components along the x, y, z axis of the "imaginary" coordinate system $(\Delta\vec{r}_x, \Delta\vec{r}_y, \Delta\vec{r}_z)$. By the definition of $\vec{\Gamma}$, which is the vector that captures the translations between director frames, we have $(\Delta\vec{r}_x, \Delta\vec{r}_y, \Delta\vec{r}_z) = (\vec{\Gamma}_1, \vec{\Gamma}_2, \vec{\Gamma}_3) = \vec{\Gamma}$. Note that $\vec{\Gamma}$ is represented in the internal frames (can be treated at that the start of the origin point in the "imaginary" coordinate system), so we need to apply $\mathbf{D}$ to it to rotate it to the external frame (the coordinate system that the space curve belongs), which applies to Equation 2.1.

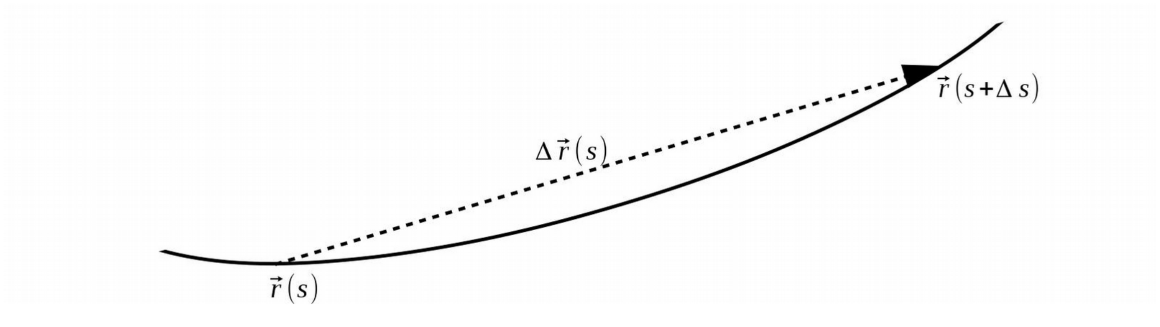


Figure 2.2: The region from s to $s + \Delta s$ on a space curve.

At the same time, $\vec{\Omega}$ is only for the rotations. As shown in Figure 2.3, the rotations are from the director frame at s to the director frame at $s + \Delta s$. Before we apply $\mathbf{D}$, director frame s is just an identity matrix (as in the "imaginary" coordinate system). If we first study $\hat{d}_1$, let $\vec{\Omega} = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3)$, then the rotation $\vec{\Omega}_1$ does not affect $\hat{d}_1$ because it is rotating along it; when $\vec{\Omega}_2$ rotated along $\hat{d}_2$, then $\hat{d}_1$ will move to the negative direction of $\hat{d}_3$ by the amplitude of $|\vec{\Omega}_2|$; and when $\vec{\Omega}_3$ rotated along $\hat{d}_3$, then $\hat{d}_1$ will move to the positive direction of $\hat{d}_2$ by the amplitude of $|\vec{\Omega}_3|$. As a result, we have $\Delta\hat{d}_1 = (0, |\vec{\Omega}_3|, -|\vec{\Omega}_2|) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times (1, 0, 0) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times \hat{d}_1$. In the same way, we have $\Delta\hat{d}_2 = (-|\vec{\Omega}_3|, 0, |\vec{\Omega}_1|) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times (0, 1, 0) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times \hat{d}_2$, and $\Delta\hat{d}_3 = (|\vec{\Omega}_2|, -|\vec{\Omega}_1|, 0) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times (0, 0, 1) = (\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3) \times \hat{d}_3$. Hence, $\frac{d\hat{d}_i}{ds} = \Delta\hat{d}_i = \mathbf{D}\vec{\Omega} \times \hat{d}_i$ as shown in Equation 2.2.

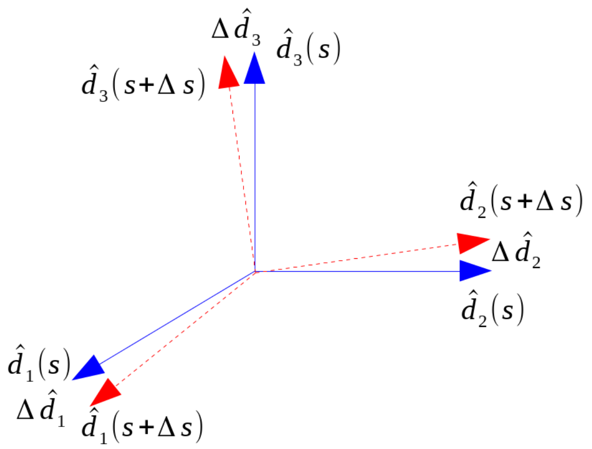


Figure 2.3: Same region as shown in Figure 2.2, $\Delta\hat{d}_i$ represents the rotation from blue to red.

By obtaining these equations, we also notice that we are able to solve these equations in both directions. That is, given $\mathbf{RD}(s)$, we can calculate $\mathbf{SP}(s)$, and given $\mathbf{SP}(s)$, we can calculate $\mathbf{RD}(s)$. It may not be possible to have analytical solutions for such equations, but with the help from numerical techniques such as 4th-order Runge-Kutta, we are able to find numerical solutions for these equations. The problem is that the cross product may not have an inverse or unique solution. In order to obtain $\vec{\Omega}$, we can trace back to how we derived it. That is, with $\Delta\hat{d}_1 = (0, \Omega_3, -\Omega_2)$, we can calculate $\Delta\hat{d}_1$ by $\frac{d\hat{d}_1}{ds}D^T$ first, then we already get $\Omega_3 = \Delta\hat{d}_1 \cdot (0, 1, 0)$, and in the same way, we will observe $\vec{\Omega}$.

Discrete Methods

From a discrete point of view, piecewise differentiation and integration can be associated with discrete director frames. In this case the space curve can even represent a sequentially numbered, yet otherwise unrelated, collection of points in space. In addition to the piecewise differentiation and integration, there are two widely used algorithms in the modeling of DNA. One is the Euler-Angle (E-A) based method [7] used in 3DNA [8], and one is the Euler-Rodrigues (E-R) based method [9] used in CURVES+ [10]. Both E-A and E-R methods adopt the three translations (Shift, Slide, Rise) and three rotations (Tilt, Roll, Twist) as the representation between DNA base pairs. Unlike the continuous differentiation described above, E-A and E-R, as methods implement on double-stranded DNA, they need a mid-step triad to help identify the translations and rotations from one step to the next step. With the method the values obtained are independent of which strand of DNA is identified as the reading strand. Below, the E-A and E-R methods are described.

Euler-Angle Method: Here, we provide the expressions for rotation matrices utilized in the E-A method. A rotation matrix is a matrix that is used to accomplish a rotation in Euclidean space. In 3D, the rotation matrices are given by:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where R_x rotates the object along the x-axis, R_y rotates the object along the y-axis, and R_z rotates the object along the z-axis.

As shown in Figure 2.4, the E-A method utilizes an approximation with Tilt and Roll by letting $Bend = \sqrt{Roll^2 + Tilt^2}$, so that the three rotations $(Tilt, Roll, Twist)$ are reduced to two rotations: $Twist$ and $Bend$. ϕ is defined as $arctan(\frac{Tilt}{Roll})$. By setting up mid-step triad (mst), the E-A method first uses the rotation matrices to rotate the director frame along $\hat{d}_3$ direction by $\frac{Twist}{2} - \phi$, which puts $\mathbf{D}_1 = R_z(\frac{Twist}{2} - \phi)\mathbf{D}_i$. Then, rotate the director frame along $\hat{d}_2$ direction by $Bend$, which is $\mathbf{D}_2 = R_z(\frac{Twist}{2} -$

$\phi)R_y(Bend)\mathbf{D}_i$. Finally, rotate the director frame along $\hat{d}_3$ axis by $\frac{Twist}{2}+\phi$ and we will have $\mathbf{D}_{i+1} = R_z(\frac{Twist}{2}-\phi)R_y(Bend)R_z(\frac{Twist}{2}+\phi)\mathbf{D}_i$, which completes the rotation from step i to step i+1. In the same way, we will get $\mathbf{D}_{mst} = R_z(\frac{Twist}{2}-\phi)R_y(\frac{Bend}{2})R_z(\phi)\mathbf{D}_i$. The translation then is $r(i+1) = r(i) + [Shift, Slide, Rise] \cdot D_{mst}$.

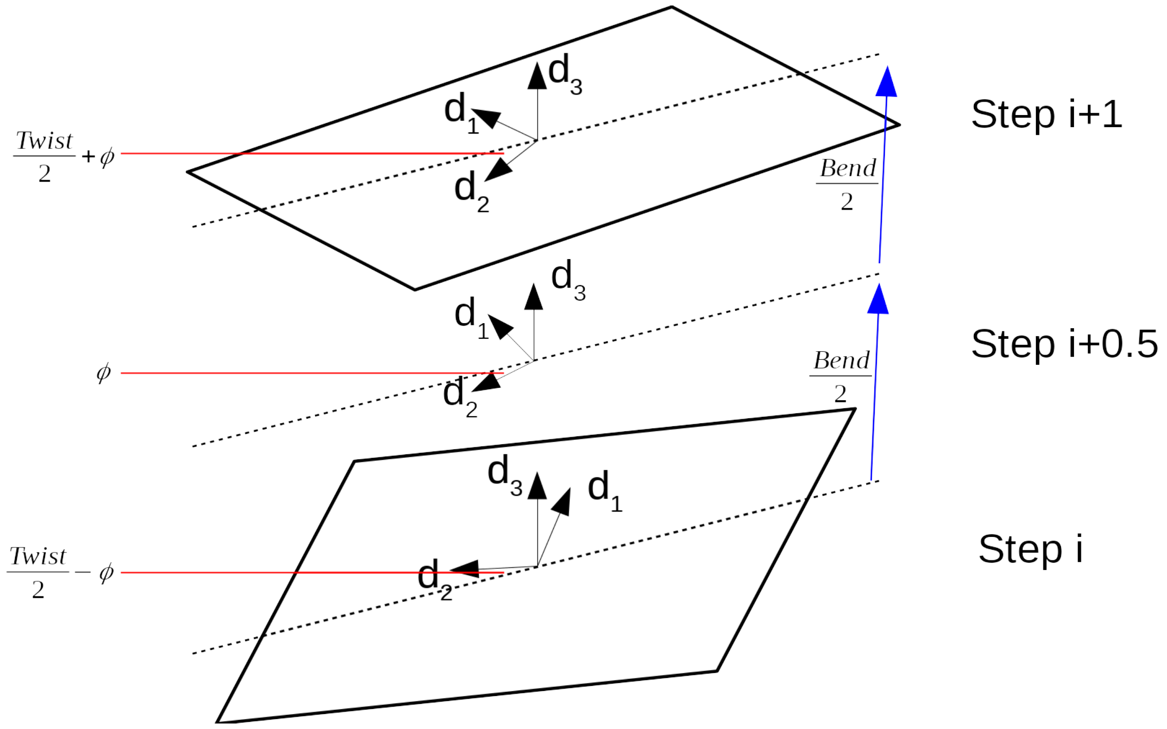


Figure 2.4: The Euler Angle method used to calculate director frames on space curve.

Euler-Rodrigues Method: The E-R method utilizes the the Euler-Rodrigues formula, $Q = cos(\phi)I + (1-cos(\phi))kk^T + sin(\phi)k^{\times}$, where ϕ is given by $arccos(\frac{trQ-1}{2})$, k is given by $[Tilt, Roll, Twist]$, and $k^{\times}$ is a matrix representation of the cross product associated with k. Then $\mathbf{D}_{i+1} = Q\mathbf{D}_i$, and the mid-step rotation is just the square root of Q, and $r(i+1) = r(i) + [Shift, Slide, Rise] \cdot \sqrt{Q}$.

Two applications of applying the space curve based on the properties of the objects themselves are described in detail for Magic Snake and DNA in the next chapters for Magic Snake and Genome Dashboards.

2.1.2 SMILES

Space curves are general enough to describe any slender body, but branched entities such as chemical structures cannot be represented simply by space curves. Hence, in order to represent and convert such entities, we use SMILES that are designed to describe the structure of chemical species using short ASCII strings. SMILES is an abbreviation for "Simplified Molecular Input Line Entry System." There are five general rules to convert a chemical structure into a SMILES string, Figure 2.5.

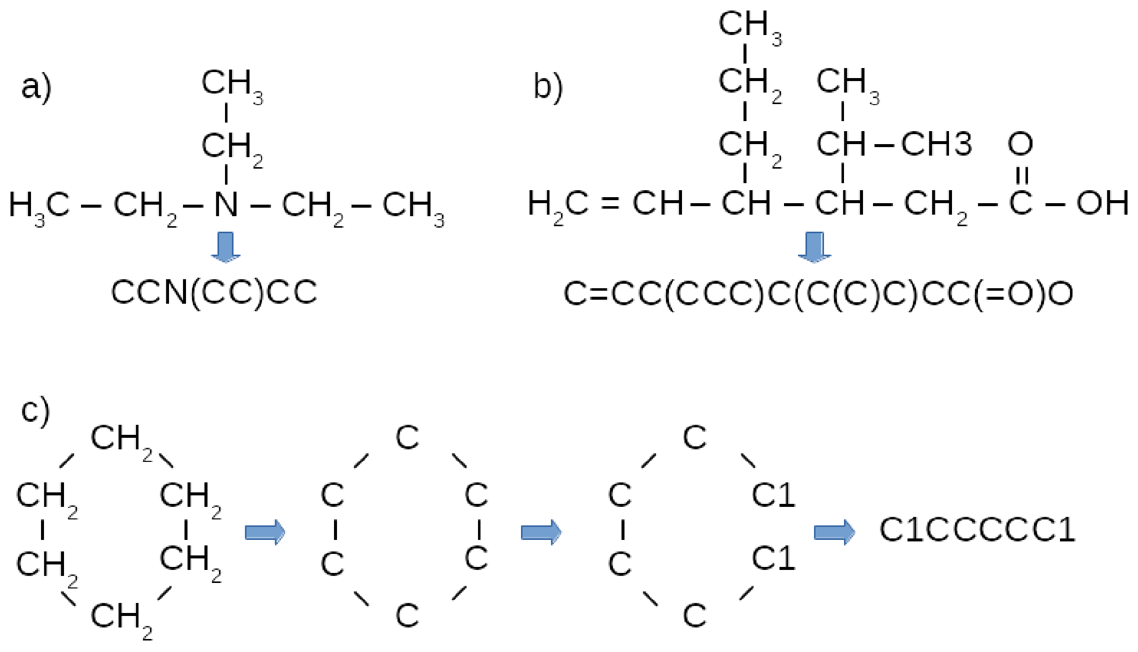


Figure 2.5: Rules for SMILES: a) A simple branch. b) A branch with sub-branches, and different types of bonds. c) Steps for rings

1. Atoms and Bonds

   Atoms are the symbols of elements in the periodic table, and bonds are denoted as shown below:

   - Single bond: - or none
   - Double bond: =
   - Triple bond: #
   - Aromatic bond: *
   - Disconnected structures: .

2. Simple Chains

   Simple Chains are represented by combining atoms and bonds. Hydrogen atoms are omitted when converting a structure into a SMILES string.

3. Branches

   The atom from a chain that has branches appears before a parentheses, followed by the branch between parenthesis, Figure 2.5 a). If it is connected not by a simple bond, the bond symbol appears within the parentheses, Figure 2.5 b).

4. Rings

   SMILES first breaks a ring into a chain structure, then the ring structures are identified by using numbers to identify the opening and closing ring atom, Figure 2.5 c). Normally, a ring has different valid SMILES, and SMILES software would output a unique SMILES string when input a valid SMILES for the same chemical structure.

5. Charged Atoms

SMILES software understands the number of possible connections that an atom can have, and charges on an atom can be used to override this information. A charged atom is followed by brackets which enclose the charge on the atom. e.g. CCC(=O)O{-1}

The same approach can be used as a general purpose method of identifying a branched object. Atoms are just the symbol of names of individual points on a branched object. Bonds are the connecting patterns, then the other rules for SMILES follow. In this manner, a “SMILES-like” string can be defined to expand the slender space curve into branched objects; e.g. we can define a branched object as a collection of space curves. Each space curve has its own **RD** and **SP**. By indexing the points on a space curve, we can have something like $S_1S_1S_1(S_2S_2(S_3)S_2)S_4S_4$ to obtain a “branched space curve”, which can be converted into 1D and 3D representations.

2.2 Multi-scale Modeling

2.2.1 Masks

Now, we have math-based (space curve) and rule-based (SMILES) conversions between 1D and 3D. In order to make the models for any object, we propose the Mask concept that works in both math-based and rule-based conditions.

Masked Space Curve

In the math-based perspective, any external agents can alter the properties of a contiguous length of a space curve from s to $s + n$. We label this a Mask, $M(s, s+n)$. Any number of identical or unique Masks may be associated with a space curve. In the material reference frame, the conformation of the mask is denoted as

$[\vec{\Gamma}(s, s+n), \vec{\Omega}(s, s+n)]$ and spans n base pairs. As the name suggests, the Masks may replace the $[\vec{\Gamma}(s, s+n), \vec{\Omega}(s, s+n)]$ values associated with a space curve. If a Mask is a rigid entity, then the components of the masks are described by fixed lists of internal and Cartesian coordinates. The Cartesian coordinate representation of $M(s, s+n)$ requires only a single translation and rotation to position each rigidly Masked element in the laboratory reference frame.

Masked SMILES

In the rule-based perspective, any naming conventions can change the complexity of a 1D string for s to $s+n$. We label this a Mask, $M(s, s+n)$. For example, Adenosine as a Nucleobase in DNA is labeled as "A", while Adenosine as a molecule can be also represented by a SMILES string:

C1=NC(=C2C(=N1)N(C=N2)C3C(C(C(O3)COP(=O)(O)O)O)O)N

In this case, a DNA sequence with ACGTs can be treated as a Masked representation of much longer SMILES string.

From a modeling point of view, we can mask any slender (or branched) entity from s to $s+n$ with one single entity, such as Adenosine demonstrated above, which can simplify the structure and display the overall structure with less items; or we can mask a single point on this entity with any number of items related to it, which gives us the ability of modeling an object in a more detailed structure. Thus, Masks enable multi-scale modeling.

2.3 Conclusion

In this chapter, we have described the background and the previous work of the proposed framework for Multi-dimensional and Multi-scale modeling. For Multi-dimensional modeling, our framework utilizes space curve and SMILES to accomplish the conversion between 1D and 3D. The definition of a space curve and the previous work for discrete methods for studying a space curve are described. For Multi-scale modeling, our framework utilizes the Mask concept to accomplish the conversion between coarse-grained to all-atom to continuous models. The concept of a Mask is described.

CHAPTER 3

COMPARISON OF METHODS FOR STUDYING SPACE CURVES WITH SUPER-HELIX GEOMETRY

In Chapter 2 we introduced a mathematical description of the oriented space curve by differentiation and integration, and two widely used numerical algorithms, E-A and E-R. Some differences were observed when applying the E-A and E-R method to DNA and chromatin as a discrete representation of the oriented space curve. There is little research on how these methods differ [11]. The question that remains is how these methods are related, and when they are equivalent descriptions. This chapter utilizes mathematical expressions to study the relationship between these methods.

3.1 Geometry of Helices

A helix is a curve in 3D space, with the center line represented by:

$$\begin{aligned} x(s) &= a\ cos(s) \\ y(s) &= a\ sin(s) \\ z(s) &= b\ s \end{aligned} \tag{3.1}$$

The six step parameters of a helix represented in [12] are:

$$\begin{aligned} Sh &= \gamma sin(Tw_o s) \qquad Ti = \kappa sin(Tw_o s) \\ Sl &= \gamma cos(Tw_o s) \qquad Ro = \kappa cos(Tw_o s) \\ Ri &= Ri_o \qquad Tw = Tw_o + \tau \end{aligned} \tag{3.2}$$

Two types of Helices are defined in [12], Torsion Helix and Shear Helix. A Torsion Helix (TH) is defined when $\gamma = 0$, and $\tau \neq 0$ is the torsion. A Shear Helix (SH) is defined when $\tau = 0$ and $\gamma \neq 0$ is the shear. Note that when both γ and τ are zero, $Tw_o \neq 0$ and $\kappa \neq 0$, a helix becomes a circle. If $Tw_o \neq 0$, the model is a circular DNA.

The following analysis begins with $\gamma = \tau = 0$, which is a circle, then studies the case of TH and SH, respectively.

3.2 Methods

With the formulas of Helices for both $\mathbf{RD}(s)$ (Equation 3.1) and $\mathbf{SP}(s)$ (Equation 3.2), we have set up three cases to study. The first is a circle, which can be treated as a helix with no shear, $\gamma = 0$ or torsion $\tau = 0$. We set 70 steps per turn. In order to make it similar to a circular DNA, we added a constant Twist (average about 32.5° for DNA). As we also want the circle to loop back exactly with the director frames, we set the Twist value as $\frac{360^\circ \times 6}{70} \approx 30.86^\circ$. For the Shear helix, there is no torsion $\tau = 0$, and we set $\gamma = 0.3$, and set a constant Twist of 32.5°. For the Torsion Helix, there is no shear $\gamma = 0$, and we set $\tau = 0.5$, and set a constant Twist of 32.5°. For all these three cases, we have 70 steps per turn, and we modeled two turns to study each case.

By studying each case, we used Equations 3.1 and 3.2 to obtain $\mathbf{RD}_{math}$ and $\mathbf{SP}_{math}$. With $\mathbf{RD}_{math}$, we use E-A and E-R methods to convert it into $\mathbf{SP}(s)$, which are $\mathbf{SP}_{E-A}$ and $\mathbf{SP}_{E-R}$, respectively. With $\mathbf{SP}_{math}$, we use E-A and E-R methods to convert it into $\mathbf{RD}(s)$, which are $\mathbf{RD}_{E-A}$ and $\mathbf{RD}_{E-R}$, respectively. Then we analyze the results by plotting the different between $\mathbf{SP}_{E-A}$, $\mathbf{SP}_{E-R}$ with $\mathbf{SP}_{math}$, and by RMSD of all the atoms between $\mathbf{RD}_{E-A}$, $\mathbf{RD}_{E-R}$ with $\mathbf{RD}_{math}$.

3.3 Analysis and Results

3.3.1 Circle

With the help of mathematical formulas, we are able to obtain step parameters ($\mathbf{SP}(s)$) of 1-base-pair step, 2-base-pair steps, 1/2-base-pair steps, or any multiples of base pair steps. We can also obtain the corresponding continuous mathematical $\mathbf{RD}(s)$ of the helix.

We then converted 1-base-pair, 1/2-base-pair, 1/4-base-pair, 1/8-base-pair Step Parameters ($\mathbf{SP}_{math}$) into Director Frames ($\mathbf{RD}(s)$) using E-A and E-R methods, and compare with the circle calculated mathematically. Figure 3.1 shows the positions by the E-A method ($\mathbf{RD}_{E\text{-}A}$) and the mathematical determined circle ($\mathbf{RD}_{math}$) before fitting. As shown in the figure, although all the "circles" start at the exact same position, there is a deflection of the circle by the E-A method from the x-z plane that does not exist for the mathematical determined circle. The same deflection happened with the E-R method.

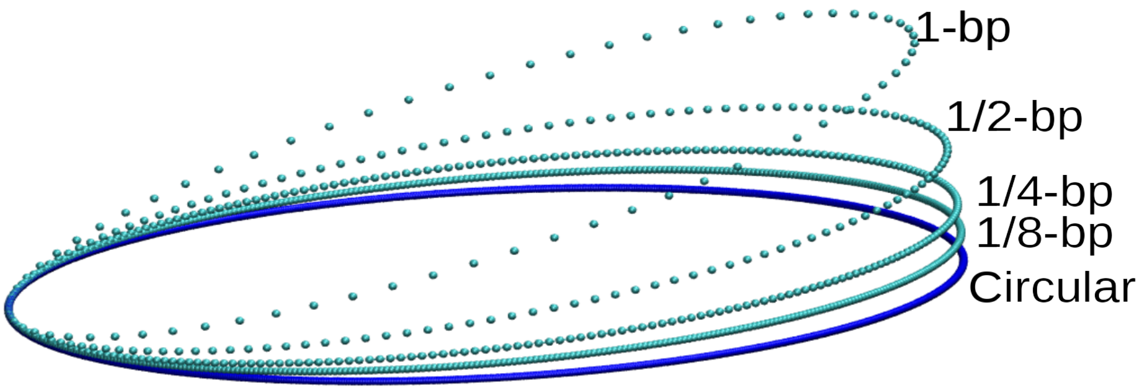


Figure 3.1: Center line of converted 1-base-pair, 1/2-base-pair, 1/4-base-pair, 1/8-base-pair step parameters using E-A method ($\mathbf{RD}_{E-A}$), compare with mathematical decided circle ($\mathbf{RD}_{math}$).

In order to figure out why there is a deflection, we convert the $\mathbf{RD}(s)$ of mathematical circle ($\mathbf{RD}_{math}$) into $\mathbf{SP}(s)$ using E-A ($\mathbf{SP}_{E-A}$) and E-R ($\mathbf{SP}_{E-R}$) methods with the step size of 1-base-pair. In Figure 3.2, we see that for Tilt and Roll, both $\mathbf{SP}_{E-A}$ (red) and $\mathbf{SP}_{E-R}$ (blue) seem to fit with SP_{math} (black). The E-A method also agrees with mathematics values for the Twist, Shift, Slide and Rise, but the E-R method does not. Because **SP**s for a circle are just pure sine and cosine functions or constant values, we can study only one period for better observation.

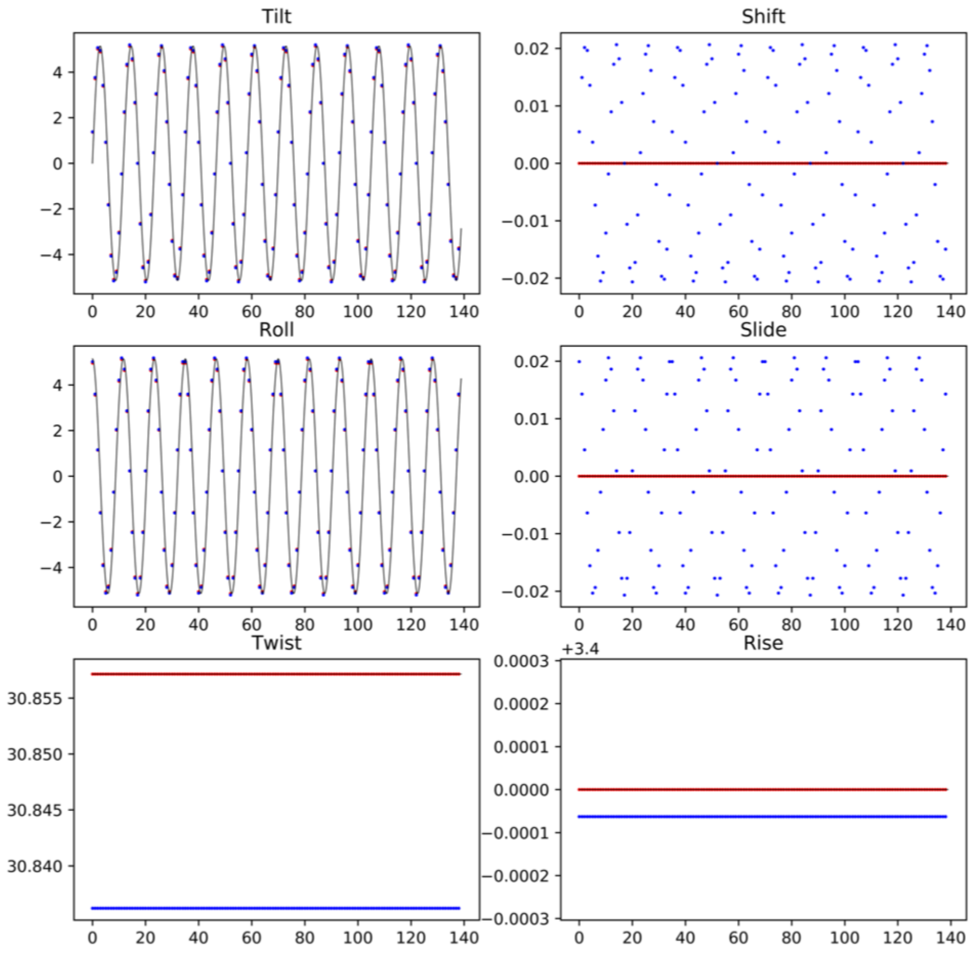


Figure 3.2: Step parameters by E-A method ($\mathbf{SP}_{E-A}$) (red), E-R method ($\mathbf{SP}_{E-R}$) (blue), and mathematics (SP_{math}) (black) of a circular DNA.

In Figure 3.3, we see that although in Figure 3.2, E-A and E-R methods seem to agree with the mathematics of Tilt and Roll, they actually differ. We recall that, mathematically, $\mathbf{SP}(s)$ represents a continuous space curve, but E-A and E-R are discrete, and they both use a mid-step plane to construct a space curve. Hence, the $\mathbf{SP}(s)$ of E-A and E-R methods are described at the "half-step." We shifted the $\mathbf{SP}_{math}$ of mathematics by half steps and plotted as shown in Figure 3.4, with the half-step shifted $\mathbf{SP}_{E-A}$ mapping with $\mathbf{SP}_{math}$ exactly. We also obtained a sine and cosine image for Shift and Slide of the E-R method with small amplitude. To verify that the E-A method agrees exactly with the mathematics model, and to see how the E-R method is different from the mathematics model, we plotted the difference of E-A and E-R methods with the mathematics values, as shown in Figure 3.5. This figure shows that $\mathbf{SP}_{E-A}$ does agree with $\mathbf{SP}_{math}$ exactly, but the E-R method differs at every step parameter. There is a clear pattern in this plot showing that the E-R method converts these $\mathbf{RD}(s)$ for a circle to the $\mathbf{SP}(s)$ for a helix with shear and torsion. As a result, if we convert the mathematical values with a phase of half steps of $\mathbf{SP}_{math}$ for a circle to $\mathbf{RD}_{E-A/E-R}$ using E-A and E-R methods, E-A would give exactly the same circle as the mathematics one, and E-R would give a helix. Figure 3.6 shows the results agree with this conclusion, that the RMSD between the E-A method and mathematics values is 0 without fitting, and E-R gives a RMSD of 3.48 $\AA$. The question arises is that why this happens. We have found the answer to this question describes in the following context.

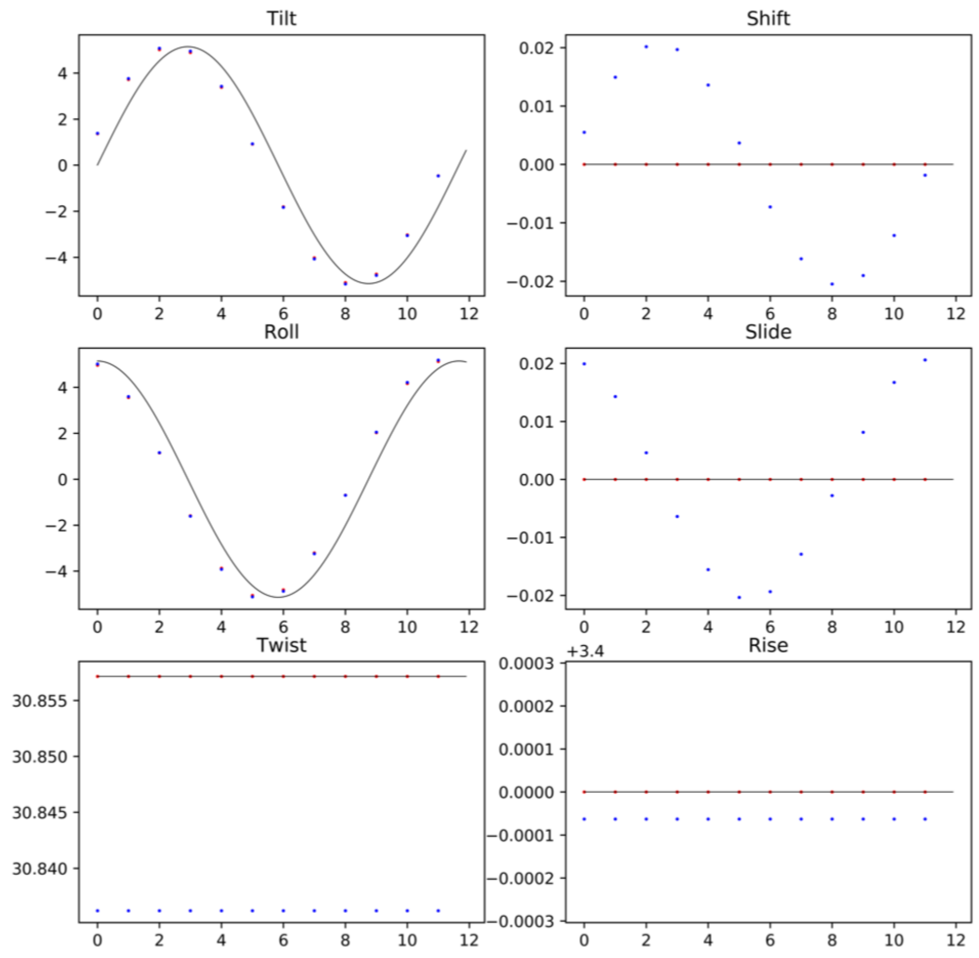


Figure 3.3: $\mathbf{SP}_{E-A}$ (red), $\mathbf{SP}_{E-R}$ (blue), and $\mathbf{SP}_{math}$ (black) of a circular DNA in one period.

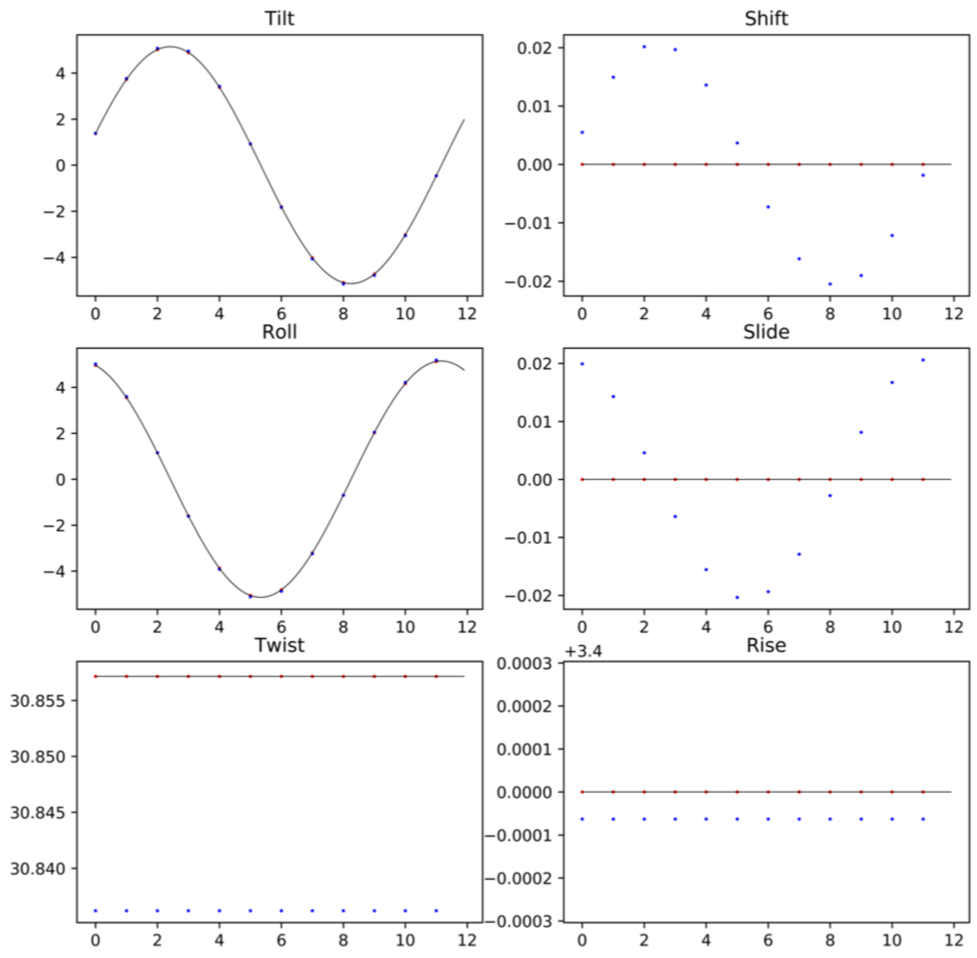


Figure 3.4: $\mathbf{SP}_{E-A}$ (red), $\mathbf{SP}_{E-R}$ (blue), and $\mathbf{SP}_{math}$ (black) with one period, $\mathbf{SP}_{math}$ is shifted by half steps.

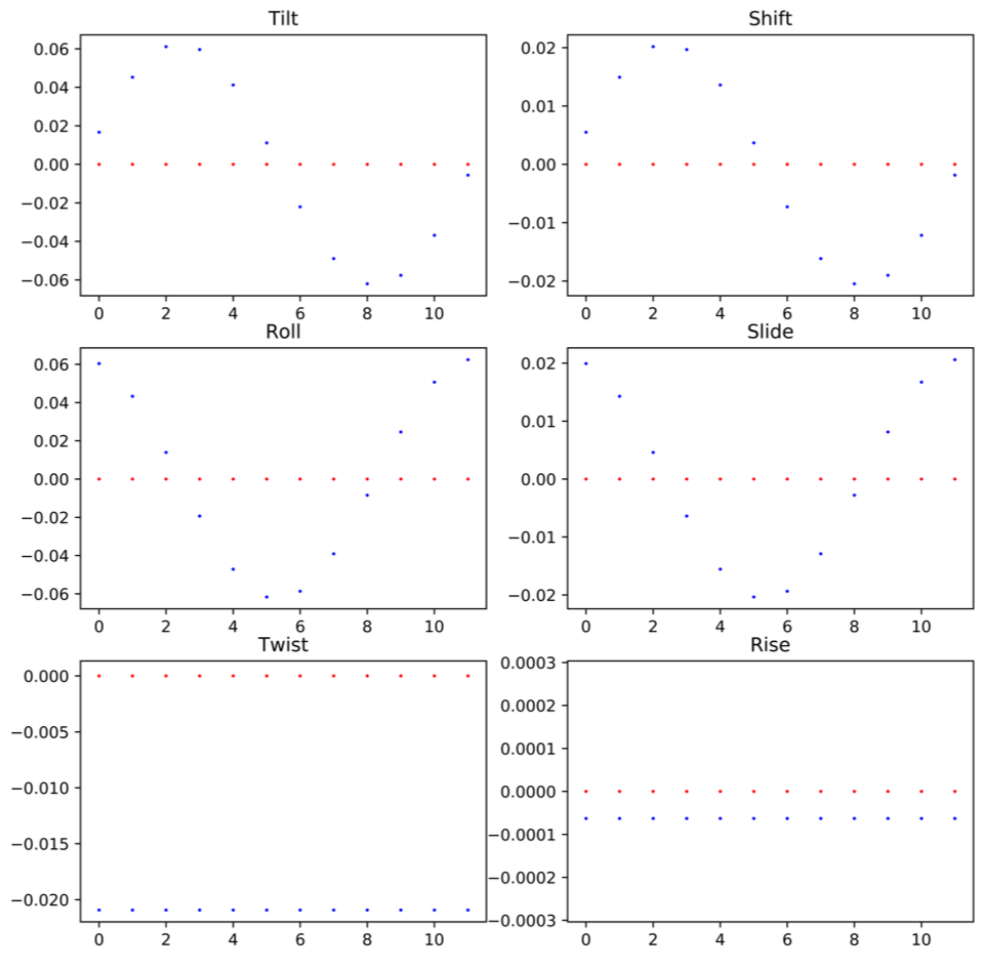


Figure 3.5: Difference of E-A method with mathematical values, $\mathbf{SP}_{E-A}$ - $\mathbf{SP}_{math}$ (red), and E-R method with mathematics values, $\mathbf{SP}_{E-R}$ - $\mathbf{SP}_{math}$ (blue).

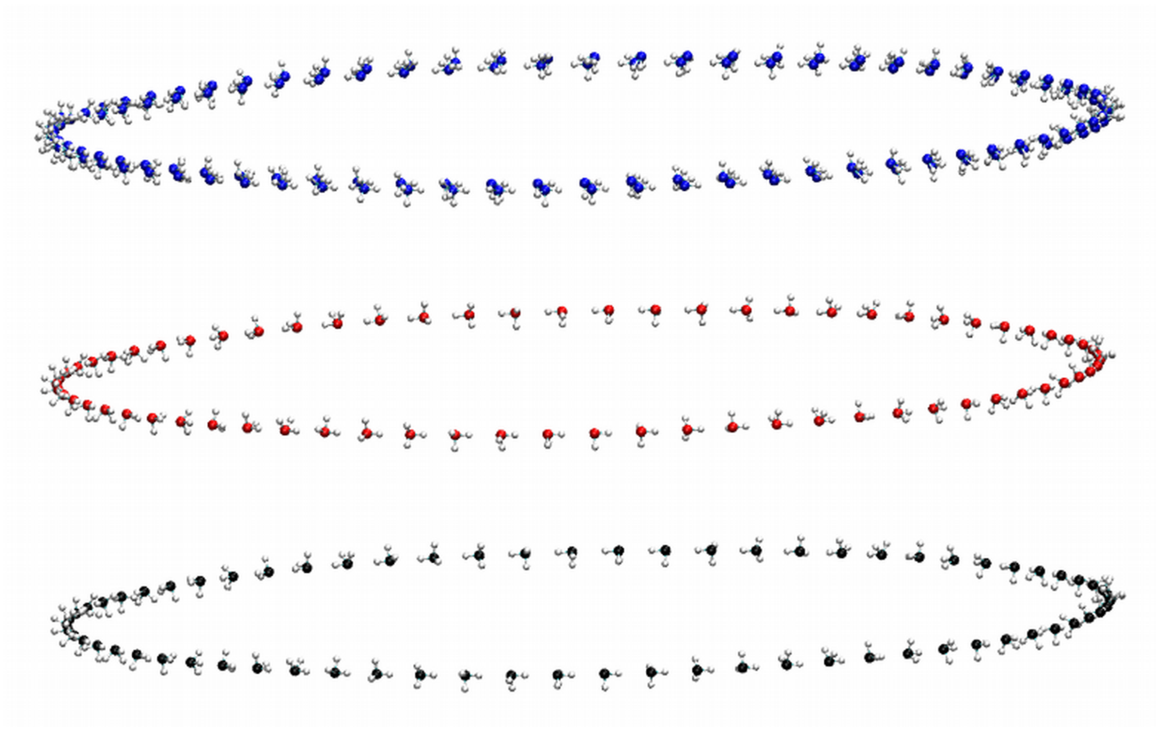


Figure 3.6: Convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-R}$ (blue), and convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-A}$ (red) of a circular DNA with discrete methods and compare with $\mathbf{RD}_{math}$ (black). (The representation in the figure are shifted along z-axis to have a more clearly visualization.)

Recalling the mid-step plane constructions by E-A and E-R methods mentioned in Chapter 2, the E-A method first converts Tilt and Roll into bend as one rotation, and applies the half bend and half Twist to construct the mid-step-triad; the E-R method first calculates the rotation from the current step to the next step, and takes a square root of this rotation as the rotation to the mid-step plane. Then consider the case of a circle, shown in Figure 3.7. If we do not consider Twist, for the mathematics continuous construction, the “mid-step” should be on the circle, and d_3 has the direction of the tangent to the circle, and d_1 points to the center of the circle. For the E-A and E-R methods, d_3 has the same direction of the tangent at the half step point of the circle, and d_1 also points to the center of the circle, with d_2 being the cross product of d_3 and d_1. In this case, the $\mathbf{D}$ happens to be the same between discrete construction and continuous construction. However, when applying the constant Twist to this **D**, the E-A method takes half of the twist, so after twisting, the **D** is still the same with mathematics construction; for the E-R method, it takes the square root to the rotations from step i to step $i+1$ to calculate the mid-step plane. The rotation from step i to step $i+1$ can be treated as a rotation along the d_2 axis for Roll degrees and a rotation along the d_3 axis for the Twist degrees. By applying the Rotation matrices mentioned in Chapter 2, we can calculate the rotation and the square root of it. Then we can obtain that $\sqrt{R_x(Roll)R_z(Twist)} \neq R_x(\frac{Roll}{2})R_z(\frac{Twist}{2})$, and thus obtain the difference between the E-R method with mathematical values.

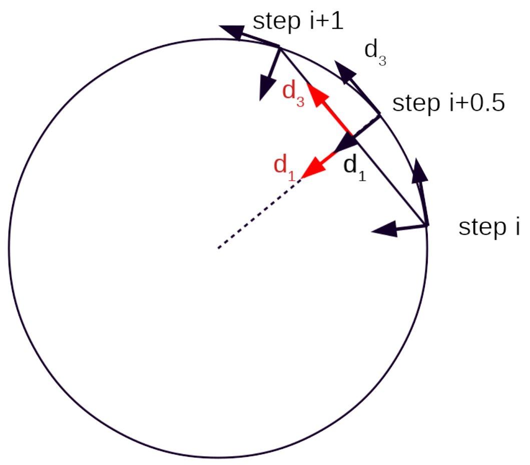


Figure 3.7: Without considering the constant Twist applied, the mathematics step i+0.5 (black), and the mid-step constructed by discrete methods (red).

As a result, we have two conclusions about this mid-step construction: one is that when Twist is zero, E-A and E-R methods should agree with the mathematical solutions, and the other is that the $\mathbf{RD}_{E-A}$ and the $\mathbf{RD}_{math}$ happens to be the same for circles, both E-A and E-R methods will give different answers compared with the mathematical solution when applied to a helix. When Twist $= 0$, we convert the **SP** to **RD** and obtain E-A, E-R and mathematics with exactly the same values (RMSD is 0), as shown in Figure 3.8. The analyses for helices are described in the following sections.

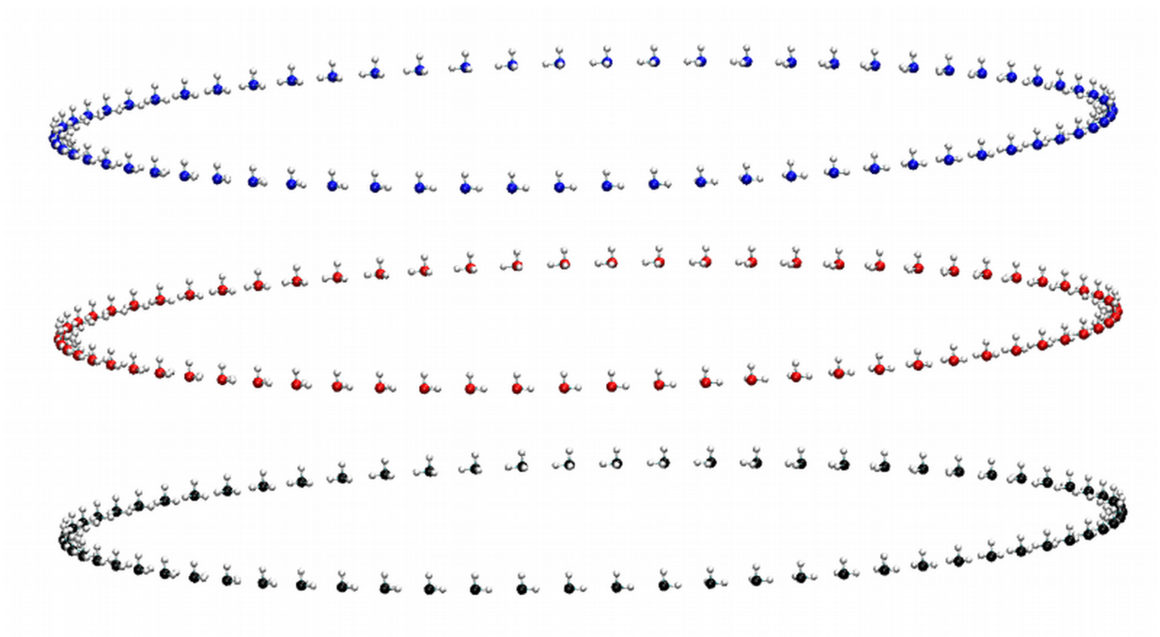


Figure 3.8: Convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-R}$ (blue), and convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-A}$ (red) of a circular DNA with discrete methods at Twist = 0 and compare with $\mathbf{RD}_{math}$ (black). (The representation in the figure are shifted along z-axis to have a more clearly visualization.)

We also calculated the RMSD between $\mathbf{RD}_{E-A}$, $\mathbf{RD}_{E-R}$ compared with $\mathbf{RD}_{math}$ as shown in Table 3.1. We see that without the half-step phase, even though the RMSD for center atoms are almost the same for the E-A method, the RMSD for all atoms are different.

Table 3.1: RMSD between $\mathbf{RD}_{E-A}$, $\mathbf{RD}_{E-R}$ compared with $\mathbf{RD}_{math}$ with different step sizes, with and without the half-step phase, and with and without RMSD fitting for circular DNA. Under Selection, "all" calculates the RMSD between all the atoms $\mathbf{RD}$ that includes director frames, "CA" calculates center atoms $\vec{r}(s)$ only.

Circular DNA

Step Size	RMSD Fitting	Selection	With Phase		Without Phase	
			E-A (Å)	E-R (Å)	E-A (Å)	E-R (Å)
1	yes	all	0	1.68	0.19	1.69
		CA	0	1.68	6.22×10^{-6}	1.68
	no	all	0	3.48	12.46	12.67
		CA	0	3.48	12.46	12.66
$\frac{1}{2}$	yes	all	0	0.42	0.10	0.43
		CA	0	0.42	5.29×10^{-6}	0.42
	no	all	0	0.87	6.24	6.22
		CA	0	0.87	6.24	6.22
$\frac{1}{4}$	yes	all	0	0.11	0.05	0.12
		CA	0	0.11	5.96×10^{-6}	0.11
	no	all	0	0.22	3.12	3.11
		CA	0	0.22	3.12	3.11
$\frac{1}{8}$	yes	all	0	0.03	0.02	0.04
		CA	0	0.03	6.03×10^{-6}	0.03
	no	all	0	0.05	1.56	1.56
		CA	0	0.05	1.56	1.56

3.3.2 Shear Helix

By the analyses of the circles, we already know that **SP** for mathematics and **SP** for the discrete methods have a half-step differential difference at where the parameters are being calculated, and for the helices (the circle can be treated as a helix with zero pitch), the **SP** are sine, cosine, or constant functions. Hence, from this section, we can study the E-A and E-R methods for helices after applying the half-step shift, and we can study the **SP** values in one period of sine or cosine functions.

A shear helix is the helix with $\tau = 0$. If we analyze it like we do in Figure 3.7, we can see that the tangent of the curve at the half step is different than the mid-step on the chord. With the discrete method, the mid-step triad is a matter of defination rather than observation, but as long as the method is self-consistent in both directions (**SP** to **RD**, and **RD** to **SP**), it could be utilized to represent a 3D space curve. For the case of the shear helix, the E-A method, E-R method, and the mathematical method would generate different **RD** with the same **SP**. In Figure 3.9, we obtained these differences. As the mid-step plane of the E-R method is more similar to the mathematical values at step $i + 0.5$, we obtained a smaller RMSD between the E-A method and mathematics, which is 0.22 $\AA$; a larger RMSD between the E-R method and mathematics, which is 4.58 $\AA$. RMSD reported are without fitting and with all the atoms in both cases.

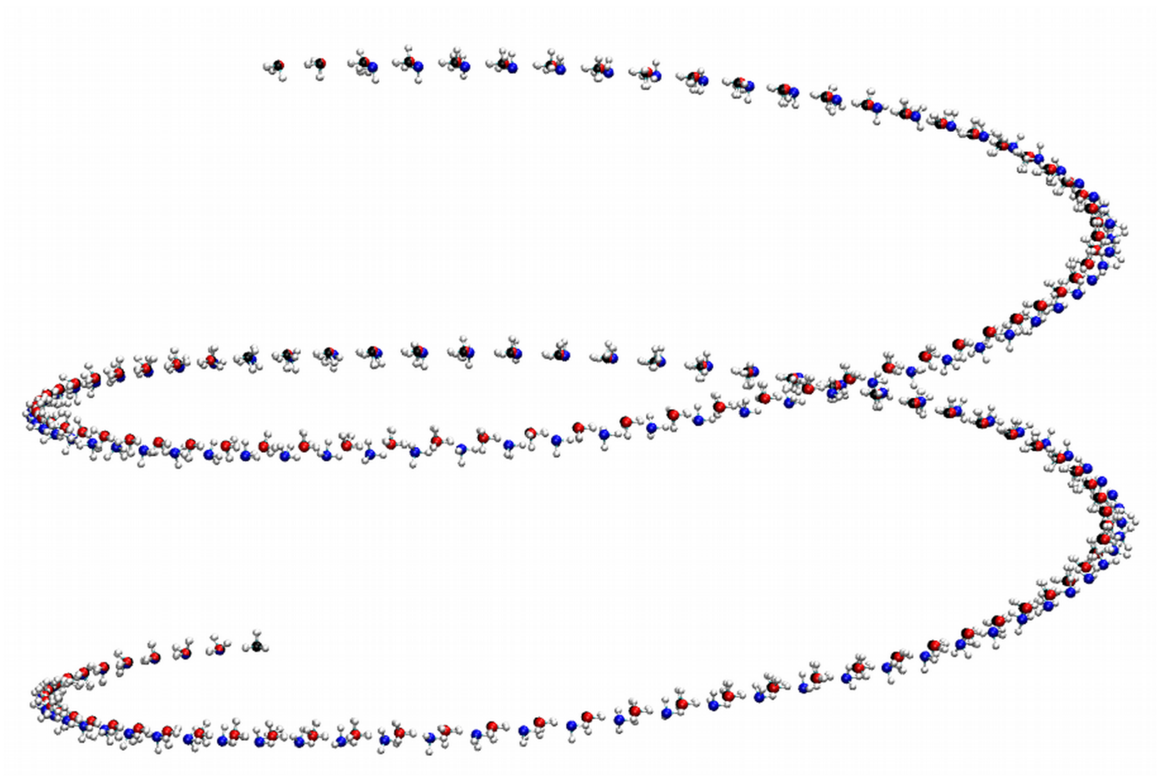


Figure 3.9: Convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-R}$ (blue), and convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-A}$ (red) of a Shear Helix with discrete methods and compare with $\mathbf{RD}_{math}$ (black).

At the same time, although the three methods are different, when the step size is getting smaller, the mid-steps between two steps of discrete methods should converge to the mathematics at step $i + 0.5$. Hence, if we reduce the step size, E-A and E-R methods should converge to the mathematical values. In Figure 3.10, we obtained the converge, and the RMSD is less than 0.22 for both E-A and E-R methods compared with mathematical values.

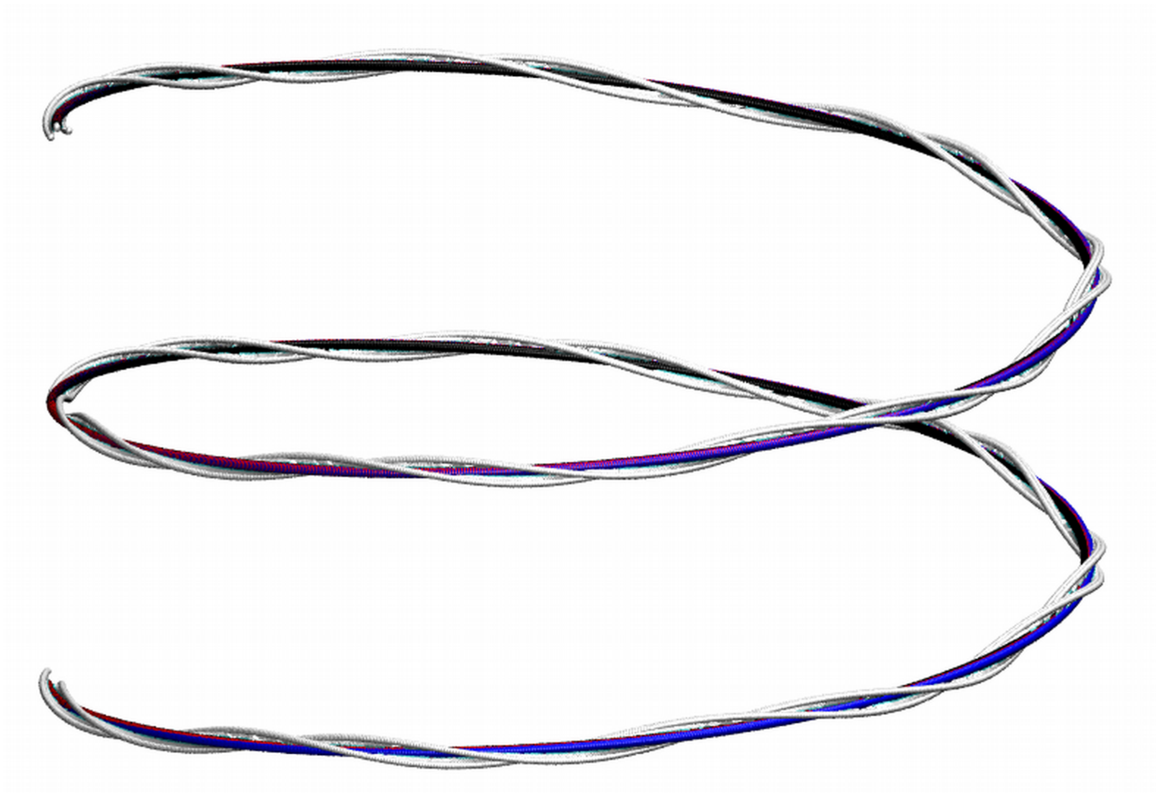


Figure 3.10: Convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-R}$ (blue), and convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-A}$ (red) of a Shear Helix with discrete methods and 1/16 base-pair steps and compare with $\mathbf{RD}_{math}$ (black).

RMSD analysis for Shear Helix is shown in Table 3.2. We see that for the E-A method, the RMSD remains the small difference with $\mathbf{RD}_{math}$, which arises from the difference between tangent and secant of the mid-step; for E-R method, the RMSD is getting smaller with the step-size reduced, which is due to the reduces of the difference between the mid-step triad of E-A and E-R methods.

Table 3.2: RMSD between $\mathbf{RD}_{E-A}$, $\mathbf{RD}_{E-R}$ compared with $\mathbf{RD}_{math}$ with different step sizes, with and without half-step phase, and with and without RMSD fitting for Shear Helix. Under Selection, "all" calculates the RMSD between all the atoms **RD** that include the director frames, and "CA" calculates center atoms $\vec{r}(s)$ only.

Shear Helix

Step Size	RMSD Fitting	Selection	With Phase		Without Phase	
			E-A (Å)	E-R (Å)	E-A (Å)	E-R (Å)
1	yes	all	0.16	2.21	0.27	2.22
		CA	0.15	2.21	0.15	2.21
	no	all	0.22	4.58	15.91	15.54
		CA	0.21	4.57	15.92	15.55
$\frac{1}{2}$	yes	all	0.16	0.60	0.19	0.61
		CA	0.15	0.60	0.15	0.60
	no	all	0.22	1.21	7.95	7.76
		CA	0.21	1.20	7.96	7.78
$\frac{1}{4}$	yes	all	0.16	0.23	0.17	0.06
		CA	0.15	0.22	0.15	0.22
	no	all	0.22	0.40	3.94	3.89
		CA	0.21	0.40	3.96	3.90
$\frac{1}{8}$	yes	all	0.16	0.17	0.16	0.17
		CA	0.15	0.16	0.15	0.16
	no	all	0.22	0.25	1.94	1.93
		CA	0.21	0.24	1.96	1.94

3.3.3 Torsion Helix

A Torsion helix is a helix with $\gamma = 0$ and $\tau \neq 0$. Similar to SH, both E-A and E-R methods are different from the mathematics; however, with the τ applied, notice that this τ is applied at the step i for the mathematics equations, and this τ is continuously applied even at step $i + 0.5$ for the mathematics method, while for the discrete methods, this τ is halved at the mid-step plane, so we will obtain a deflection by applying E-A and E-R methods. In Figure 3.11, we see that both E-A and E-R methods are deflected compared with mathematical values, and the RMSD without fitting is 3.91 $\mathring{A}$ and 4.80 $\mathring{A}$ for E-A and E-R methods, respectively. The RMSD after fitting is 0.64 $\mathring{A}$ and 1.43 $\mathring{A}$ for E-A and E-R methods with mathematics, which validates that the "shape" of the curve remains a helix. RMSD reported are without fitting and with all the atoms in both cases.

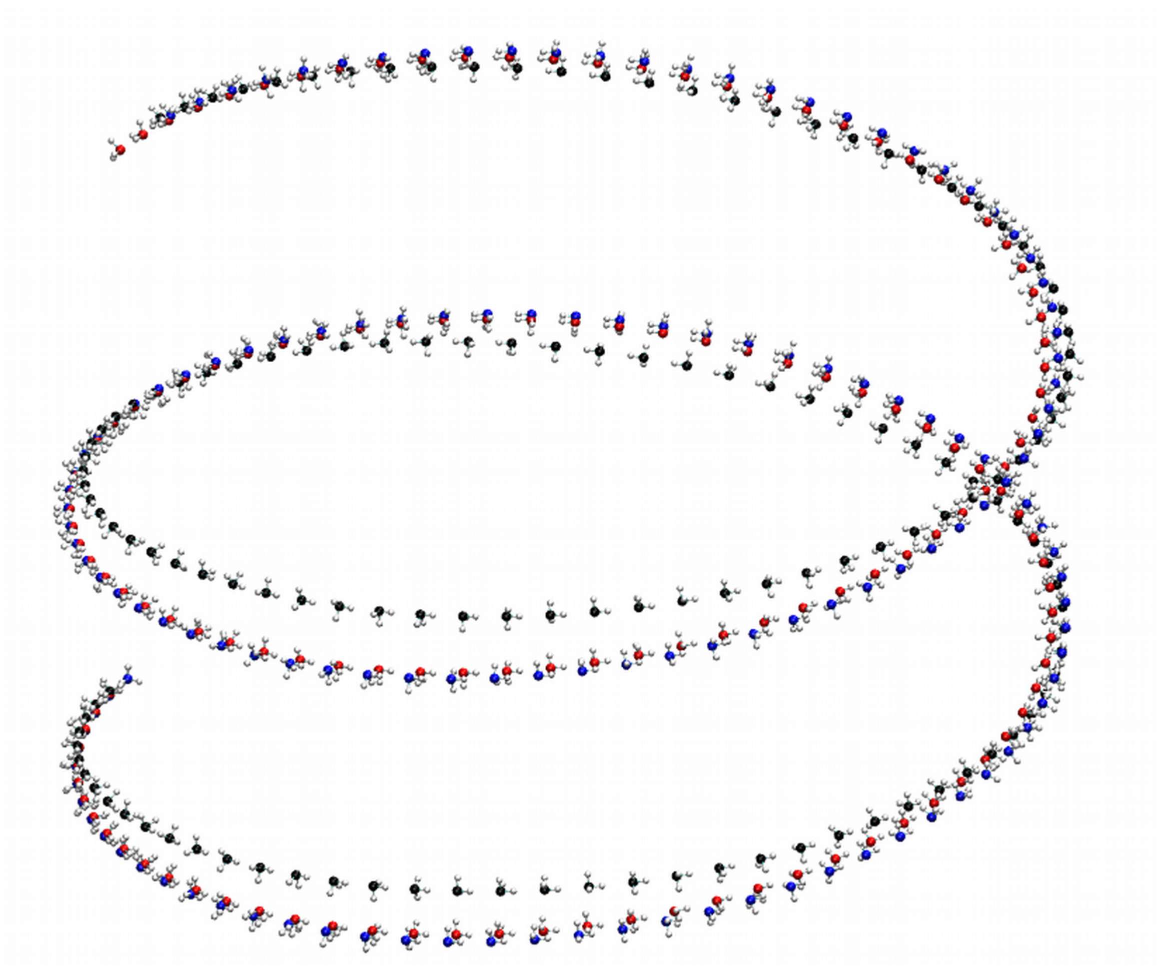


Figure 3.11: Convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-R}$ (blue), and convert $\mathbf{SP}_{math}$ to $\mathbf{RD}_{E-A}$ (red) of a Torsion Helix with discrete methods and compare with $\mathbf{RD}_{math}$ (black).

We also studied the **SP** for SH and TH obtained from $\mathbf{RD}_{math}$ using these methods. They all show similar patterns that both E-A and E-R methods will obtain small torsion and shear that are different from the mathematics values. We have provided the **RD** representation of how E-A and E-R methods converge to the mathematics values in the section above. Here, we provide a **SP** representation of how they converge. Figure 3.12 shows that by reducing the base pair's step size, the E-A method is converging to the mathematics method. (The E-R method has the same behavior, which is not provided here.)

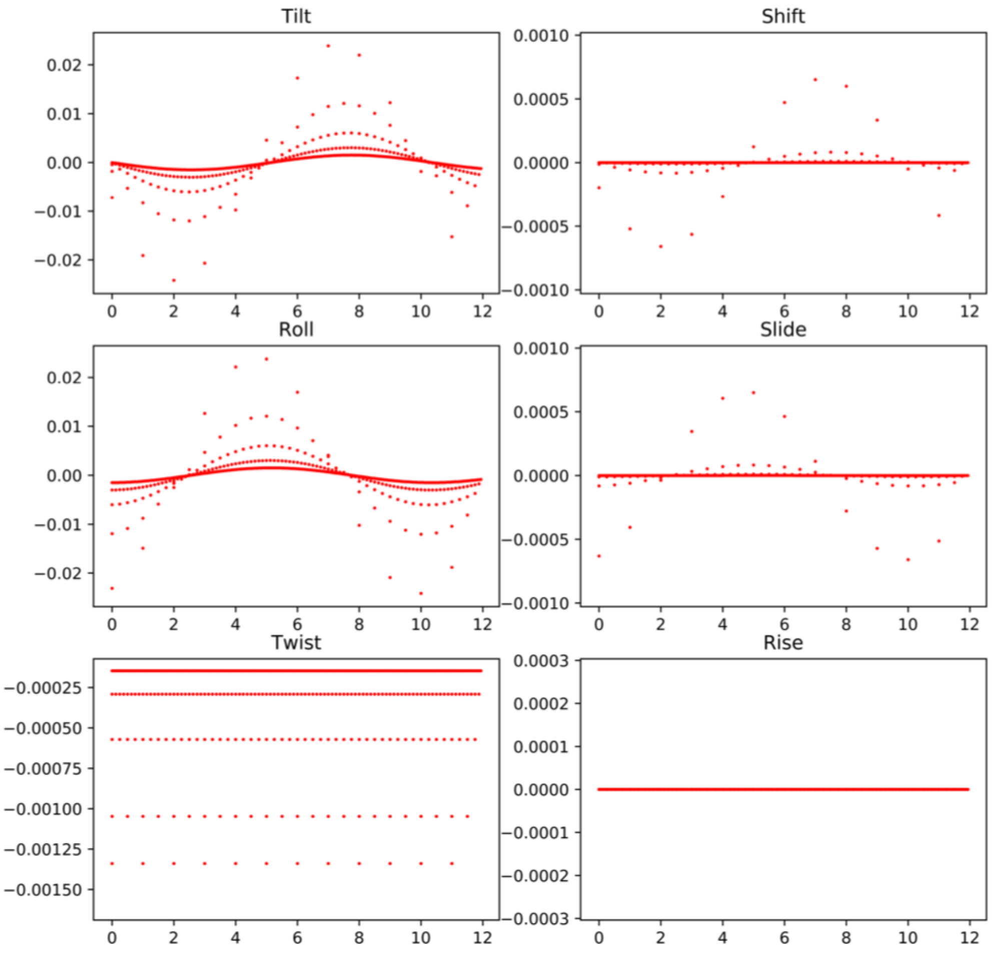


Figure 3.12: The $\mathbf{SP}_{E-A}$ converted from $\mathbf{RD}_{math}$ with 1-bps, 1/2-, 1/4-, 1/8-, and 1/16-base pair steps. Less dots in the plot indicate larger base pair steps.

The RMSD analysis for Torsion Helix is shown in Table 3.3. We notice that for Torsion Helix, RMSD does not reflect the fact that all the methods converge with the step size getting smaller. That is because for Torsion Helix, the deflection caused by the torsion produces more errors than the choice of the mid-step triad produced.

Table 3.3: RMSD between $\mathbf{RD}_{E-A}$, $\mathbf{RD}_{E-R}$ compared with $\mathbf{RD}_{math}$ with different step sizes, with and without half-step phase, and with and without RMSD fitting for Torsion Helix. Under Selection, "all" calculates the RMSD between all the atoms **RD** that include the director frames, and "CA" calculates center atoms $\vec{r}(s)$ only.

Torsion Helix

Step Size	RMSD Fitting	Selection	With Phase		Without Phase	
			E-A (Å)	E-R (Å)	E-A (Å)	E-R (Å)
1	yes	all	0.64	1.43	0.68	1.44
		CA	0.64	1.43	0.64	1.43
	no	all	3.91	4.80	16.70	16.07
		CA	3.91	4.80	16.69	16.06
$\frac{1}{2}$	yes	all	0.64	0.13	0.65	0.17
		CA	0.64	0.13	0.64	0.13
	no	all	3.93	3.71	9.13	8.82
		CA	3.92	3.71	9.12	8.82
$\frac{1}{4}$	yes	all	0.64	0.51	0.64	0.51
		CA	0.64	0.51	0.64	0.51
	no	all	3.93	3.85	5.76	5.67
		CA	3.93	3.85	5.76	5.66
$\frac{1}{8}$	yes	all	0.64	0.61	0.64	0.61
		CA	0.64	0.61	0.64	0.61
	no	all	3.93	3.91	4.51	4.48
		CA	3.93	3.91	4.50	4.48

3.4 Conclusions

In this chapter, we have analyzed the difference between discrete methods (E-A and E-R) with the continuous mathematical description of oriented space curve that are helical. We have found that the main difference arises from the mid-step plane, which is applied for the construction in discrete methods but not in the continuous description. The Step Parameters for mathematics representation are half-steps ahead of the E-A and E-R methods, which should be kept in mind when analysing **SP**; otherwise, RMSD will show the two structures are similar, but this error can be avoided. The construction of the mid-step plane is also different between E-A and E-R methods, so the **SP** cannot be exchanged, and neither can it be for the **RD**.

This chapter also shows that as the step becomes smaller, the three methods will eventually converge to the mathematics representation. With this analysis, we are then able to convert a set of sequential points to a space curve, and by applying methods such as Fourier Transform to obtain equivalent mathematical representation, and by integration and differentiation, we can then have a set of equations of **SP** mathematically to analyze.

CHAPTER 4

MULTI-DIMENSIONAL AND MULTI-SCALE MODELING OF MAGIC SNAKE

4.1 Magic Snake Introduction

A Magic Snake [13] is a 3D object consisting of n congruent wedges, shown in Figure 4.1. Each wedge is a right isosceles triangular prism (half of a cube), that contains six vertices and five faces (two square faces, two triangular faces, and a rectangular face). Typically, a Magic Snake toy sold in stores consists of 12*n wedges (n is an integer greater than 1), and the adjacent wedges are connected by the center of a square face and can be rotated by integer multiples of 90 degrees around the center axis normal to the square. By rotating the square face centers (S_{1c} and S_{2c}) between each neighboring wedge, a Magic Snake can be formed into different shapes.

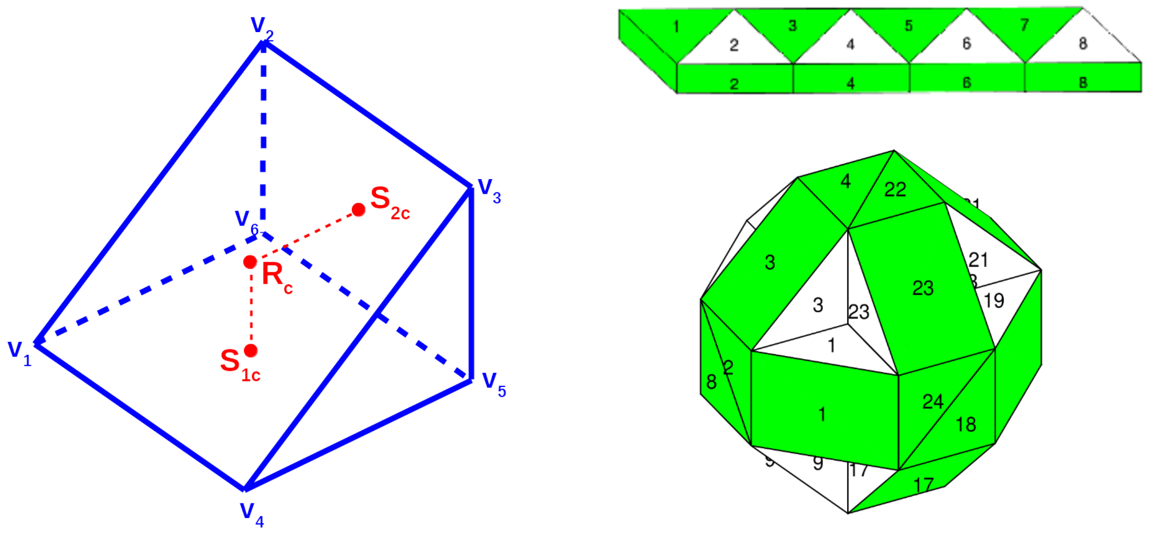


Figure 4.1: Left: A wedge is the basic unit of a Magic Snake. It has 6 vertices and 5 faces. We define the center of the rectangle face as R_c, the two square face center as S_{1c} (entry) and S_{2c} (exit). Right: Two examples of a Magic Snake with a different number of wedges and different structures.

Compared with the Rubik's Cube that has been researched [14], the Magic Snake has much more combination possibilities [15]. However, there is little research related to the Magic Snake, or use of its properties to enlighten researchers in various fields, such as protein folding [16, 17] and robot designing [18, 19, 20, 21, 22]. Systematic studies of the mathematical properties and computational design of a Magic Snake are still missing.

Some mathematical properties of a Magic Snake can be found when the rotation is restricted to integer multiples of 90 degrees. Here, more general properties of a Magic Snake, which is neither limited by the number of wedges, nor by the discrete rotations, are explored. This gives us the ability to explore the infinite possibilities of conformations for a Magic Snake.

In this chapter, we first provide modeling strategies that can map the 3D structure of a Magic Snake to a 1D sequence and convert this 1D sequence to the 3D structure of a Magic Snake. Then we propose two design strategies; one is to fit a Magic Snake to any curve in space, which not only enables gamers and hobbyists to design a Magic Snake from any curve but also applies to the general design ideas, such as a robot arm design. Another design strategy is to systematically expand a complex Magic Snake to a more complex structure, which can be used to design a Magic Snake from simple structures.

4.2 Multi-dimensional Modeling of Magic Snake

In order to model a Magic Snake, we first need to describe the conformation of a Magic Snake in a simple notation. Naming Magic Snake conformations as "cat" or "dog" is descriptive, but such labels are not unique and cannot be computed. Here, we describe a Magic Snake as a sequence of rotations and then build up a 3D model based on these rotation sequences.

4.2.1 3D to 1D

A Magic Snake is a set of wedges connected together. We define the "Degree of the Junctions" of a Magic Snake, which is a list of the rotations between the adjacent wedges. To define these rotations properly, we choose the square face containing S_{2c} of the prior wedge and rotate the next wedge counterclockwise. For simplicity, we divide these rotation degrees by $\pi/2$ (90 degrees) then modulo 4, so that we can get an integer for multiples of 90 degree rotations $R = 0, 1, 2, 3$, and $R \in [0, 4)$ for any rotations, as shown in Figure 4.2. If we record these degrees in order, we get the 1D sequence $seq = [R_1, R_2, \ldots, R_n]$ that represents the 3D structure of a Magic Snake. For example, the "ball" structure shown in Figure 4.1 can be represented by $seq = [1, 3, 3, 1, 3, 1, 1, 3, 1, 3, 3, 1, 3, 1, 1, 3, 1, 3, 3, 1, 3, 1, 1, (3)]$. Notice that the last rotation in this sequence is marked in parentheses; that is because the "ball" is a closed loop (the exit of the last wedge S_{2c} is overlapped with the entry of the first wedge S_{1c}, and these two square faces are in the same plane). We add an extra rotation (the rotation taken from the last wedge to the first wedge in the Magic Snake) to describe the closed loops.

For a n wedge Magic Snake, we obtain a 1D sequence with the length of $n - 1$, and for closed loops, we have length n sequence with the "closure" rotation. Each 1D rotation sequence represents one 3D Magic Snake structure. Imagine if we are rotating the Magic Snake from the first wedge to the last, and by rotating each wedge, we obtain an exact number that tells how much and in which direction to rotate. Eventually, we will always get exactly the same structure by using the same sequence. In the same way, if the sequential path of a 3D Magic Snake is determined, we obtain one rotation sequence. However, if the overall 3D structure of a Magic Snake is the only consideration, we may have multiple rotation sequences to represent it. A simple example is the reverse of the rotation sequence.

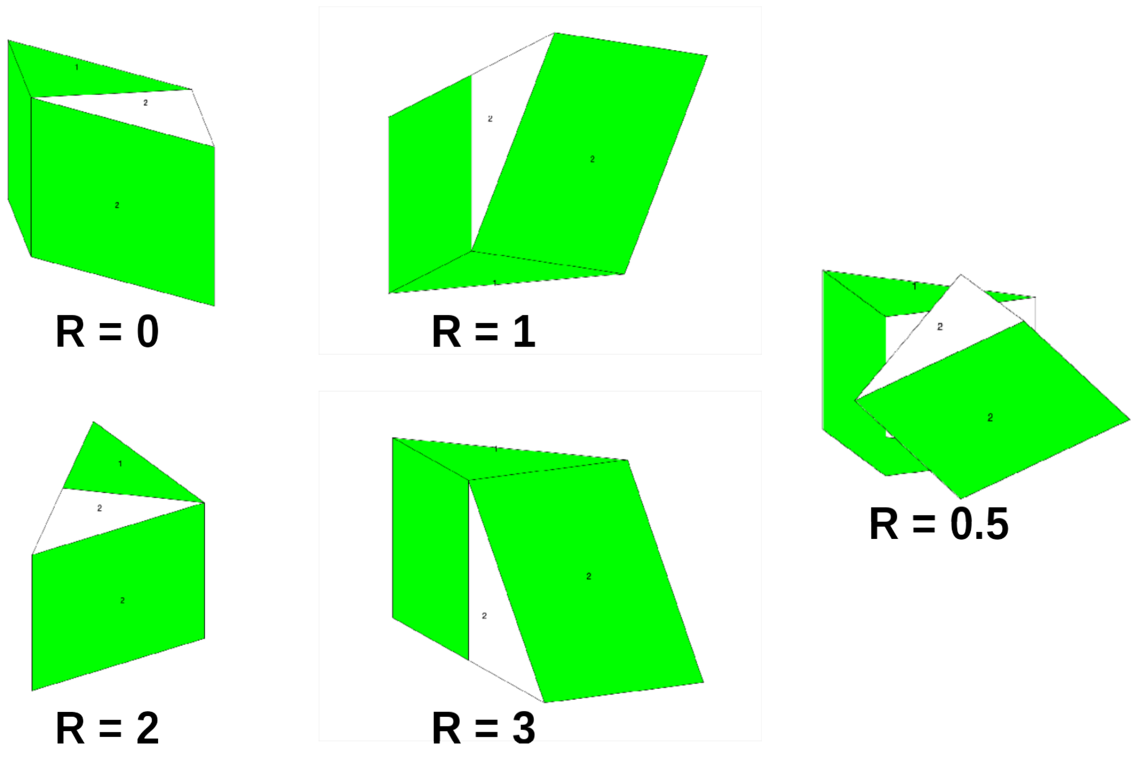


Figure 4.2: Rotations between each adjacent wedge. We define the "Degree of Junctions" as the rotate angle divided by 90 degrees and modulo 4.

Some properties of how certain structures related to these 1D sequences for integer multiples of 90-degree rotations can be found. It is useful to convert a 3D structure into a simplified form in order to understand the properties behind it. It is more productive if we can figure out more properties in a simplified model that is applied to general cases. However, 1D information is not enough to reveal all the properties of a 3D object, e.g. self-contact or loop closure. Thus, we will need to explore both 1D and 3D properties and computationally convert between 1D and 3D.

4.2.2 1D to 3D

To analyze the mathematical properties and design a 3D structure, the location of every vertex of the Magic Snake is required. There are numerous approaches to calculate the orientation and location of all wedges in a Magic Snake. Here, we describe a solution using rotation matrices described in Chapter 2.

As shown in the left in Figure 4.3, assume that we seek to calculate the location of the blue wedge relative to the red wedge.

First, add a local direction frame $\hat{d}_1, \hat{d}_2, \hat{d}_3$, where $\hat{d}_1$ is the unit vector pointing from the rectangle center (R_c) to the exit square face center (S_{2c}); $\hat{d}_2$ is the unit vector pointing from the entry square face center (S_1c) to the rectangle center (R_c); and $\hat{d}_3$ is the cross product of $\hat{d}_1$ and $\hat{d}_2$ shown on the right of Figure 4.3. Let the edges on the square face of all wedges be unit length, and let the initial wedge align with the coordinate axis, let the rectangle center (R_c) of the wedge be located at the origin, and put $\hat{d}_1, \hat{d}_2, \hat{d}_3$ along with x,y,z axis, as shown in Figure 4.4 a, so that the 3 by 3 director matrix is:

$$\underline{D}_a = \begin{bmatrix} \hat{d}_1 \\ \hat{d}_2 \\ \hat{d}_3 \end{bmatrix}$$

As shown in Figure 4.4 a.

$$\underline{D}_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Second, rotate the initial wedge along the z axis by 90 degrees and translate along the x axis by 1 unit, as shown in Figure 4.4 b. Regardless of the translation, the direction frame is now

$$\underline{D}_b = R_z(\pi/2) \cdot \underline{D}_a \tag{4.1}$$

Third, rotate the wedge along the x-axis by 180 degrees; now it is like we put the wedge at the place where the next wedge should be when the degree of the junction is 0, as shown in Figure 4.4 c. Then we have

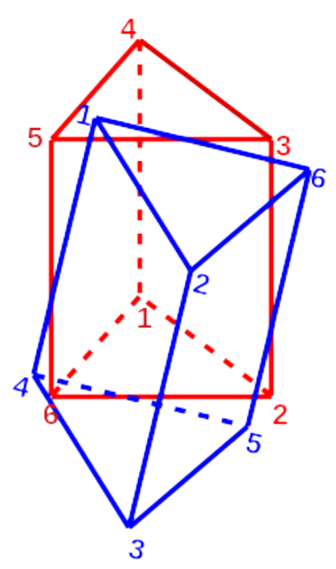
$$\underline{D}_c = (R_z(\pi/2) \cdot R_x(\pi)) \cdot \underline{D}_a \tag{4.2}$$

Finally, rotate the wedge along the x axis by θ degrees. Now, by doing this final rotation and translation, we obtain the location where the next wedge should be when we rotate the next wedge by θ degrees, as shown in Figure 4.4 d. Then we can calculate the location and orientation of it using the rotation matrices (Note that $R_x(\pi) \cdot R_x(\theta) = R_x(\pi + \theta)$):

$$\underline{D}_d = (R_z(\pi/2) \cdot R_x(\pi + \theta)) \cdot \underline{D}_a \tag{4.3}$$

That is,

$$\underline{D}_2 = (R_z(\pi/2) \cdot R_x(\pi + \theta)) \cdot \underline{D}_1 \tag{4.4}$$



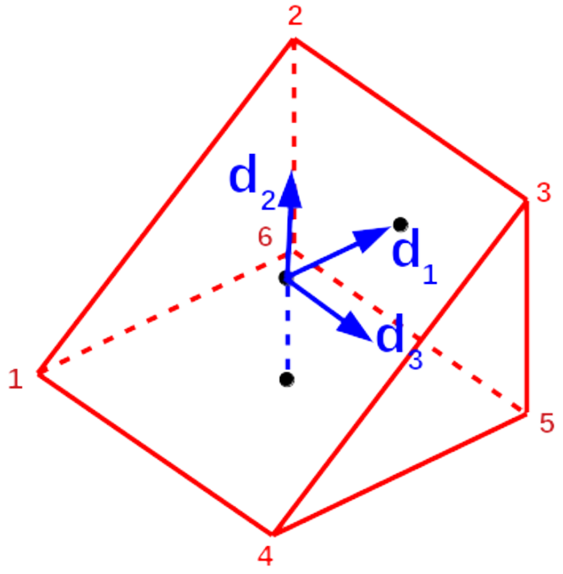


Figure 4.3: Left: Relative positioning of adjacent wedges can be described by a rotation and a translation. Right: Assignment of director frame to a wedge.

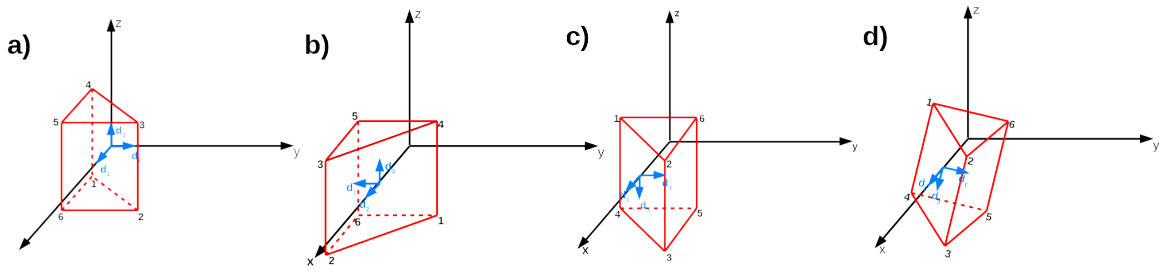


Figure 4.4: Procedure for building a Magic Snake wedge by wedge. a). The wedge (Red wedge shown on the left of Figure 4.3) is located at the origin of a coordinate system. b). Flip the initial wedge along edge 5-6, and the wedge can also be treated as rotated along the z axis 90 degrees; then translate along x axis by 1 unit. c). Rotate the wedge along the x axis. d). Rotate the wedge along the x axis by θ degrees (The wedge is now at the position of the blue wedge shown on the left of Figure 4.3).

Then we consider the translation vector $\vec{r}$. Let $\vec{r}_1 = [0, 0, 0]$, the origin. As a local translation, it travels along the x axis for 1 unit. If we add the global rotation to it, which is just the $\underline{D}$ for the last frame, then we have

$$\vec{r}_2 = \vec{r}_1 + [1, 0, 0] \cdot \underline{D}_1 \tag{4.5}$$

In the same way, we have

$$\underline{D}_3 = (R_z(\pi/2) \cdot R_x(\pi + \theta_2)) \cdot \underline{D}_2 \tag{4.6}$$

$$\vec{r}_3 = \vec{r}_2 + [1, 0, 0] \cdot \underline{D}_2 \tag{4.7}$$

and for wedge n:

$$\underline{D}_n = (R_z(\pi/2) \cdot R_x(\pi + \theta_{n-1})) \cdot \underline{D}_{n-1} \tag{4.8}$$

$$\vec{r}_n = \vec{r}_{n-1} + [1, 0, 0] \cdot \underline{D}_{n-1} \tag{4.9}$$

In summary, if we know the start position of the initial wedge, which we could put wherever we want, and the degree of the junctions between every two adjacent wedges, we can calculate the location and orientation for every wedge in the Magic Snake. In addition, we also obtain a matrix expression for the general Magic Snake to explore.

4.3 Multi-scale Modeling of Magic Snake

With the ability to convert between 1D and 3D representations of a Magic Snake, we are now able to compute the geometry of it. This enables us to design general structures of Magic Snakes without limitations. Here, we propose two design strategies, fitting a Magic Snake to any curve in space, and expanding one Magic Snake conformation to another.

4.3.1 Line Skeleton Representation as a Mask

To fit a Magic Snake to a curve in space, a line skeleton representation of Magic Snakes and a "Spoon" design concept are described.

Line Skeleton Representation

Magic Snakes become much more complex objects when we do not limit the rotations to be multiples of 90 degrees. Knowing all the vertices' locations is useful but a simplified model of a Magic Snake is desired. Here, we describe a line skeleton representation (LSR) of a Magic Snake to show only the skeleton of it. We ignore collisions and the shape of the wedges, and use a line to sketch the overall structure of a Magic Snake. This representation is easier to study even when the rotations are arbitrary angles.

We notice that for a single wedge of Magic Snake, if we know the location of three points – two centers for the square face (S_{1c}, S_{2c}) and the center for the rectangle face (R_c) – the unique location and orientation of this wedge is determined. If we connect the two centers of the rectangle faces (R_c) for adjacent wedges, the line will always pass the center of one of the square faces (S_c) on both wedges. Thus, if we have a conformation of a 3D Magic Snake structure, and we connect each rectangle center on each adjacent wedge (ignoring the first and last wedge on a Magic Snake, for they do not have previous or next R_c to connect to) we can identify the overall structure of a Magic Snake and simplify the model, see Figure 4.5. Note that, similar to rotation sequences, there are n line segments in LSR for the n wedge closed loop Magic Snake, and $n-1$ line segments in LSR for the n wedge open Magic Snake.

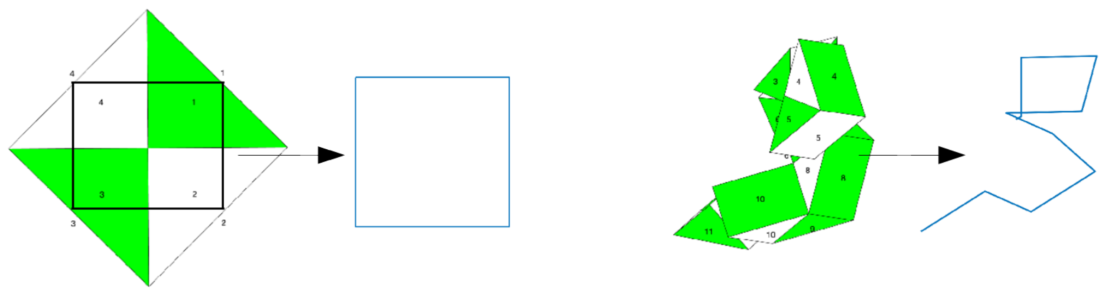


Figure 4.5: Left: Line skeleton representation (LSR) for a simple Magic Snake. Right: An arbitrary conformation of a Magic Snake and LSR of it.

Properties of line skeleton representation

We find that there is a useful property of LSR. Every two adjacent lines in this structure are perpendicular. If we consider the local wedges that generate these lines, the line bending only occurs at the rectangle center within each wedge, and if we connect "square center"(S_{1c}) – "rectangle center"(R_c) – "square center"(S_{2c}), it will always form a 90-degree angle for every wedge. Thus, any two adjacent lines in the line skeleton representation will always be perpendicular, even if the Magic Snake is not restricted to multiple 90 degree rotations. In addition, each line segment in LSR has equal lengths.

As a result, if we observe a piecewise curve that contains n equal length segments, and each adjacent segment is perpendicular, then we can treat it as a LSR of a Magic Snake and convert it back to a 3D representation of a Magic Snake. Without considering the collision that a Magic Snake might encounter during the arbitrary angle rotations, we can systematically construct any structure by simply designing these LSRs. To ensure a curve is piecewisely divided into n equal length segments with sufficient accuracy is straightforward, but restricting adjacent segments to be perpendicular is not.

“Spoon” design

To solve the problem of obtaining perpendicular line segments, we construct a “Spoon” structure as a basic unit, shown in Figure 4.6. The “Spoon” starts with a handle e_1 and ends with a tip e_4. Each adjacent edge in a “Spoon” is perpendicular, and all the edges in a “Spoon” are in the same plane. If given any two equal length connected lines (L_1, L_2), shown in Figure 4.6, we can always locate the “Spoon” on it and make the adjacent edges on the united “Spoon” structure perpendicular with each other. That is, we first put a “Spoon” along the first line (L_1 in Figure 4.6), so that the handle in the “Spoon” overlaps with L_1. Then we calculate the cross product of the given two lines and set the tip of the second “Spoon” overlapping with it; then this tip from the second “Spoon” must be perpendicular with the handle of the first “Spoon” for it is perpendicular with L_1. If the two given lines are in the same line and the cross product is invalid, we can put the second “Spoon” wherever we want. We choose the two “Spoons” in the same plane and in the same orientation for computation. As a result, if we have n equal length line segments in a chain, we can fit the “Spoon” on them one by one. This forms a line skeleton representation of the Magic Snake with $4 * n$ line segments, and then we can map this representation back to a Magic Snake with $4 * n$ wedges, as shown in Figure 4.7. To clarify, this strategy does not guarantee that collision will not occur in a 3D structure of a Magic Snake.

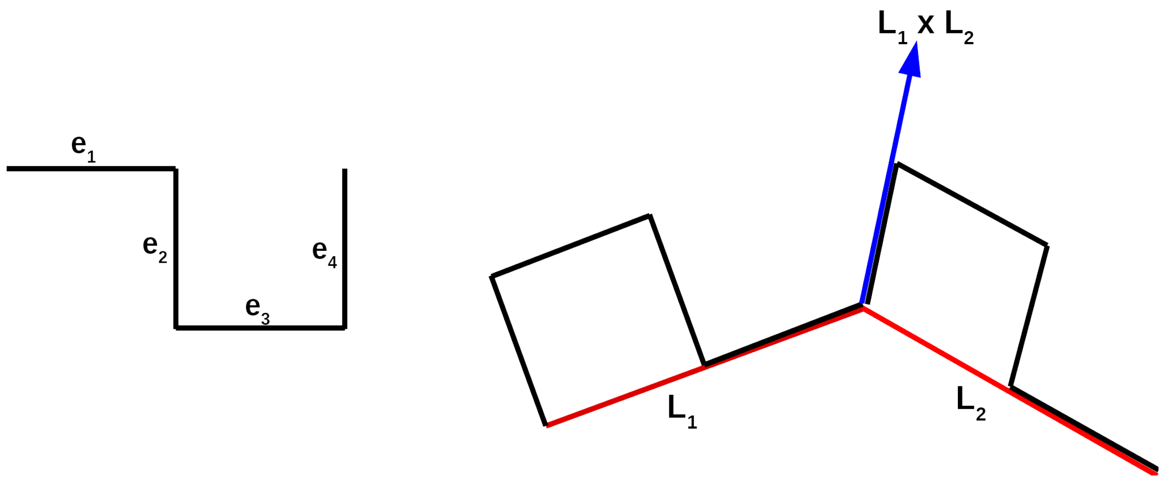


Figure 4.6: Left: "Spoon" desgin for construction of LSR. Right: The approach to fit any two random oriented equal length connected line (Red, L_1 and L_2).

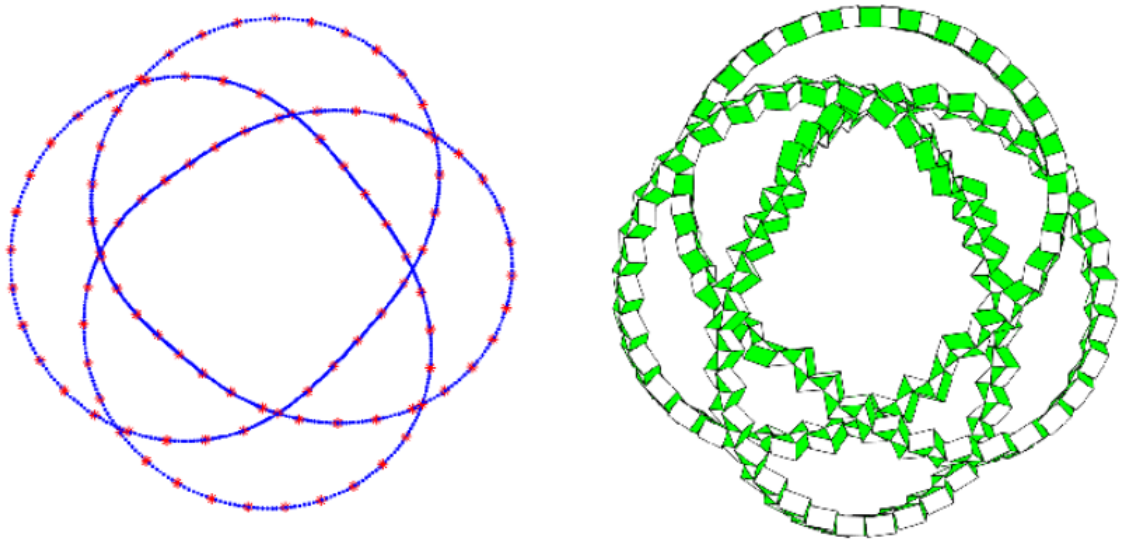


Figure 4.7: Left: We divide a structure of space curve into equal length line segments. Right: Fitting the Magic Snake to this space curve.

4.3.2 Embedded Wedge as a Mask

Now that we are able to design a Magic Snake based on any curve in space, then the question arises whether there is a way to convert or expand an existing Magic Snake structure into some other structure.

Our approach is that if we look "inside" a wedge, as shown on the left side of Figure 4.8, that is, dividing each edge into a trisection, then incise it along all the sections, then the one shown in red can be treated as a short sequence with 1D representation [0,0] embedded inside this wedge. Moreover, the entry square face and the exit square face of this short sequence is overlapping the square faces of the wedge, and sharing the same center points of the square faces. Then, for a Magic Snake consisting of n such wedges, imagine that for each wedge, we do the same thing and replace each wedge by "smaller wedges" inside it. Then the Magic Snake will convert into a thinner Magic Snake with more junctions. In addition, the new structure is self-similar to the original one, e.g. Figure 4.8 right. The conversion is as simple as changing the 1D rotation sequence:

$$\begin{bmatrix} R_0 & R_1 & \dots & R_n \end{bmatrix} \Longrightarrow \begin{bmatrix} R_0 & 0 & 0 & R_1 & 0 & 0 & \dots & R_n & 0 & 0 \end{bmatrix}$$

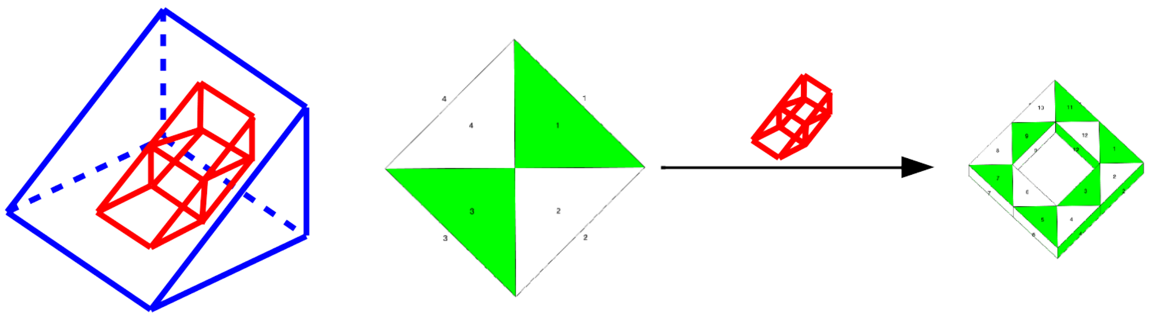


Figure 4.8: Left: "Embedded" structure inside a wedge. Right: Using [0,0] as the embedded sequence to expand a Magic Snake.

Notice that for now, we are just using the property of the "embedded structure" that shares the square center and face overlapping. That means if we are using this strategy, it is not limited to the simple [0,0] sequence. For example, the sequence

[1,1,2,1,0,3,2,3,3,0] in Figure 4.9 has the same property except it is longer and more complex, but the expansion should remain the same. As shown in Figure 4.9, the resulting structure is again self-similar. In addition, Magic Snake expansion could also be done by n iterations.

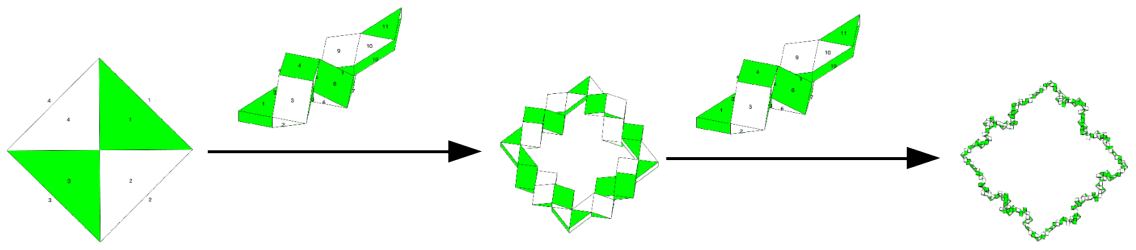


Figure 4.9: Using [1,1,2,1,0,3,2,3,3,0] as the sequence to expand Magic Snake once and twice.

4.4 Collision detection

4.4.1 Methods

Minkowski difference

Collision detection is also an essential task in 3D modeling. For the simplified case of modeling a space curve, if we represent each point on a space curve as a sphere, it is easy to identify whether two spheres are colliding. That is, for the two spheres i and j, calculate the distance of the two centers of the spheres, d_{ij}, and compare them with the sum of the radius of two spheres, $r_i + r_j$. If d_{ij} is larger, the two spheres are not colliding, and if not, the two spheres are colliding.

However, for an arbitrary 3D object, it is not trivial to find out whether two objects are colliding. This section seeks approaches of collision detection for arbitrary objects. Here, a method named Minkowski sum and difference [23] is described, Figure 4.10. The Minkowski sum and difference work on convex objects. For two convex objects A and B, suppose the vertices on A and B are in set AV and BV, then the formula of the Minkowski sum is

$$A \oplus B = \{a + b | a \in AV, b \in BV\},$$

and the Minkowski difference is

$$A \ominus B = \{a - b | a \in AV, b \in BV\}$$

Some important properties of the Minkowski difference used for collision detection are:

- The convex hull of the Minkowski sum/difference of A and B is the Minkowski sum/difference of the convex hull of A and B.
- If the origin point is outside the convex hull formed by the Minkowski difference of A and B, then A and B are not colliding; otherwise, they are colliding.
- The Minkowski sum/difference works for n dimension convex objects, $n = 1, 2, 3, \ldots$

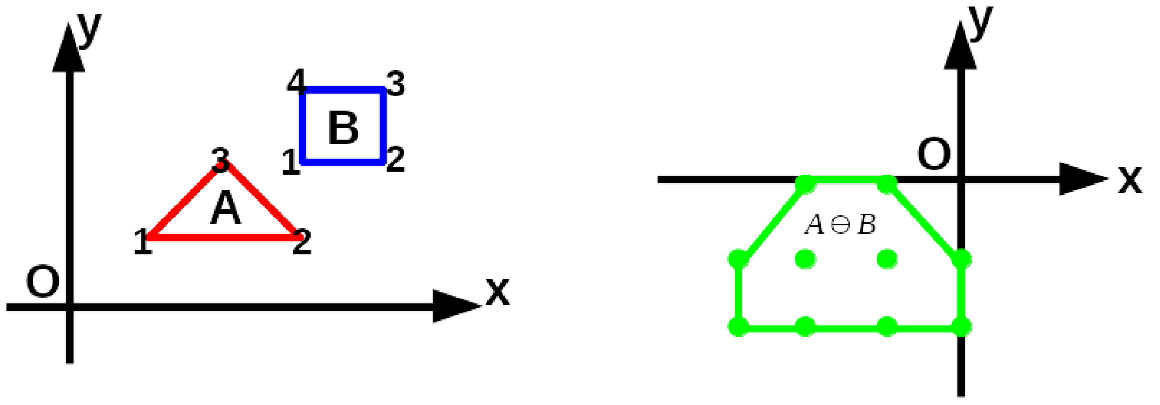
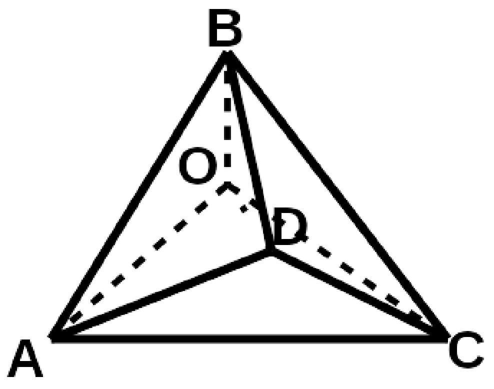


Figure 4.10: Minkowski difference: Left, two convex shape A and B, with coordinates $AV = \{a_1(1,1), a_2(3,1), a_3(2,2)\}$, and $BV = \{b_1(3,2), b_2(4,2), b_3(4,3), b_4(3,3)\}$. Right, the Minkowski difference convex hull (Green line) formed by subtraction of each $a_i \in AV$ and $b_i \in BV$. In this case, A and B are not colliding, so the origin point is outside the convex hull of $A \ominus B$.

The reason the Minkowski difference works is because when we are determining whether two objects are colliding, we are determining whether these two objects have points in common. Assume p is the point in common for the two objects A and B, then p - $p = 0$ (the origin point) must be in the convex hull of the Minkowski difference of A and B. Otherwise, if there are no points in common, the origin point will not be in the Minkowski difference of A and B. With these properties, we can then check whether two convex objects are colliding by calculating the Minkowski difference of them, and check whether the origin point is inside the Minkowski difference convex hull. A straightforward solution would be calculating the Minkowski difference exactly according to the formula, and for two convex objects with n and m vertices, the Minkowski difference will contain $n * m$ points. Whether a point in this Minkowski difference is inside or on the convex hull is not determined yet, if any tetrahedron formed by 4 points from these $n * m$ points of the Minkowski difference contains the origin point, the Minkowski difference contains the origin point.

In order to check whether a point is inside a tetrahedron, we can check the volume. Let A, B, C, D be the vertices of the tetrahedron, and O be the origin point. If the sum of $V_{OABC}, V_{OABD}, V_{OACD}, V_{OBCD}$ is equal to V_{ABCD}, the origin point is inside or on the surface of the tetrahedron. Otherwise, if $V_{OABC}+V_{OABD}+V_{OACD}+V_{OBCD} > V_{ABCD}$, the origin point is outside the tetrahedron, Figure 4.11. Then we have the complete solution of checking whether two convex objects are colliding by this method.



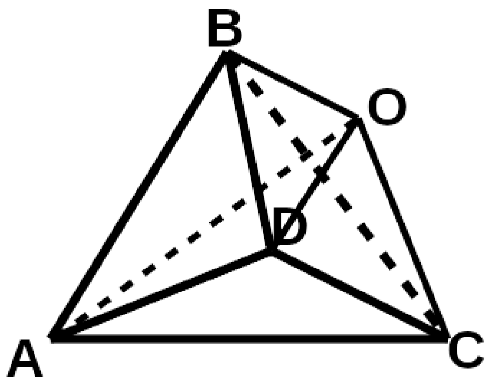


Figure 4.11: The idea to prove the property of checking a point is inside or outside a tetrahedron is that when the point is inside, the tetrahedron can be treated as the sum of several sub-tetrahedrons divided by connecting this point with each vertex on this tetrahedron; when the point is outside, it can be treated to calculate the volume of the overall object formed by the point and all four vertices on the tetrahedron, which will always contain the tetrahedron itself, so the overall volume is the sum of the volume of tetrahedron and some extra non-zero part for any possible cases. e.g. Left: O is inside the tetrahedron ABCD, $V_{OABC} + V_{OABD} + V_{OACD} + V_{OBCD} = V_{ABCD}$, as tetrahedron ABCD is formed exactly by tetrahedron OABC, tetrahedron OABD, tetrahedron OACD, and tetrahedron OBCD. Right: O is outside the tetrahedron ABCD, $V_{OABC} + V_{OABD} + V_{OACD} + V_{OBCD} > V_{ABCD}$, as $V_{OABC} + V_{OABD} + V_{OACD} + V_{OBCD} = V_{ABCD} + V_{OBCD}$ in this case, and $V_{OBCD} > 0$.

For convex objects with a small number of vertices, the computation cost is limited. However, for two convex objects with m and n vertices, the time complexity is O $((m * n)^4)$. It is almost impossible to apply this method in real time. Hence, an optimization of this method is desired.

Gilbert-Johnson-Keerthi algorithm

A method named Gilbert-Johnson-Keerthi (GJK) algorithm [24] is designed for this situation. GJK uses the properties of the Minkowski difference, but it does not need to calculate all the points in a Minkowski difference, Figure 4.12. The strategy is, for a given direction, if a point falls on the convex hull of the Minkowski difference of convex shape A and B, it should be using the furthest vertex in this direction of

A and the furthest vertex in the opposite direction of B. Then by starting with a random direction, we can calculate a line $\bar{ab}$ formed by two vertices on the convex hull of the Minkowski difference. Then by using this line, we can find in which direction the origin point is to this line $\bar{ab}$ and find another point c on the convex hull based on this direction. Now we have this triangle Δabc, and it is not hard to find out the direction of the origin point as to this triangle and find point d in this direction. By finding the direction and check the origin point as to tetrahedron abcd iteratively, we can determine whether the origin point is inside the Minkowski difference without exhausting all the tetrahedrons, or even calculating the Minkowski difference itself.

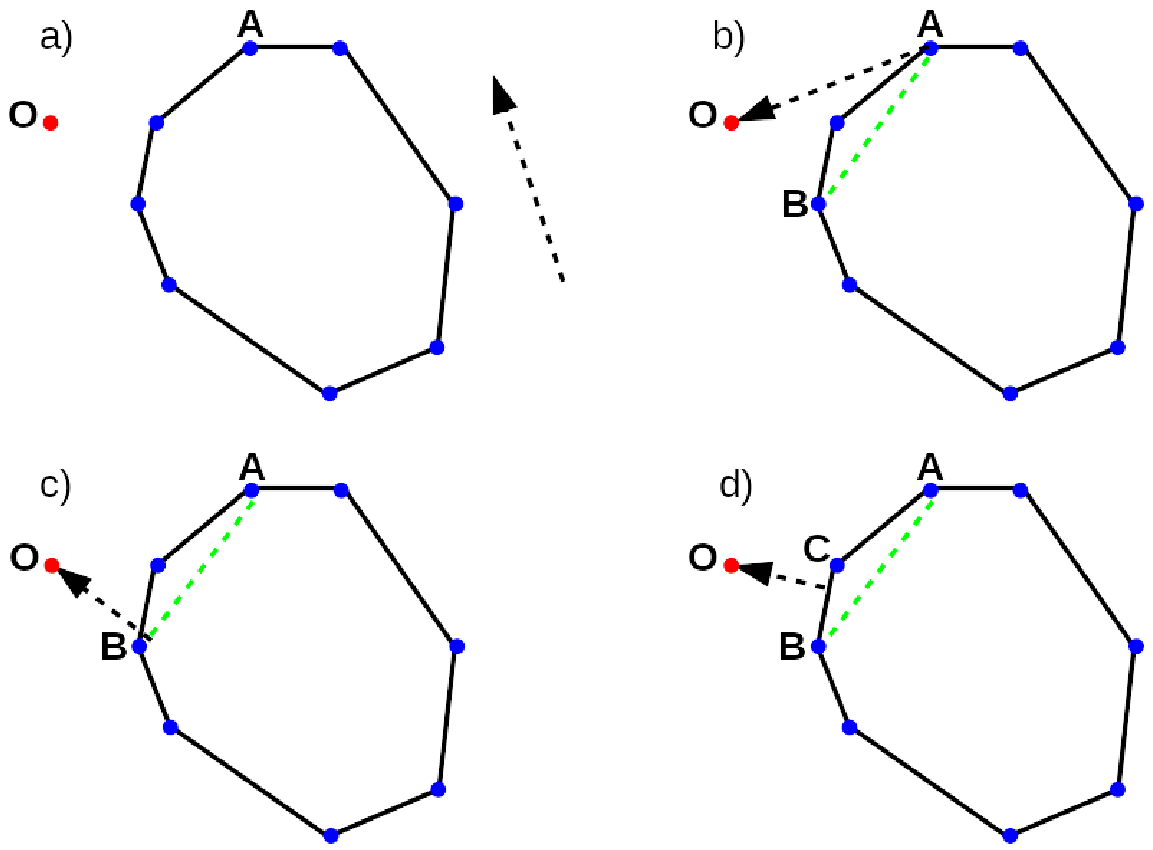


Figure 4.12: Let the convex shape in this figure is a Minkowski difference. a). Given a random direction and by finding the furthest vertex on this Minkowski difference by this direction, we identified vertex A. Note that we can directly determine this vertex by just using the convex shape A and B that generate this Minkowski difference as mentioned in the context. b). Let direction -A be the new direction and identify vertex B. c). Let the new direction perpendicular to AB and point to O, and identify vertex C. d). In the same way, we find a new direction perpendicular to BC and point to O, but there is no vertex that exists in this direction any more, so we know that there is no collision between convex objects A and B.

Now, we have a solution for detecting the collision for all the convex objects, even if a convex object has infinite vertices, i.e. $m, n \to \infty$. As to the concave objects, we can treat them as a combination of several convex objects, and check the collisions one by one.

In summary, in this chapter, we have provided a mathematical and logical foundation, and computational implementation that can build models from slender, branched to any objects multi-dimensional and multi-scale.

4.4.2 Results

For the study of Magic Snake, if the rotation is multiples of 90 degrees, there is a trivial way of detecting the collisions (naming “trivial solution”). That is, the collision only happens when two wedges are sharing the same cube, and the orientation of the two wedges are not “opposite” (the two wedges are not forming a cube). We have exhausted all the conformations that when the wedges of a Magic Snake is less than 14, and without collision used this way. By doing this, we just exhausted all the combination of [0,1,2,3], and check whether the Magic Snake colliding or not. We have also used the GJK algorithm to check the collision and compared it with the result produced above. We have checked all Magic Snakes with less than 10 wedges and the results are exactly the same, as shown in Table 4.1.

Table 4.1: We have exhausted all the possibilities using trivial solution and GJK algorithm to check collisions and record the number of Magic Snake that are not colliding under 10 wedges.

Number of wedges	Trivial solution	GJK solution
2	4	4
3	16	16
4	64	64
5	241	241
6	920	920
7	3384	3384
8	12585	12585
9	46471	46471

We then compared the timing between Volume check and GJK algorithm based on the results in the table above. We stored all the possible shapes of the Magic Snake into a hdf5 file, and in checking the collisions one by one, record all the time utilized as shown in Table 4.2. It shows that with the number of wedges and shapes increasing, GJK algorithm remains fast and much improved the performance than the volume check.

Table 4.2: Checking collision using volume check and GJK algorithm using the data from Table 4.1. We have recorded the time for collision detection up to 7 wedges. The PC and CPU specification is shown in Table 4.3 and Table 4.4

Number of wedges	Number of shapes	Volume (s)	GJK (s)
2	4	0.08	0.15
3	16	2.88	0.17
4	64	27.09	0.93
5	241	173.95	4.33
6	920	1014.90	9.50
7	3384	4773.30	35.30

Note that for Table 4.1, we only compared the results of the trivial solution with the results generated by the GJK algorithm, this is because we want to verify the accuracy of the GJK algorithm. By doing this, we have the correct answers provided by the trivial solution but limited to multiple of 90 degrees rotations. We can verify the GJK algorithm by limiting the GJK algorithm to check only the multiple of 90 degrees rotations and check the results. Table 4.1 does not contain the results from the Volume check algorithm because the Volume check algorithm will take too much time.

At the same time, Table 4.2 is the timing comparison between the Volume check and the GJK algorithms, it excludes the trivial solution because these two algorithms (GJK and Volume check) have the same application(not limited by multiple of 90 degrees rotations). Although we still time the number of structures with multiples of 90 degrees rotations, it is desired to compare these two algorithms with the same application, instead of comparing them with an algorithm with a different application.

Table 4.3: The computer specification for the timing in this dissertation.

CPU	Intel Xeon E5-2620 v4
GPU	nVidia GP104
Chipset	Intel C610/X99 series
Main Memory	64GB
OS	openSUSE Leap 15.0

Table 4.4: CPU specification for the timing in this dissertation.

Number of Cores	8
Number of Thread	16
Processor base frequency	2.10 GHz
Max Turbo frequency	3.00 GHz

4.5 Conclusions

Our research on Magic Snake aims to design the conformation of Magic Snakes computationally. This chapter first describes modeling strategies that provide both 1D and 3D descriptions of a Magic Snake. The 1D description can be used to share exact information of a conformation of a Magic Snake with others, and 3D modeling provides a general formula to convert any 1D sequence to the 3D structure. This chapter then describes two design strategies. One fits a Magic Snake with $4 * n$ wedges to any discrete curve with n equal length segments. This enables us to design a Magic Snake by designing any curve in space, but the limitation is that it does not provide collision detection. Another design strategy is to expand the n wedge Magic Snake into a $k^m * n$ wedge Magic Snake, where k is the number of wedges of the embedded Magic Snake, and m is the number of iterations. This enables us to design complex structures of a Magic Snake starting with simple ones. Of course, design strategies for a Magic Snake are various, and we expect more design strategies for the Magic Snake to be developed.

According to Erno Rubik, "The snake is not a problem to be solved; it offers infinite possibilities of combination. It is a tool to test out ideas of shape in space. Speaking theoretically, the number of the snake's combinations is limited. But speaking practically, that number is limitless, and a lifetime is not sufficient to realize all of its possibilities"[15]. For integer multiples of 90 degree rotations, a Magic Snake has at most 4^{n-1} possibilities of the combination with $n-1$ rotations among n wedges. As we are extending the rotations to any degrees, we are really dealing with these infinite possibilities. However, there are limited efforts of designing these combinations systematically. Even without these efforts, the structure of Magic Snakes already enlightens researchers in different areas (e.g. protein folding). With developing and designing Magic Snakes computationally, the outputs will not only be limited to the structures of Magic Snakes, but also provide better understanding of general slender body or even any 3D objects.

CHAPTER 5

MULTI-DIMENSIONAL AND MULTI-SCALE MODELING OF NANOCARS

5.1 Nanocar Introduction

With the improvement of computational performance, the simulations, such as Molecular Dynamics, are maturing. However, there is no systematic workflow to perform simulations, so researchers generate their own scripts to study interest aspects. In most cases, an initial conformation must be obtained. At the same time, chem-informatics provides a huge amount of data that can be utilized in initializing models for simulations and validating results from simulations. Here, we demonstrate a Nanocar Racer Web application as proof of concept to show how to support integrating high-level simulations with chem-informatics and achieve multi-dimensional modeling by data unification from 1D to 4D.

A nanocar is a molecule designed in 2005 at Rice University by a group headed by Professor James Tour [25]. It is designed to have functional groups that roll like wheels on a car, Figure 5.1. Nanocar Races are held as international scientific competitions with the aim of testing the performance of molecular machines and the scientific instruments used to control them. The races take place on a 100 nanometer gold surface for the first time held in Toulouse in April 2017 [26]. Nanocar races are a gamification of technoscience, but they are also a new way to carry out experiments [27].

From an experimental point of view, the design and testing of a Nanocar takes a whole series of stategies [28]. From a computational point of view, Nanocar can be designed and tested with all atom molecular dynamics simulation in real time. This process only takes computational power. The results can be validated by chem-informatics to inspire nanocar design and molecule mechanisms in general.

In this chapter, a general framework that unites chem-informatics and physical modeling is described. The framework utilizes off-the-shelf (OTS) software tools to get all the needed data and computation. A Model-View-Controller (MVC) design principles is used to unite the items. We demonstrate a "Nanocar Racing Webpage" that enables one to build nanocars, search chemical structures, simulate the chemical structure diffusing on a gold surface, and sharing simulation results in a "Nanocar Library". Our goal is not to provide a robust Nanocar Racing web application as a tool for general usage, but to demonstrate a framework that links pre-determined simulations to chem-informatics, and sharing or easily maintaining the user's Nanocar design data.

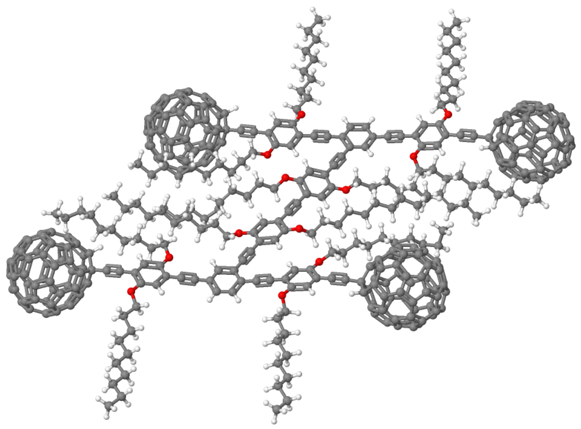


Figure 5.1: The Original Nanocar.

5.2 Multi-dimensional Modeling of Nanocars

For the multi-dimensional modeling of a Nanocar, we will describe the software needed to make a Nanocar Racing application, which is the off-the-shelf (OTS) idea of seeking applications that will provide desired inputs. The simulation strategy is described as a black box that computes diffusion as the race metric.

5.2.1 Off-The-Shelf

To build a "Nanocar Racing" application, we need a molecule representing a Nanocar, a gold surface for the track, simulation tools to run the simulation, and visualization tools to view the results. Based on these concerns, we have found the following applications.

Data Visualization from 1D to 4D

As shown in Table 5.1, from the Nanocar point of view, a Nanocar can be described as a 1D SMILES string, a 2D chemical structure, a 3D physical model, or 4D molecular dynamics. To visualize these descriptions, we select the OTS items as shown below:

Table 5.1: Nanocar Models from 1D to 4D.

Informatics	1D: XML	SMILES
	2D: JSME	Chemical Structure
Physical Modeling	3D: JSmol	Physical Structure
	4D: JSmol	Molecular Dynamics

1D: A SMILES string is just text. As in our web application, it is a string embedded in XML format so that it can be displayed on a webpage by HTML.

2D: JME is a Java-based molecule editor, and JSME [29] is the JavaScript version of JME. We utilize JSME to provide the 2D chemical structure view to show how a molecule is structured. It also provides the user the ability to draw molecules and return informatics data by Molfile, SMILES, or JME String. The informatics will provide enough information to convert these strings into a physical structure that contains all the coordinates of each atom in a molecule.

3D and 4D: JSmol [30] is the JavaScript version of Jmol, which is a software for molecular modeling chemical structures in 3D. With the informatics provided by JSME, the 3D structure of a molecule can be visualized. JSmol also provides the function of loading simulation trajectory data, which is a 4D (3D + time) visualization.

Data Conversions from 1D to 4D

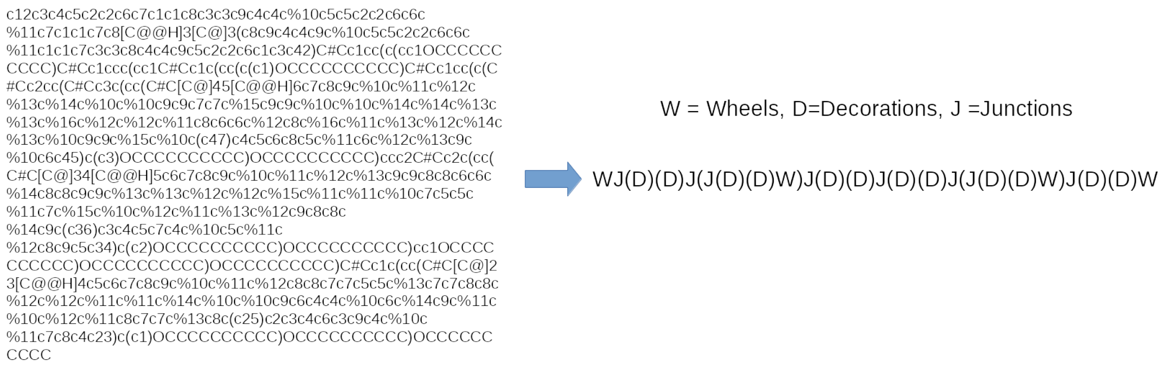
1D, 2D, 3D: Open Babel is an expert chemical system mainly used to interconvert chemical file formats. In the Nanocar case, Open Babel provides the conversions between a SMILE string (1D), a mol (2D) and a PDB (3D) file format. PDB file format is widely used by running simulations.

4D: After obtaining 3D structure by Open Babel, we selected a collection of simulation tools for converting 3D into 4D. Several packages of AMBERTools [31] are used in "Nanocar Racing" application. "antechamber" can be used to automatically apply atom types, bonds, and charges for most organic molecules in a database, where the SQM calculates to determine the charges, while "parmchk2" can be used to check parameters, and "tleap" can be used to build structures. VMD [32] is a Visual Molecular Dynamics tool, which can also be used to manipulate the locations of the atoms. We use VMD to put the "Nanocar" at the center of the gold surface. NAMD [33] is the simulation tool used to run the simulations that eventually outputs the 4D data in the Nanocar case. Note that we used these tools instead of something much simpler because we also want to show that we can make the computational "Black Box" arbitrarily complex.

5.3 Multi-scale Modeling of Nanocars

For the multi-scale modeling of the Nanocar, we have described the Masked SMILES in Chapter 2, which can be treated as a "coarse-grained" SMILES. Similar with the example of Nucleobase (ACGT) in Chapter 2, we can then define the Nanocar as a collection of Wheels (W), Decorations (D), and Junctions (J). As a result, the Nanocar shown in Figure 5.1 can then be written as:

"WJ(D)(D)J(J(D)(D)W)J(D)(D)J(D)(D)J(J(D)(D)W)J(D)(D)W", instead of the SMILES containing 953 characters, as shown in Figure 5.2.



```
c12c3c4c5c2c2c6c7c1c1c8c3c3c9c4c4c%10c5c5c2c2c6c6c
%11c7c1c1c7c8[C@@H]3[C@]3(c8c9c4c4c9c%10c5c5c2c2c6c6c
%11c1c1c7c3c3c8c4c4c9c5c2c2c6c1c3c42)C#Cc1cc(c(cc1OCCCCCCC
CCCC)C#Cc1ccc(cc1C#Cc1c(cc(c(c1)OCCCCCCCCCCC)C#Cc1cc(c(C
#Cc2cc(C#Cc3c(cc(C#C[C@]45[C@@H]6c7c8c9c%10c%11c%12c
%13c%14c%10c%10c9c9c7c7c%15c9c9c%10c%10c%14c%14c%13c
%13c%16c%12c%12c%11c8c6c6c%12c8c%16c%11c%13c%12c%14c
%13c%10c9c9c%15c%10c(c47)c4c5c6c8c5c%11c6c%12c%13c9c
%10c6c45)c(c3)OCCCCCCCCCCC)OCCCCCCCCCCC)ccc2C#Cc2c(cc(
C#C[C@]34[C@@H]5c6c7c8c9c%10c%11c%12c%13c9c9c8c8c6c6c
%14c8c8c9c9c%13c%13c%12c%12c%15c%11c%11c%10c7c5c5c
%11c7c%15c%10c%12c%11c%13c%12c9c8c8c
%14c9c(c36)c3c4c5c7c4c%10c5c%11c
%12c8c9c5c34)c(c2)OCCCCCCCCCCC)OCCCCCCCCCCC)cc1OCCCC
CCCCCC)OCCCCCCCCCCC)OCCCCCCCCCCC)C#Cc1c(cc(C#C[C@]2
3[C@@H]4c5c6c7c8c9c%10c%11c%12c8c8c7c7c5c5c%13c7c7c8c8c
%12c%12c%11c%11c%14c%10c%10c9c6c4c4c%10c6c%14c9c%11c
%10c%12c%11c8c7c7c%13c8c(c25)c2c3c4c6c3c9c4c%10c
%11c7c8c4c23)c(c1)OCCCCCCCCCCC)OCCCCCCCCCCC)OCCCCCCC
CCCC
```

W = Wheels, D=Decorations, J =Junctions

WJ(D)(D)J(J(D)(D)W)J(D)(D)J(D)(D)J(J(D)(D)W)J(D)(D)W

Figure 5.2: Coarse-grained SMILES. Left: the SMILES for the Nanocar as shown in Figure 5.1. Right: the coarse-grained "SMILES" for the same structure.

5.4 Computational Implementation

5.4.1 Data Sharing

All the necessary OTS items for multi-dimensional modeling of a Nanocar are obtained, which will process and generate simulation data for researchers to analyze. It is essential to have a solution for maintaining and sharing these data. TMB-Library [34] utilizes iBiOMES-Lite [35] to generate html pages to maintain and share data. Nanocar Racing also use iBiOMES-Lite to provide simple deployment of data sharing.

5.4.2 Our Simulation Workflow

With the help of the OTS items provided above, we design an initial molecule by JSmol, JSME or type in a SMILES string. JSmol and JSME provide a mol file. Both mol file and SMILES string can be converted into a pdb file using Open Babel. Then, we use antechamber to assign atom types, bonds and charges. We use parmchk2 to check molecular mechanics modeling parameters and tleap to build topology. In order to put the Nanocar at the center of the gold surface, we calculated the center of mass of the Nanocar and the center of mass of the gold surface, then moved the car center to the gold surface center and shifted a constant value in z direction, such that the Nanocar does not overlap with the gold surface. Finally, we run a pre-determined molecular dynamics simulation using NAMD.

We point out that this is a general simulation that is treated as a “Black-Box” from the Nanocar’s point of view. Here, we just provide set-ups for our simulation. Tools such as CHARMM-GUI [36] and QwikMD [37] provide simplier solutions. However our goal here is not to identify the most efficient solution but rather the demonstrate that the black box can be arbitrarily complex.

5.4.3 Model-View-Controller Design

The idea of integrating modeling and informatics is shown in Figure 5.3. We are utilizing the idea of Model-View-Controller (MVC). MVC is an implementation of the Separation of Concerns (SoC) design principle in which a computer program is separated into distinct features that overlap as little as possible [38]. An MVC design enables one to separate the informatics and physical data (models), the user interface (views), and the logic that connects them (controller). To achieve integrate modeling and informatics, we concentrate on developing the controller part as a “black-box” which connects models and views from different researchers.

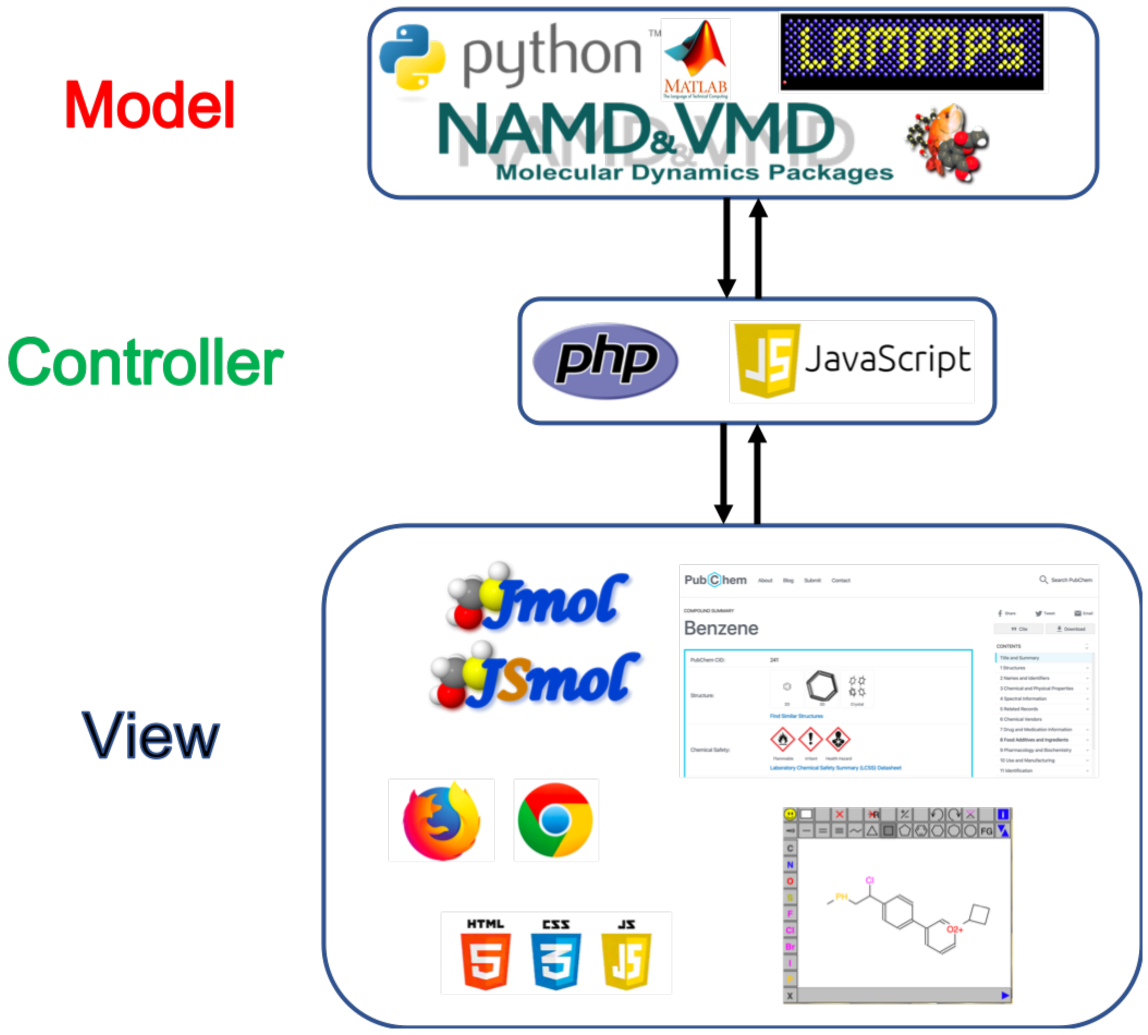


Figure 5.3: The software associated with the different components (Models, Views, Controller) are listed above.

Model

Different researchers utilize different physical and data models. The idea is to encapsulate these models in a “black-box” such that when specified inputs are provided, the model will automatically calculate and report relevant observable. In the case of the Nanocar Racing application, as shown in Table 5.1, the models could have conversions between informatics (chemical structure) and physics 3D structures where 3D structures are necessary for MD simulations, and Chemical Structure are necessary for informatics analysis (e.g. PubChem search).

View

View displays all the desired data in 1D to 4D, as shown in Figure 5.4. Besides the model that converts between these representations, users can also search in a database to get the informatics and display them on a webpage, and type in a SMILES string as a 1D view in Nanocar's case, build a 2D chemical structure on JSME panel, load up a xyz/pdb file as 3D physical models, and load xyz frames as a 4D view. Nanocar Library also provides a view of the database and the analyses.

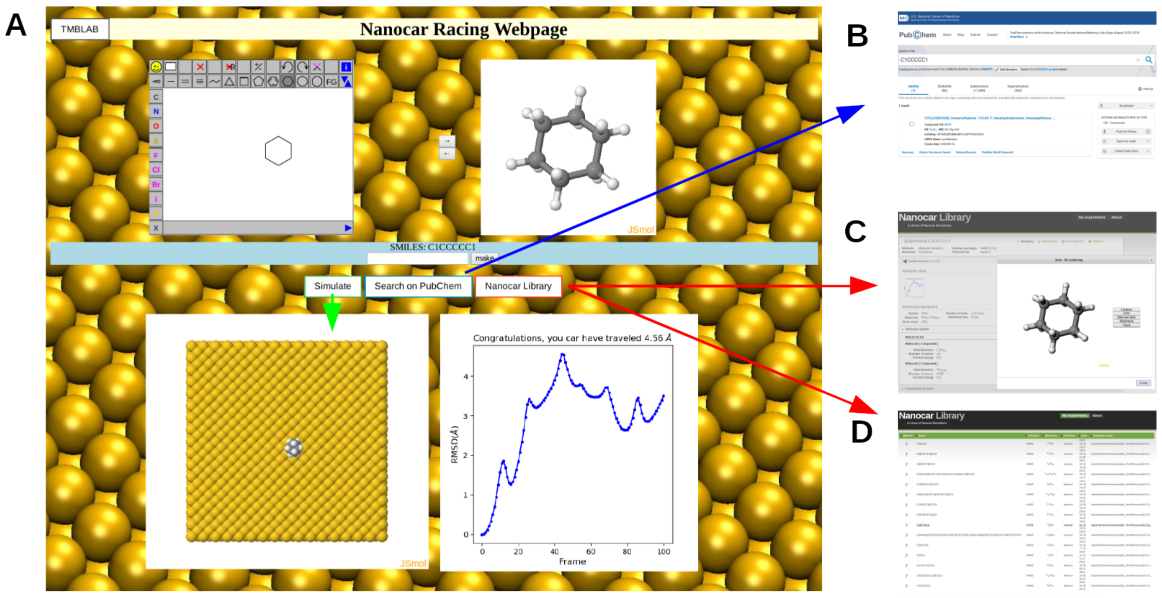


Figure 5.4: A) The Nanocar Racing Webpage, the upper components provides the representation of 2D chemical structure in JSME, 3D molecule in JSmol, and 1D SMILES in text. The lower components are generated after the click "Simulate" button, which provide a 4D view of the simulation in JSmol, and a sample analysis showing the RMSD of the Nanocar of each simulation frame to the first frame. The analysis indicates how far the Nanocar has traveled. B) "Search on PubChem" button in A) will open a PubChem page that redirect to the chemical structure created by the user. C) "Nanocar Library" button in A) will submit the simulation to the Nanocar Library. If no simulation has been done, it will redirect to the homepage of Nanocar Library in D) that can browse the simulations done by others.

Controller

The controller manages the exchange of data between model and view. In the Nanocar example, we can run simple simulation on our own server, and open the PubChem page to search informatics data as shown in Figure 5.4. They are all done by posting data through javascript and PHP, so that the model side knows which model to make and which parameters to use, and the view side knows which model to display. For example, for a typical usage of Nanocar Racing Web application, we first draw any chemical structure with embedded JSME. JSME is a javascript application itself, and by the embedded commands jme._getSmiles() or jme._getMol(), it will return a JSON (JavaScript Object Notation) string with SMILES or MOL format. Then, by the functionalities of jQuery [39], a javascript module, these JSON strings can be posted to PHP. PHP is server-side language which is able to write these JSON strings into files and run unix commands on the server side. As a result, for any researchers who have their own simulation scripts, PHP can be used to execute these scripts on the server-side.

In our case, PHP first writes the JSON string into a smi(SMILES) or a mol file, then executed a tcsh script that first called for Open babel to convert this file into a pdb format file, and do everything mentioned in Section 5.4.2. After this is done, PHP will notify the javascript side that posting is done, and on the javascript side, it can arrange html elements such as update the JSmol to load the xyz file just generated on the server side, thus accomplishing the whole procedure of communicating between the Model and the View. Of course, this is not the only way of doing these. Web application frameworks such as Django [40] and MERN can also achieve what has been mentioned in this section with proper setups.

5.5 Results

5.5.1 Nanocar Racing Webpage

Based on the methods provided above, we have developed a Nanocar Racing Webpage as a web application as shown in Figure 5.4. A user is able to type in 1D SMILES string in the text box, draw 2D chemical structures in JSME or upload 3D molecules in JSmol. Nanocar Racing Webpage will generate all three representations

(1D, 2D, 3D) when any one representation is provided. The pre-defined simulation is in “black box” and can be done by simply clicking the “Simulation” button. After the simulation, Nanocar Racing Web application will output the result of the simulation as an animation shown in JSmol, and a sample analysis described below indicates how far the Nanocar has traveled. The Nanocar Racing Webpage also enables web searching for the structure in PubChem, which associates with any chem-informatics and experimental data. Nanocar Library utilizes iBiOMES-lite [35] to manage simulation data, which was also utilizes in TMB Library [34].

We have recorded the time for our simulation workflow in Section 5.4.2, Table 5.2. It shows that for our workflow, building the structure and running the simulation does not take much time. Assigning atom types, bonds and charges is time consuming, and increases with more atoms and more complex structures. The charge assignment uses SQM to construct a semi-empirical quantum mechanics electron density calculation. Such QM calculations do not scale well and are cost prohibitive for large systems (typically less than 100 to 1000 atoms).

Table 5.2: Timing for our Nanocar simulation workflow mainly include three parts: use antechamber to assign atom types, bond and charges; use tleap to build structure; and use NAMD to run the simulation. The PC and CPU specification is shown in Table 4.3 and Table 4.4

Molecular Formula	Assign attributes (s)	Build structure (s)	Run simulation (s)
C_3H_6	0.090	0.026	5.409
C_4H_8	1.104	0.027	5.740
C_5H_{10}	3.360	0.026	5.688
C_6H_{12}	2.047	0.026	5.853
C_6H_6	4.586	0.021	5.868
C_6H_{10}	17.026	0.033	5.794
C_7H_{12}	17.914	0.027	5.842
$C_{38}H_{50}$	687.463	0.025	9.579
$C_{40}H_{70}$	991.691	0.031	8.667

5.5.2 Analysis

Nanocar Racing Web application provide a simple analysis accomplished by a VMD script. The script selects atoms in the first frame that are not gold, which is all the atoms in the molecule in the Nanocar, as the reference atoms. Then among all the frames generated by the simulation, we select atoms in the same way, then we can calculate RMSD (Root Mean Square Deviation) between these atoms with the reference atoms (without any fitting). We record these RMSD values as a function of time and plot it along the frames. As a result, we have this RMSD plot that reflects the distance and orientation a Nanocar travels vs time. As shown in Figure 5.4-A, the Nanocar is not travelling straight. Instead, it is tumbling around. This can be observed in this RMSD plot without watching the whole series of animations. The Nanocar is not going in a straight line because we do not apply any type of external force, so it moves only due to the attractive force between the molecule and the gold surface.

As a sample analysis, this demonstrates how easy it is to integrate analyses with our framework. Researchers with specific simulations and analyses could easily adopt this idea and take advantage of the framework.

5.6 Conclusions

We have developed a Nanocar Racing Web application as a generic web tool for integrating chem-informatics, physical modeling and simulation. It provides a computational platform to test Nanocar molecules racing on a gold surface. It is also a gamified application that introduces Nanocar to one that may have no knowledge about either nano-technique or computational simulation.

We emphsize that we are not planning to host the Nanocar Racing Webpage as a tool for general usage, but use it as a template for any multi-dimensional applications. The current Nanocar Racing web application is simplified and does not provide electrical impulses and electron transfer as in real Nanocar Racing. The OTS items for simulation in our workflow is too much for applying simple simulations, so it runs slow even on small molecules. However, this idea can be adopted to setup simulations properly, and the OTS items are replaceable. There is no common workflow for simulations in individual labs [41], but each individual is a "black box" to our means of usage that can be applied to our framework.

In this manner, the "Nanocar" can be replaced by any multi-dimensional modeling objects, and build applications computationally follows the MVC framework. It is not even required the knowledge to convert the desired object into different data representations as long as the OTS items exist. The Nanocar Library, which is also a replaceable OTS item, provides data accessibility. This enables researchers get advantage of the application with the knowledge of certain data representation. As a result, computational implementation proposed in this dissertation not only provides the "communication" between different data representations, but also provides the "communication" between researchers from different area of expertise.

CHAPTER 6

MULTI-DIMENSIONAL AND MULTI-SCALE MODELING OF GENOME DASHBOARDS

6.1 Genome Dashboards Introduction

Genomics is a sequence-based informatics science and a 3D structure-based material science. Here, we describe a framework for developing genome dashboards specifically designed to unify informatics data with structure and dynamics data. The framework describes not a single tool but a novel class of computational tools. The framework is based on the mathematical representation of geometrically exact rods and the generalization of DNA base pair step parameters. A Model-View-Controller software design approach is proposed as an efficient means of implementing a genome dashboard as a finite state machines as either a desktop or web based application. Two examples are demonstrated using our minimal genome dashboard called G-Dash. The data unification achieved with a genome dashboard supports the bi-directional exchange of data between informatics and structure. Thus, any experimentally or theoretically determined sequence based informatics track can inform DNA, nucleosome or chromatin modeling (e.g. nucleosome positions) and structure features extracted from a computational model or experimentally determined structure can be analyzed as informatics tracks in a genome browser (e.g. DNA base pair step parameters: Roll, Tilt, Twist). Here, the framework is applied to chromatin, but genome dashboards are broadly applicable. Genome dashboards are a novel means of investigating structure-function relationships for genomes that range from base pairs to entire chromosomes and for generating, validating, and testing mechanistic hypotheses.

Chromatin is the biomaterial that contains the genome in all higher organisms. There is no consensus on the structure of chromatin [42], but there is a wealth of informatics and structure data available. From an informatics point of view, efforts such as the 1000 Genomes [43], ENCODE [44] and the 4D Nucleome [45] projects provided sequence based reference data. Coupling the reference data with Next Generation Sequencing (NGS) and informatics analysis pipelines enables individual labs to conduct Genome Wide Association (GWA) studies that link chromatin reprogramming with disease and altered gene expression as described, for example, in [46, 47]. Hi-C [48, 49], Micro-C [50], and other chromosome conformation capture methods [51, 52] provide distance constraints as a measure of the large scale organization of chromatin structure. Super resolution microscopy [53, 54] provides optical visualization of 3D structure at nanometer scale resolution, while electron microscopy [55], NMR [56], and X-ray crystallography [57] provide ångström scale resolution of individual nucleosomes and nucleosome arrays. Strategies for modeling chromatin structure are rapidly maturing [58, 59, 60]. The desire to merge computational and experimental approaches is recognized [61, 62], but a significant challenge in chromatin structural biology is unifying these diverse data sets to advance our understanding of structure-function relationships and to validate genomic mechanisms of action.

Here, we categorize data according to the method typically used to display and analyse the data. Sequence based informatics is a 1D representation. Contact and distant constraints are 2D representations. X-ray, NMR and super-resolution microscopy are 3D representations. Molecular modeling and dynamics are 4D representations. There exists a growing collection of computational tools that convert 2D data to 3D structures of chromatin [63], and computational models can promote 3D structures to dynamics or sampling data (4D). However, there remain a few tools, other than ICM-Web [64], that directly link sequence (1D) with chromatin structure (3D) or dynamics (4D). Thus, researchers utilizing 1D sequence based methods are missing 3D and 4D structure and dynamics data including steric and geometric constraints in their analyses, and researchers utilizing 3D and 4D computational and experimental methods are missing the wealth of informatics data available in sequence based data sets.

A genome dashboard, like an automobile dashboard or airplane cockpit, integrates a console for managing data with controllers for navigating a physical world that appears in a window. A "genome dashboard" unifies informatics (1D), contact and two-angle representations (2D) [65], structure (3D), and dynamics (4D) data describing DNA, nucleosomes and chromatin. For the purpose of developing such genome dashboards, we have identified a framework that unifies 1D and 3D representations and a general method for implementing it that supports data visualization and manipulation. The framework is bi-directional, i.e. it can map 3D representations to 1D and 1D representations to 3D.

The framework is based on mathematical representations of geometrically exact rods [3], presented below as “The Model”, followed by “Design Considerations” for implementing it using Model-View-Controller software development principles. This approach enables a commodity-off-the-shelf (COTS) approach for assembling genome dashboards that is both extensible and portable. Two examples are then presented to demonstrate that G-Dash, a minimal web based implementation of a genome dashboard, can function in real time in both directions to convert informatics data into physical structures and physical structures into informatics data.

6.2 Multi-dimensional Modeling with Genome Dashboards

From the genome dashboard perspective, informatics (1D) is any data that maps to a DNA sequence. Generally speaking, NGS relies on aligning experimental data with chromosome coordinates and information theory for analysis. Physical structures include computational models and direct imagining by experiment. For the computational models, energy functions and physical laws are employed for analysis. The energy functions are typically grouped into external and internal energies. $U = U_{ext} + U_{int}$, where U_{ext} captures through space interactions and U_{int} captures local conformation and dynamical properties. Both types of energy functions require knowledge of material properties and geometry. Material properties such as van der Waals radii, dielectric properties, partial charge distributions, moments of inertia, mass, stiffness parameters, bond angles, etc. are all parameters associated with a specific physical model or force field. The model itself may employ atomic, coarse-grained or even continuum approximations. In all cases, the physical structure may be expressed in a laboratory (external) or material (internal) reference frame. The structure itself may be obtained from theoretical or experimental techniques.

Our strategy for unification (merging data from different sources) is based on the idea that DNA is the common thread in chromatin structural biology. Unification is achieved through laboratory (Cartesian coordinate) and material (internal coordinate) representations of DNA as an oriented space curve or just ribbon for simplicity, Figure 6.1. Since unification is based on geometric considerations, our strategy is independent of the parameters associated with a specific physical model. Associating an energy landscape with a physical structure requires one to choose a physical model, but an experimentally determined structure can be compared to informatics data without recourse to any such physical model applied. Our framework does not provide a model, but it also does not restrict the user's choice of the model. Various implementations of the genome dashboard concept may support one or many models or rely solely on experimental determined structure data.

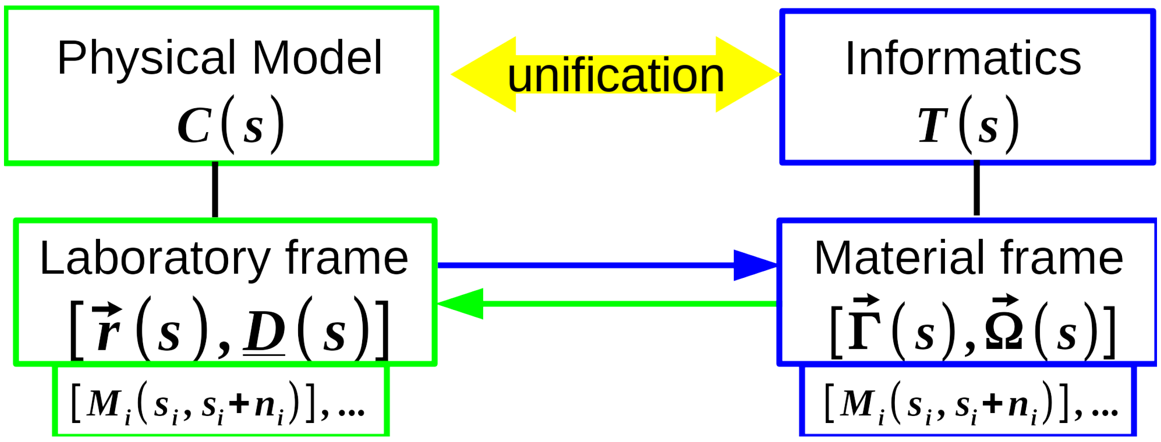

Physical Model
$C(s)$
unification
Informatics
$T(s)$
Laboratory frame
$[\vec{r}(s), \underline{D}(s)]$
Material frame
$[\vec{\Gamma}(s), \vec{\Omega}(s)]$
$[M_i(s_i, s_i+n_i)], \ldots$
$[M_i(s_i, s_i+n_i)], \ldots$

Figure 6.1: Unification is the process of merging data from different sources. Physical models and informatics data are unified by mathematical representations of an oriented space curve in laboratory $[\vec{r}(s), \mathbf{D}(s)]$ and material $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ reference frames. The conformation of a physical model $C(s)$ is associated with the laboratory frame, and informatics data $T(s)$ is associated with the material frame. Masks $M(s)$ alter the material properties of DNA. Exchanging data between laboratory and material frames unifies the physical model and informatics.

In a laboratory reference frame, a continuous ribbon has a centerline $\vec{r}(s)$ and unit length directors $\hat{d}_i$ embedded in the ribbon that capture the local orientation of the ribbon. The directors can be represented by a director frame matrix $\mathbf{D} = \{\hat{d}_1(s), \hat{d}_2(s), \hat{d}_3(s)\}$ [66]. This matrix also serves to transform a representation in the material (internal) frame to a representation in the laboratory (Cartesian coordinate) frame.

An equivalent description of the ribbon is based on the director frames themselves. This description is a material reference frame description that captures the translations and rotations connecting one director frame to the next, represented here by $[\vec{\Gamma}(s), \vec{\Omega}(s)]$. The two representations $[\vec{r}(s), \mathbf{D}(s)]$ and $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ are equivalent descriptions of the conformation of an oriented space curve, denoted simply as $C(s)$

for the Cartesian coordinate representation and $T(s)$ when expressed in the material frame and interpreted as informatics tracks.

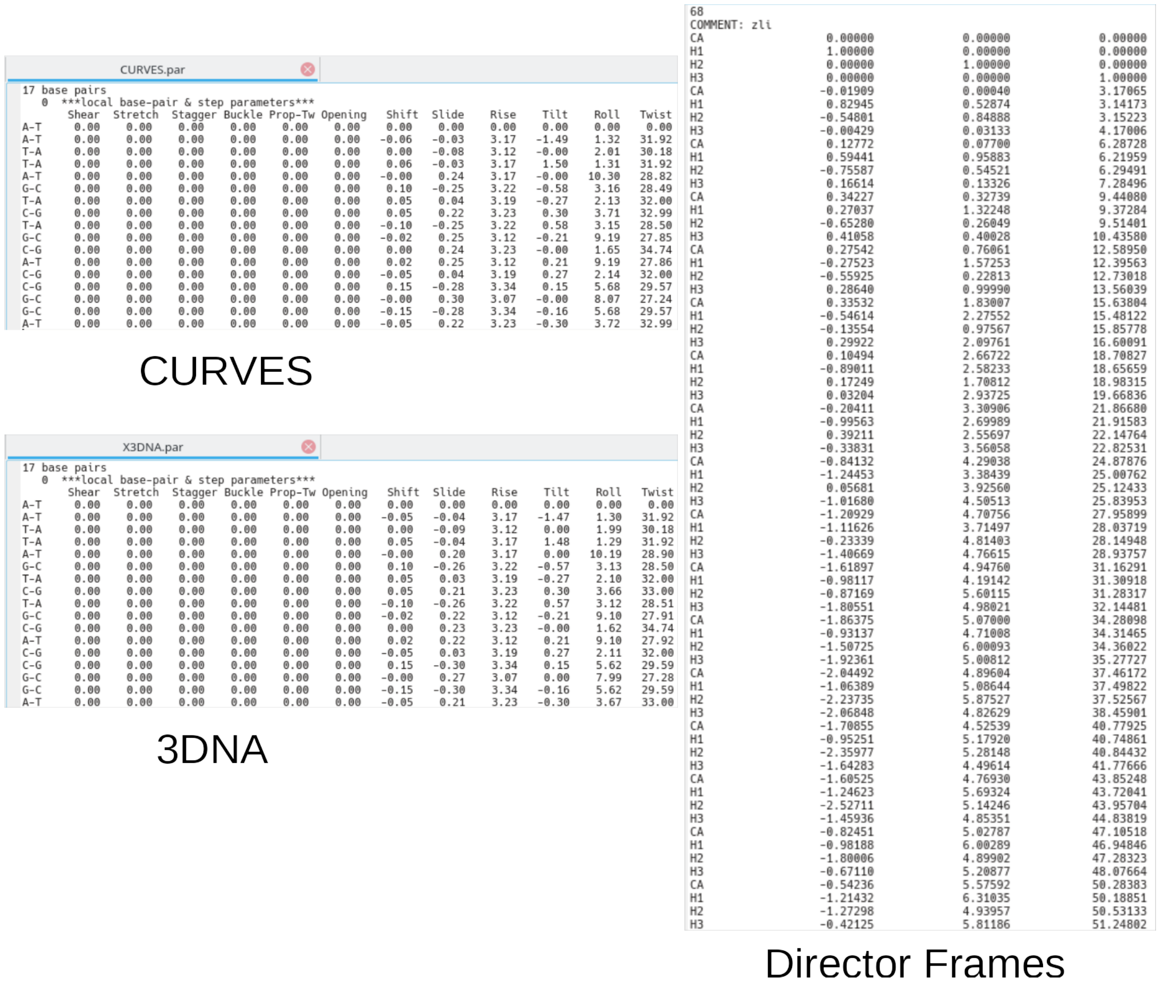
Converting between the $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ and $[\vec{r}(s), \mathbf{D}(s)]$ representations requires either a differentiation or an integration as expressed by the following equations:

$$\frac{d\vec{r}}{ds} \equiv \vec{t} = \mathbf{D}\vec{\Gamma} \qquad \frac{d\hat{d}_{\mathrm{i}}}{ds} = \mathbf{D}\vec{\Omega} \times \hat{d}_{\mathrm{i}} \tag{6.1}$$

$$\vec{r}(s_1, s_2) = \int_{s_1}^{s_2} \mathbf{D}\vec{\Gamma} ds \qquad \hat{d}_i(s_1, s_2) = \int_{s_1}^{s_2} \mathbf{D}\vec{\Omega} \times \hat{d}_i ds \tag{6.2}$$

Here, $\vec{t}(s)$ is recognized as the unnormalized tangent to the ribbon expressed in the laboratory frame. $\vec{\Gamma}(s)$ is the same vector expressed in the internal frame. $\mathbf{D}\vec{\Omega}(s)$ is recognized as the vector corresponding to the instantaneous axis of rotation of the director frames located along the ribbon at position s, as represented in the laboratory coordinate frame. Discrete approximations to Equation 6.1 and piecewise integration as expressed in Equation 6.2 can be employed to obtain a collection of discrete director frames. Different models may require different numerical algorithms to achieve the required discretization. Numerical methods suitable for DNA are discussed below. They are reading strand invariant. Together, the material (internal) and laboratory (Cartesian coordinate) representations provide a basis for unifying informatics ($[\vec{\Gamma}(s), \vec{\Omega}(s)]$) and structure ($[\vec{r}(s), \mathbf{D}(s)]$) data.

DNA conformation $C(s)$ is at best a base pair discrete approximation to a continuous oriented space curve [67, 68]. Base pair step parameters [4, 69] and associated algorithms provide established methods for describing double and single stranded DNA as a discrete oriented space curve at atomic resolution. A sequence specific di-nucleotide accurate model of dsDNA in the $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ (base pair step parameter) representation can be obtained from x-ray [70] or molecular dynamics [71, 72] studies. A $[\vec{r}(s), \mathbf{D}(s)]$ description is obtained by integrating $[\vec{\Gamma}(s), \vec{\Omega}(s)]$. There are two widely used tools for base pair step parameter analysis. 3DNA [8] uses a Euler Angle (E-A) based method and employs a "RollTilt" approximation [7]. Curves+ [10] uses a Euler-Rodrigues (E-R) based method [9], Figure 6.2. Both methods utilize a mid-step plane construction to ensure that the computed parameter values are not affected by the choice of reading strand. Mathematically, one must invert the sign of Tilt and Shift upon strand reversal to preserve the alignment of the director frames with the DNA major and minor groove [73].



```
17 base pairs
   0  ***local base-pair & step parameters***
        Shear  Stretch  Stagger Buckle Prop-Tw Opening   Shift  Slide     Rise    Tilt    Roll   Twist
A-T      0.00     0.00     0.00    0.00    0.00    0.00    0.00   0.00     0.00    0.00    0.00    0.00
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.06  -0.03     3.17   -1.49    1.32   31.92
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.00  -0.08     3.12   -0.00    2.01   30.18
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.06  -0.03     3.17    1.50    1.31   31.92
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.00   0.24     3.17   -0.00   10.30   28.82
G-C      0.00     0.00     0.00    0.00    0.00    0.00    0.10  -0.25     3.22   -0.58    3.16   28.49
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.05   0.04     3.19   -0.27    2.13   32.00
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.05   0.22     3.23    0.30    3.71   32.99
T-A      0.00     0.00     0.00    0.00    0.00    0.00   -0.10  -0.25     3.22    0.58    3.15   28.50
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.02   0.25     3.12   -0.21    9.19   27.85
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.00   0.24     3.23   -0.00    1.65   34.74
A-T      0.00     0.00     0.00    0.00    0.00    0.00    0.02   0.25     3.12    0.21    9.19   27.86
C-G      0.00     0.00     0.00    0.00    0.00    0.00   -0.05   0.04     3.19    0.27    2.14   32.00
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.15  -0.28     3.34    0.15    5.68   29.57
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.00   0.30     3.07   -0.00    8.07   27.24
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.15  -0.28     3.34   -0.16    5.68   29.57
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.05   0.22     3.23   -0.30    3.72   32.99
```

```
17 base pairs
   0  ***local base-pair & step parameters***
        Shear  Stretch  Stagger Buckle Prop-Tw Opening   Shift  Slide     Rise    Tilt    Roll   Twist
A-T      0.00     0.00     0.00    0.00    0.00    0.00    0.00   0.00     0.00    0.00    0.00    0.00
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.05  -0.04     3.17   -1.47    1.30   31.92
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.00  -0.09     3.12    0.00    1.99   30.18
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.05  -0.04     3.17    1.48    1.29   31.92
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.00   0.20     3.17    0.00   10.19   28.90
G-C      0.00     0.00     0.00    0.00    0.00    0.00    0.10  -0.26     3.22   -0.57    3.13   28.50
T-A      0.00     0.00     0.00    0.00    0.00    0.00    0.05   0.03     3.19   -0.27    2.10   32.00
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.05   0.21     3.23    0.30    3.66   33.00
T-A      0.00     0.00     0.00    0.00    0.00    0.00   -0.10  -0.26     3.22    0.57    3.12   28.51
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.02   0.22     3.12   -0.21    9.10   27.91
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.00   0.23     3.23   -0.00    1.62   34.74
A-T      0.00     0.00     0.00    0.00    0.00    0.00    0.02   0.22     3.12    0.21    9.10   27.92
C-G      0.00     0.00     0.00    0.00    0.00    0.00   -0.05   0.03     3.19    0.27    2.11   32.00
C-G      0.00     0.00     0.00    0.00    0.00    0.00    0.15  -0.30     3.34    0.15    5.62   29.59
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.00   0.27     3.07    0.00    7.99   27.28
G-C      0.00     0.00     0.00    0.00    0.00    0.00   -0.15  -0.30     3.34   -0.16    5.62   29.59
A-T      0.00     0.00     0.00    0.00    0.00    0.00   -0.05   0.21     3.23   -0.30    3.67   33.00
```

```
68
COMMENT: zli
CA                 0.00000            0.00000            0.00000
H1                 1.00000            0.00000            0.00000
H2                 0.00000            1.00000            0.00000
H3                 0.00000            0.00000            1.00000
CA                -0.01909            0.00040            3.17065
H1                 0.82945            0.52874            3.14173
H2                -0.54801            0.84888            3.15223
H3                -0.00429            0.03133            4.17006
CA                 0.12772            0.07700            6.28728
H1                 0.59441            0.95883            6.21959
H2                -0.75587            0.54521            6.29491
H3                 0.16614            0.13326            7.28496
CA                 0.34227            0.32739            9.44080
H1                 0.27037            1.32248            9.37284
H2                -0.65280            0.26049            9.51401
H3                 0.41058            0.40028           10.43580
CA                 0.27542            0.76061           12.58950
H1                -0.27523            1.57253           12.39563
H2                -0.55925            0.22813           12.73018
H3                 0.28640            0.99990           13.56039
CA                 0.33532            1.83007           15.63804
H1                -0.54614            2.27552           15.48122
H2                -0.13554            0.97567           15.85778
H3                 0.29922            2.09761           16.60091
CA                 0.10494            2.66722           18.70827
H1                -0.89011            2.58233           18.65659
H2                 0.17249            1.70812           18.98315
H3                 0.03204            2.93725           19.66836
CA                -0.20411            3.30906           21.86680
H1                -0.99563            2.69989           21.91583
H2                 0.39211            2.55697           22.14764
H3                -0.33831            3.56058           22.82531
CA                -0.84132            4.29038           24.87876
H1                -1.24453            3.38439           25.00762
H2                 0.05681            3.92560           25.12433
H3                -1.01680            4.50513           25.83953
CA                -1.20929            4.70756           27.95899
H1                -1.11626            3.71497           28.03719
H2                -0.23339            4.81403           28.14948
H3                -1.40669            4.76615           28.93757
CA                -1.61897            4.94760           31.16291
H1                -0.98117            4.19142           31.30918
H2                -0.87169            5.60115           31.28317
H3                -1.80551            4.98021           32.14481
CA                -1.86375            5.07000           34.28098
H1                -0.93137            4.71008           34.31465
H2                -1.50725            6.00093           34.36022
H3                -1.92361            5.00812           35.27727
CA                -2.04492            4.89604           37.46172
H1                -1.06389            5.08644           37.49822
H2                -2.23735            5.87527           37.52567
H3                -2.06848            4.82629           38.45901
CA                -1.70855            4.52539           40.77925
H1                -0.95251            5.17920           40.74861
H2                -2.35977            5.28148           40.84432
H3                -1.64283            4.49614           41.77666
CA                -1.60525            4.76930           43.85248
H1                -1.24623            5.69324           43.72041
H2                -2.52711            5.14246           43.95704
H3                -1.45936            4.85351           44.83819
CA                -0.82451            5.02787           47.10518
H1                -0.98188            6.00289           46.94846
H2                -1.80006            4.89902           47.28323
H3                -0.67110            5.20877           48.07664
CA                -0.54236            5.57592           50.28383
H1                -1.21432            6.31035           50.18851
H2                -1.27298            4.93957           50.53133
H3                -0.42125            5.81186           51.24802
```

Figure 6.2: Values of the base pair step parameters obtained from an Euler-Rodrigues (CURVES) and an Euler-Angle (3DNA) analysis of the director frames indicated on the right. The Director Frames are an oriented space curve or ribbon representation of the "AATTAGTCTGCACCGGA" model shown in Figure 6.4.

Base pair step parameter values obtained from the same DNA structure using the 3DNA and Curves+ methods are known to differ [11]. (Figure 6.4 shows the default base pair step parameters used in G-Dash as in CURVES+ or 3DNA values.) Differences may arise from at least three sources. The assignment of director frames to the base pairs may differ. However, the methods for assigning director frames are well-defined for ideal pairing [69], so these differences typically occur only for significant deviations from ideal geometries. The other source of differences is method dependent. These differences have not been well studied. Base pair step parameter values obtained from 3DNA and from Curves+ differ even when the director frames used for the calculations are identical, i.e. even when the first problem is eliminated. Thus, values obtained from one method should not in general be interchanged with the other. Finally, differences arise from implementation and usage, e.g. numerical precision during file read and write operations may differ. Nonetheless, these two methods work well for all-atom representations of the pairing and stacking of base pairs in double stranded DNA with the caveat that neither provides information about the DNA backbone. Recent efforts now support proper reconstruction of the DNA backbone [74].

6.3 Multi-scale Modeling with Genome Dashboards

6.3.1 DNA Masks

Chromatin, for our purposes, is a biomolecular structure composed of DNA and external agents, such as histones, that alter the material properties (conformation, dynamics, flexibility, energetics, chemical properties) of a contiguous length of DNA from s to $s+n$. We label any such external agent, including histones, as a Mask, $M(s, s+n)$. Any number of identical or unique Masks ($M_i(s_i, s_i+n_i)$) may be associated with a sequence of DNA (s_i to s_i+n_i). With this approach, chromatin folding is an informatics problem of managing an inventory of Masks. A geometric description of structure requires only knowledge of how Masks alter conformation. More generally, a Mask may alter the parameters associated with the energy functions for a specific physical model.

There are two strategies for activating structural changes associated with a Mask. The first is achieved in the material reference frame with the conformation of the masked DNA denoted by a list of internal coordinates $M(s, s{+}n) = [[\vec{\Gamma}_M(s), \vec{\Omega}_M(s)], ..., [\vec{\Gamma}_M(s+n), \vec{\Omega}_M(s+n)]]$ that spans n base pairs. As the name suggests, the Mask replaces the $[[\vec{\Gamma}(s), \vec{\Omega}(s)], ..., [\vec{\Gamma}(s+n), \vec{\Omega}(s+n)]]$ values associated with DNA. We can calculate $C(s)$ from Equations 6.2, as discussed above. The second approach is achieved in the laboratory reference frame with the conformation of DNA described as a rigid entity with $M(s, s+n) = [[\vec{r}_M(s), \mathbf{D}_M(s)], ..., [\vec{r}_M(s+n), \mathbf{D}_M(s+n)]]$. The Mask consists of Cartesian coordinates and director frames which can be converted to $T(s)$ using Equations 6.1 as discussed above. The Cartesian coordinate representation of $M(s, s+n)$ requires only a single translation and rotation to position each rigidly masked element in the laboratory reference frame.

In terms of Masks, a nucleosome is a DNA superhelix *and* histones. Depending on the modeling strategy, the histones can be represented independently of the DNA as a single entity (sphere, cylinder, ellipsoid), a collection of beads, or an all-atom model. Alternatively, the DNA *and* histones can be included in the nucleosome Mask as a single entity, Figure 6.3. Docking individual histones or the complete histone octamer to a superhelix or placing entire nucleosomes between linkers can be achieved with the same methods and tools used for describing the relative rotations and translations of base pairs. However, the “RollTilt” small angle approximation is no longer valid.

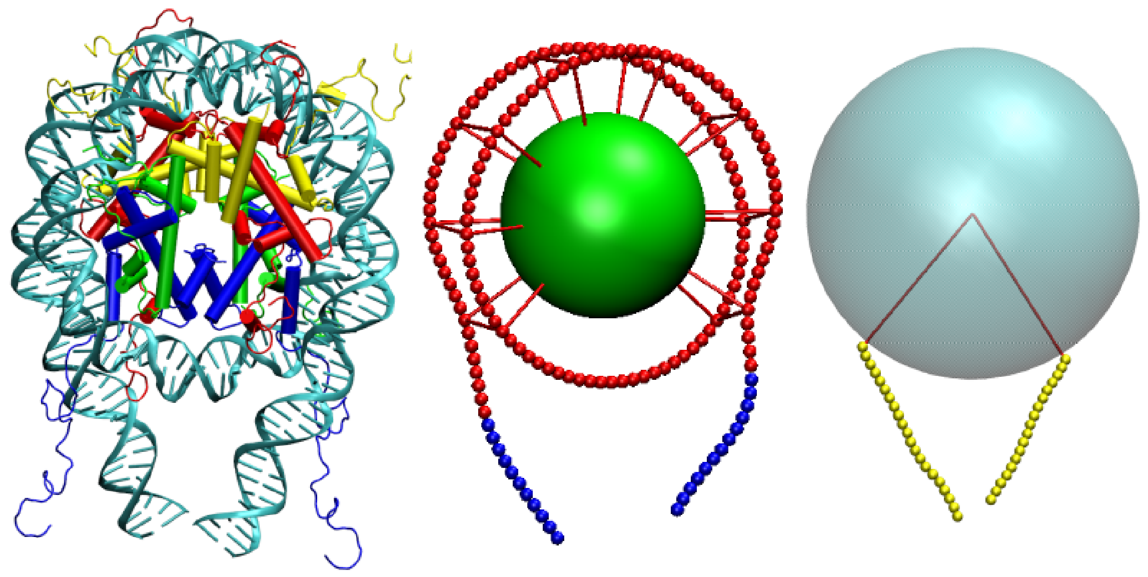


Figure 6.3: Various strategies can be employed to represent Masks. For an all atom model (left), one can prescribe a fixed set of base pair step parameters and dock an all atom representation of the histone core to the superhelix. Coarse-grain strategies may seek to explicitly represent the path of the DNA while reducing the histone representation (middle) or may represent the entire nucleosome as a single entity to which DNA linkers are attached (right). The genome dashboard framework can be applied to all of these structures, but it does not provide the physical model (energy functions) for any of them.

The above Masking strategy for nucleosomes can be applied to any protein-DNA complex. Linker DNA connecting Masks is often assumed to be free DNA, Figure 6.4, but in general, even linker DNA may be described by Masks, e.g. bent or less flexible linkers. Likewise, chemical modification of the DNA, e.g. methylation, which does not change the sequence but the physical-chemical properties of DNA, is a Mask.

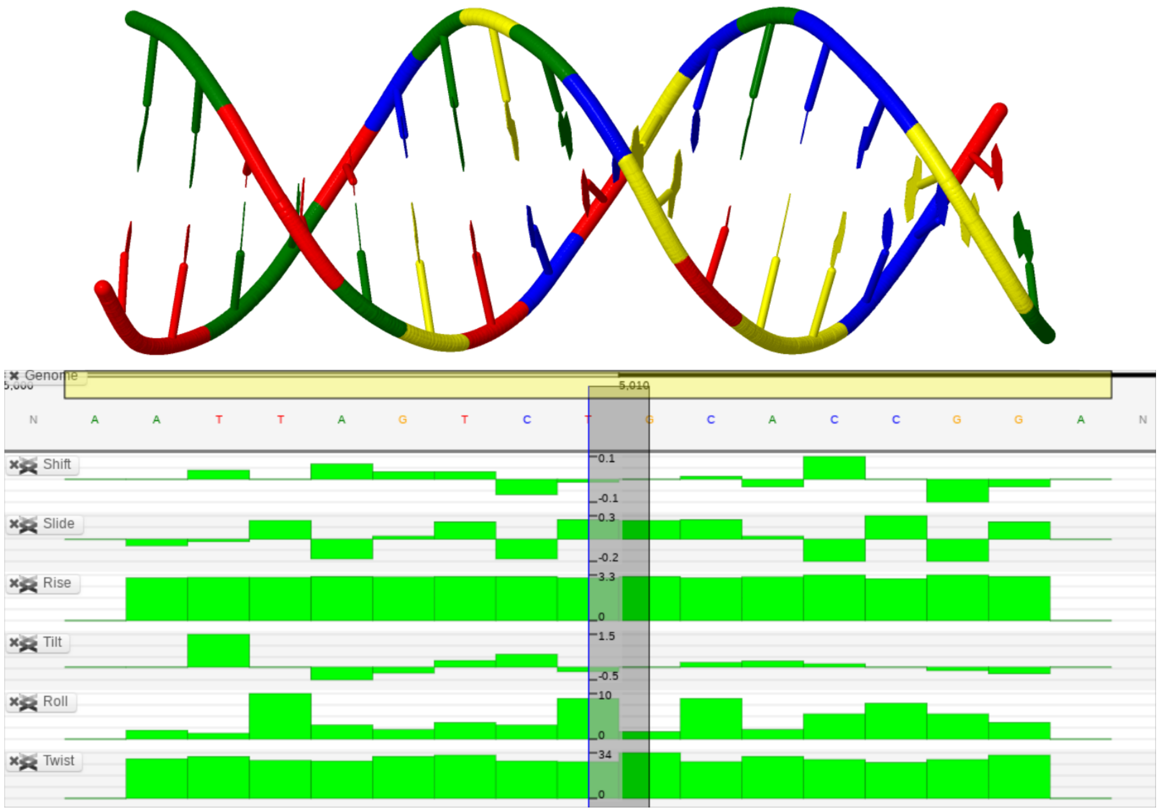


Figure 6.4: 3D structure and 1D informatics representations of the sequence "AATTAGTCTGCACCGGA" which contains all 16 possible base pair steps. This figure summarizes the structure and informatics data associated with any unmasked region of DNA in G-Dash.

In the context of a genome dashboard, chromatin folding is an informatics problem of describing the unique Masks and tracking their locations along a sequence of DNA. These Masks can be developed and manipulated based on informatics or physical analyses. In this manner, genome dashboards are designed to enable users to define and navigate chromatin folding landscapes efficiently.

6.3.2 Collision Detection in 1D

One problem occurs when a Mask is applied to a space curve with a certain strategy in one of the projects, G-Dash. The problem can be described as given a number of fixed length Masks, and assign values to each point on a space curve, and then find the minimal sum of the value of the point at the start of applied Masks such that all the Masks on the space curve do not overlap.

Inputs: Number of Masks, length of Masks, a list of values ($[v_1, v_2, v_3, \dots]$) represent the assigned values on a space curve.

Output: The occupancy of the start of the Masks ($[M_1, M_2, M_3, \dots]$), where the value donated as v_{M_i}, such that the sum of v_{M_i} is minimized.

Greedy algorithm

A straight-forward strategy is using the greedy algorithm, that is:

For i in the number of Masks, find the minimum position of the list of values, and check if putting a Mask at this position overlaps with previous ones. If not, set this position as a desired Mask position, and set this value as inf.

Algorithm 1 Greedy Algorithm for 1D Collision detection

```
1: $i \leftarrow 0$
2: $Mask\_index \leftarrow []$
3: while $i < \#\_of\_Masks$ do
4:     $index \leftarrow v_{min}$
5:     if $Check\_overlap(index, Mask\_index) == False$ then
6:         $Mask\_index \leftarrow Mask\_index.append(index)$
7:         $v_{index} \leftarrow inf$
8:         $i \leftarrow i + 1$
9:     else
10:        $v_{index} \leftarrow inf$
11:    end if
12: end while=0
```

This algorithm will return a local minimum of the solution. While it is quite fast, the time complexity is only O(n), so it does not necessarily provide the global minimum. It is not easy to find a global minimum, and in some cases, even the maximum value in $[v_i]$ could be in the minimum set of the Masks' positions. So, an exhaustive searching of possible solutions, the time complexity would be O(n^m), where m is the number of Masks.

Zero-One integer programming

In order to solve this problem, we have converted it into a zero-one integer programming problem. The zero-one integer programming is a NP-complete problem that uses a series of binary (1 and 0) answers to arrive at a solution when there are two mutually exclusive options. The mathematical model of a zero-one integer programming problem is described as

Target Function: $(Max/Min)z = c_1x_1 + c_2x_2 + \ + c_nx_n$

Condition Functions:

$a_{11}x_1 + a_{12}x_2 + \ + a_{1n}x_n \geq (\leq)b_1$

$a_{21}x_1 + a_{22}x_2 + \ + a_{2n}x_n \geq (\leq)b_2$

$\ldots$

$a_{m1}x_1 + a_{m2}x_2 + \ + a_{mn}x_n \geq (\leq)b_m$

where $x_1, x_2, \ldots, x_n = 0$ or 1

In the case of putting Masks on the space curve, let the $x_i = 1$ at the position i of the space curve where the Masks are located, and $x_j = 0$ where j is the position of Masks that are not located. Then the functions are:

Target Function: $(Min)z = v_1x_1 + v_2x_2 + \ + v_nx_n$

Condition Functions:

$x_1 + x_2 + \ + x_k \leq 1$

$x_2 + x_3 + \ + x_{k+1} \leq 1$

$\ldots$

$x_{n-k+1} + x_{n-k+2} + \ + x_n \leq 1$

$x_1 + x_2 + ... + x_n = num\_of\_Masks$

where $x_1, x_2, \ldots, x_n = 0$ or 1, and k is the length of a Mask.

The zero-one integer programming problem could be solved by branch and bound algorithm, which defines the problem as $P(x_1, x_2, , x_n)$, and the optimized solution is $f(x_1, x_2, , x_n)$. Then this problem could be treat as two sub-problems: $P1(0, x_2, , x_n)$ and $P2(1, x_2, , x_n)$, and the minimum of $f1$ and $f2$ is the solution of P, and by recursively applying this operation, a global minimum can be found.

Algorithm 2 Zero one integer programming

```
0: function MASK_POSITION($Mask_i$(Boolean), val, #_of_Masks)
1: if #_of_Masks == 0 then
2:     return
3: else
4:     return min($v_{Mask_i}$ + Mask_Position (0,val, #_of_Masks-1), Mask_Position (1,val,
       #_of_Masks))
5: end if
5: end function=0
```

Meanwhile, there are robust algorithms that exist to solve zero-one integer programming problems, such as intlinprog in Matlab. After we have set up the Target Function and Condition Functions, we can pass the coefficients to intlinprog in Matlab and get the solutions.

6.4 Computational Implementation

A genome dashboard is a finite state machine that can be efficiently developed using Model-View-Controller (MVC) design principles [38], Figure 6.5. This approach ensures that a dashboard's components are independent, replaceable, and extensible.

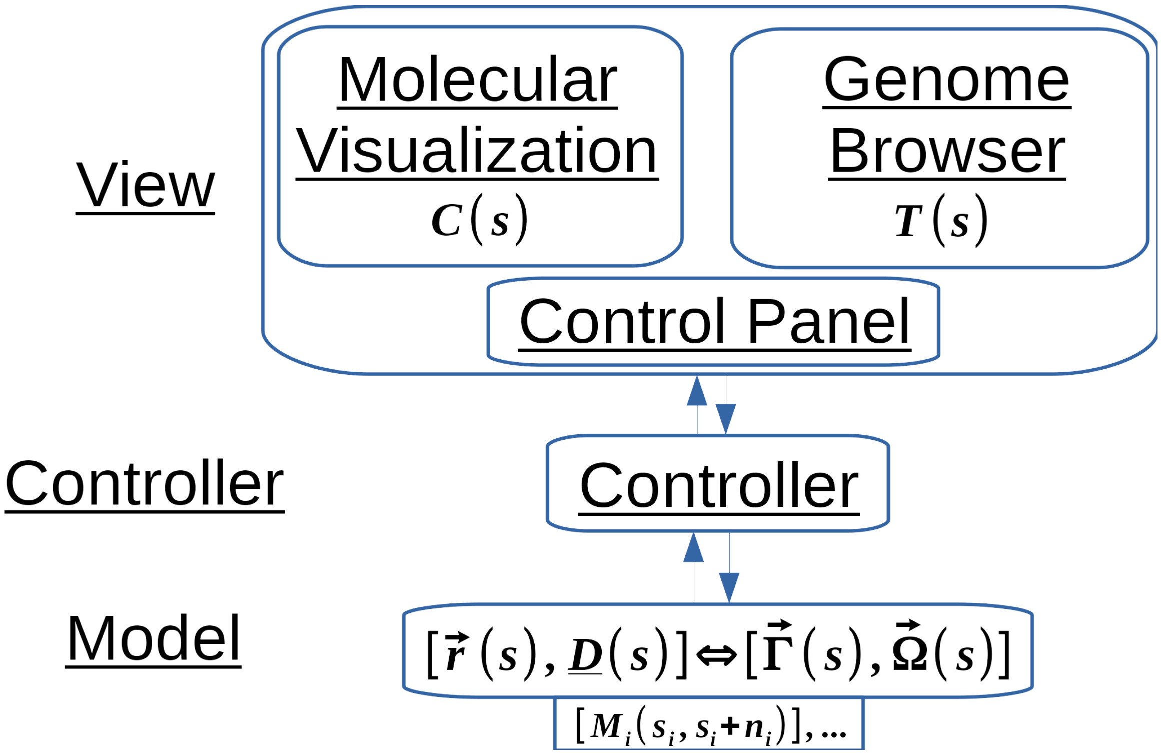

View
Molecular Visualization
$C(s)$
Genome Browser
$T(s)$
Control Panel
Controller
Controller
Model
$[\vec{r}(s), \underline{D}(s)] \Leftrightarrow [\vec{\Gamma}(s), \vec{\Omega}(s)]$
$[M_i(s_i, s_i + n_i)], \ldots$

Figure 6.5: Model-View-Controller (MVC) Design. Model: Laboratory frame $[\vec{r}(s), \mathbf{D}(s)]$ and material frame $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ descriptions of DNA as the common thread, an inventory of Masks $M(s)$, and procedures for converting between representations. View: a Molecular Visualization (MV) displays $C(s)$, a Genome Browser (GB) displays $T(s)$ and a Control Panel (CP) provides a graphical interface to the controller. G-Dash uses JSmol and Biodalliance for the MV and GB components, respectively. A commodity-off-the-shelf (COTS) approach enables a genome dashboard to use any desired MVs and GBs. Controller: manages the exchange of data between Model and Views.

The "Model" in the MVC schema is the data and related logic. For a genome dashboard, the Model includes the $[\vec{r}, \mathbf{D}]$ and $[\vec{\Gamma}, \vec{\Omega}]$ representations of DNA as a discrete (or even continuous [75, 76]) oriented space curve, the inventory of Masks,

$M_i(s_i, s_i + n_i)$, any associated track data, $T(s)$, and procedures for converting between representations. In general, the rotations and translations associated with a Mask may be large. If a Mask is a rigid entity, this information can be leveraged to improve performance. For example, representing all 147 base pairs of DNA and eight histones in a nucleosome as a single director frame along with a large deformation of the path of DNA reduces computational and data costs by approximately $n * 146$, where n is the number of nucleosomes containing 147 base pairs.

The "View" in the MVC schema provides the user interface and renders data. A genome dashboard includes a 3D/4D Molecular Visualization (MV) for rendering $[\vec{r}, \mathbf{D}]$, a Genome Browser (GB) for rendering $[\vec{\Gamma}, \vec{\Omega}]$, and a Control Panel (CP) as a graphical user interface to the Controller. A genome dashboard can be designed as a web application or a stand alone application. For web applications, javascript based MVs such as JSmol [30] and NGL Viewer [77] are optimal. For stand alone applications, MVs such as VMD [32] and PyMOL [78] are optimal. Likewise, for web applications, the GB should be javascript based, like Biodalliance [79]. For stand alone applications, JBrowse [80] or other modern genome browsers may provide advantages.

The "Controller" in the MVC schema manages the exchange of data between the View and the Model. For a given genome dashboard, the MV and GB can be COTS elements, but the Control Panel and Controller are application specific. We expect that different instances of the genome dashboard concept will target different users and utilize different physical models, or in the case of purely experimental data not even include a physical model. The Controller enables the user to manage the physical models and different strategies for managing the MV, GB, and CP will likely emerge, but the underlying Model remains as described above.

6.5 Results

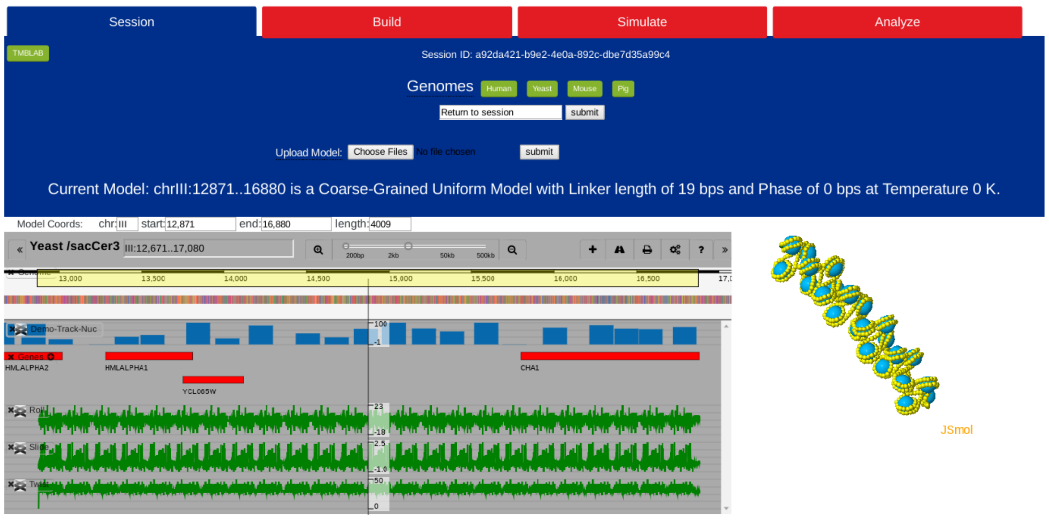
6.5.1 G-Dash Web Application

Overview: G-Dash is a prototype of the genome dashboard concept, Figure 6.6. It is a web application that uses HTML5, JavaScript and JSON(JavaScript Object Notation) for HTML functions and for embedding Biodalliance and JSmol in a single page, and PHP to pass data between modeling units. Bigwig [89] and 2bit formats are utilized to exchange and manage track data. The modeling units in the G-Dash prototype have evolved over time without selective pressure and are a collection of Unix and VMD scripts, FORTRAN, Python, and C tools. The G-Dash prototype is self-contained. Any user can install G-Dash on a computer with an Apache2 HTTP server configured to allow access to a user's "public_html" folder. However, there are a number of additional software packages that must be installed and configured. Interested users should contact the authors. A Python based implementation of the Model as an integrated compute kernel is under development. It will provide an API that can function as a command line tool independently of the graphical user interface and HTTP server. It will thus be suitable for automated workflows. This Python implementation of the Model will be publicly available via bitbucket.

Usage: In G-Dash, all-atom models are generated with 3DNA and parameterized with AMBER's tleap [31] modules. The atomic models include parmtop, crd and pdb formatted files that can be downloaded by the user to initiate modeling on their own computing resources. JSmol displays the pdb file using cartoon style by default. Coarse-grained models are stored in an xyz formatted file. The DNA space curve has Center Atoms, CA, director frame end points as H1, H2, H3 atoms, and octasome cores as OC atoms. We have applied the following rules for representing coarse-grained models in JSmol. DNA is represented by small beads and nucleosomes by large beads. A small green bead represents the start of the structure, and small yellow beads represent intermediate base pairs. Nucleosomes are represented by large blue beads, unless a steric class is detected. Then the beads are colored red to indicate a clash. Steric clash is only monitored for systems containing less than 30,000 base pairs in order to maintain the interactive nature of G-Dash. For all models, JSmol is fully functional so users can save files, change colors or representation schemes, measure distances etc., using the functions provided by JSmol. As embedded in G-Dash, Biodalliance allows users to choose between Human, Mouse, Yeast, and Pig genomes. G-Dash can be configured to use other public or private genome assemblies. For any genome, users enter the chromosome coordinates of interest or a search term to jump to a desired location. For example, entering CHA1 for the sacCer3 assembly jumps to chromosome coordinate III:5,798..26,880. Alternatively, users select a specific DNA sequence for modeling using the sequence selector (a yellow bar) in the top track of the genome browser.

In a genome browser, all data are displayed as tracks. In G-Dash, default tracks representing structural-informatics obtained from the Model are pre-selected and are automatically updated whenever a model is built, see also the Analyze section below. Biodalliance, like any modern genome browser, allows users to add tracks from numerous public or private resources. Users can manipulate the style, color, and max/min values associated with any track.

For the Control Panel, a tab based graphical user interface has been developed. The tabs correspond to familiar modeling tasks: start a session, build a model, simulate and analyze. The tabs are labelled accordingly and are described below. The G-Dash prototype demonstrates that established data and language standards can be exploited to achieve bi-directional exchange of data between informatics and structure so that any informatics track can inform a molecular structure and structural features can be extracted as informatics tracks.


Session ID: a92da421-b9e2-4e0a-892c-dbe7d35a99c4

Upload Model: Choose Files No file chosen

Current Model: chrIII:12871..16880 is a Coarse-Grained Uniform Model with Linker length of 19 bps and Phase of 0 bps at Temperature 0 K.

Model Coords: chr: III start: 12,871 end: 16,880 length: 4009

Yeast /sacCer3 III:12,671..17,080

Figure 6.6: G-Dash contains an embedded genome browser, Biodalliance, (bottom left), a Control Panel(top), and a Molecular Viewer, JSmol, (bottom right). See text for additional descriptions of Control Panel elements.

Session

The “Session” tab is used to initiate or restore a session. For each session, G-Dash provides a unique session ID, which can be used to return to a previous session using the submit button. Users start a new session by selecting the desired species and the chromosome coordinate to be modeled either by key-word search or selection with the yellow bar. A description of the current model is provided whenever a model is built. An externally developed 3D structure can also be associated with a genome assembly using “Choose Files” and the “submit” button. Currently, only DiscoTech [85] based pdb models are allowed. However, any model for which $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ data can be computed should be supported. If the model explicitly contains $[\vec{r}, \underline{D}]$ information or if the director frames can be systematically extracted from the model, then the algorithms described in Methods can be used to compute structure tracks. If, as is the case with the DiscoTech based models, only $\vec{r}(s)$ data is available, the director frames must be determined from a TNB analysis. The DiscoTech based models pose the additional challenge that there are nine base pairs per bead. The TNB approach is used to generate missing data with the assumptions that the DNA curve is shear free and has uniform twist.

The DiscoTech based models also lack sequence. Thus, the user chooses a starting location with the sequence selector (yellow bar) to associate the model with chromosome coordinates. Uploading DiscoTech based models thus demonstrates an important proof of concept of how to associate an external structure with the Model in a genome dashboard. Hi-C derived models can also be imported with this technique. This usage modality is a powerful tool for unifying externally developed models with the wealth of sequence data that is publicly available.

Build

The “Build” tab provides all the functionalities of ICM [64]. The “Global Variables” apply to all models under “Build”, “K/X” represents the stiffness(K), and free DNA geometry(X) (Here, X is shorthand notation for $[\vec{\Gamma}(s), \vec{\Omega}(s)]$); “T” is the temperature in Kelvin and determines the amount of thermal variation to be added to the helical parameters for regions of free DNA. The default value is zero, which means the sequence specific average values of $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ as observed in [72] are used. Only DNA that is not Masked is subject to thermal variations. As we reported previously [12], random thermal fluctuations added to the nucleosomal DNA helical parameters are sufficient to destroy the nucleosome superhelix. For this reason, thermal variations are not added to any region of DNA that is masked. Options “All-Atom”(under 2,000 base pairs), “Coarse-Grained”(between 2,000 to 30,000 base pairs), “Super Coarse-Grained”(over 30,000 base pairs) are provided and automatically highlighted according to the length of sequence selected.

DNA: As with ICM, all-atom or coarse-grained models of DNA require only a sequence and temperature to be specified. G-Dash reads DNA sequence information directly from Biodalliance. Users select the sequence using the sequence selector (yellow bar in genome browser window), and the chromosome coordinates of the selected sequence appear above the genome browser as the Model coordinates. G-Dash will generate a single 3D structure or an ensemble containing ten different sequence specific thermal variants by choosing one or ten frames.

Uniform: The Uniform option provides a uniform linker length between Masks, and all Masks correspond to 1KX5. As in ICM, the user can control the phase and linker length. The default value is a phase of 0 and linker length is 19 base pairs. The first nucleosome is placed at the start of the DNA sequence or shifted by the phase value. All successive nucleosomes are spaced 19 base pairs from the end of the previous one. This produces a regular chromatin fiber structure. As shown in Figure 6.7 E, the uniform chromatin fiber is not necessarily straight even when the temperature is zero because the linker possesses sequence specific conformation properties. Figure 6.8 also includes two uniform models with very short linker lengths.

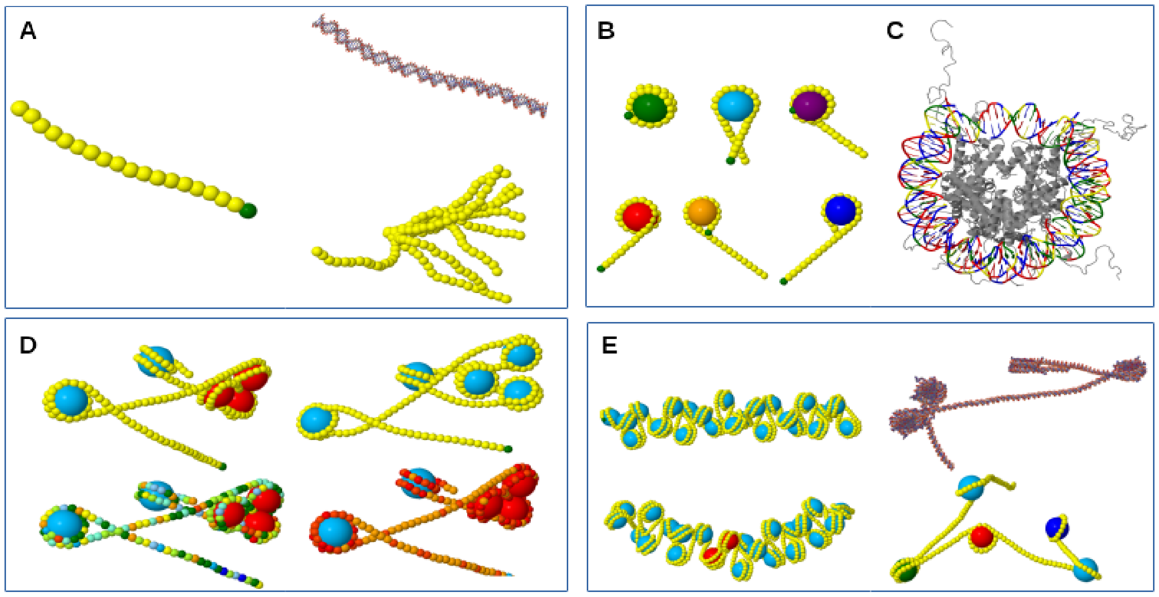


Figure 6.7: G-Dash Modeling: A) A single sequence specific coarse-grained or all-atom model of DNA or ensemble of conformations representing thermal variations can be generated and displayed. B) Different conformations of the nucleosome superfamily of states can be assigned to specific locations using informatics tracks or the Nucleosome Widget. C) An all-atom model for any single nucleosome can also be generated. D) The default coloring of coarse-grained nucleosome models uses small yellow beads for DNA and large blue beads for the histones. Steric clash is indicated by red nucleosomes and may be resolved with a short minimization. Tracks from the genome browser can be mapped on the coarse-grained models. E) Options for generating structures associated with various distributions of nucleosomes along the DNA are provided and can be converted to all-atom models.

Track: With this option, any single informatics track can be used to position nucleosomes. By uploading nucleosome positions as a track in the genome browser, a user can overlay a desired chromatin folding motif on any segment of DNA. The chosen track may be experimentally or theoretically determined nucleosome positions or any combination of data uploaded by the user as a track. To achieve this method of chromatin folding, the user selects “Track” in the “Build” tab and clicks the desired track name. The selected track will show up so that users can verify the selection. Clicking “Build Model” will build the model.

Auto: The auto option automatically places nucleosomes in the Nucleosome Energy Landscape, E_{1D}, utilizing the same method developed for ICM. The default is 70% occupancy of the maximum number of allowed nucleosomes and a minimum linker length of 19 base pairs, Figure 6.7 E. Varying the minimum linker length determines how extended or condensed this non-uniform model will be. In general, any track can be used as the energy landscape, thus opening possibilities for arbitrarily complex knowledge based potentials for nucleosome positioning.

Simulate

The "Simulate" tab is designed to provide a gateway to models and compute engines developed by others. A Nucleosome Widget has been developed for manipulating and manually positioning nucleosomes in the one dimensional Nucleosome Energy Landscape, E_{1D}. G-Dash currently supports a generation of all-atom models of individual nucleosomes and a method for quickly relaxing steric clashes in coarse-grained chromatin models. We expect to provide gateway support for various coarse-grained models of DNA, nucleosomes and chromatin as G-Dash continues to evolve.

Nucleosome: Here, G-Dash is configured to make an all-atom mono-nucleosome model for any nucleosome selected from the Nucleosome Energy Landscape by clicking the "All Atom" button in the "Simulate" tab, Figure 6.6 C. The models are based on the 1KX5 x-ray structure [90] and include AMBER formatted parmtop, crd, and pdb files that can be downloaded for computational studies by the user. For these models, a DNA superhelix is constructed for the selected sequence of DNA and docked onto 1KX5's histone octamer.

Chromatin: Chromatin modeling is achieved by assembling linker (free) DNA interspersed with nucleosome Masks $M(s)$. The Nucleosome Widget represents and controls the location and type of nucleosomes in a Nucleosome Energy Landscape. A Mask Widget is the general purpose solution for managing inventories of numerous Masks. Such a widget should also allow the user to define and edit Masks. In the G-Dash prototype, users are able to add, delete, move, or alter nucleosome types using the Nucleosome Widget by clicking on the corresponding block in the Nucleosome Energy Landscape. Collectively, these tools provide a novel means of investigating structure-function relationships.

Minimization: As a gateway, the Simulation tab in the G-Dash prototype is limited to providing access only to short minimizations that run on the G-Dash web server. Steric clashes or knotting may occur for any model assembled in the material reference frame. When these problems occur, as indicated by red nucleosomes, users should click the "Minimize" button to relax the model. G-Dash utilizes LAMMPS [91] to relax G-Dash structures represented in the laboratory reference frame using a minimal coarse-grained model. This model utilizes harmonic bonds and angles to preserve local geometry and a soft pair repulsion to push apart overlapping beads. No effort is made to capture electrostatic or van der Waals interactions in this model. The purpose of the LAMMPS compute engine is only to solve steric clashes and knotting as rapidly as possible without causing significant variation from the initial structure. It is not intended for thermodynamic sampling. In this regard, relaxation is a necessary first step in creating inputs to be used for more extensive sampling with external compute engines. The genome dashboard framework supports generating atomic, coarse-grained and super-coarse grained models of chromatin as shown in Figure 6.3. If the problems can be resolved with a short minimization, the user may assume the indicated 3D conformation can be physically realized and continue with more sophisticated models, such as [92, 93]. If the problems are not solved by the minimizer, the model is likely not physically realizable. The “Minimize” button will appear under “Chromatin” in the “Simulate” tab only after a “coarse-grained” model has been built.

Analyze

Structural Informatics: A unique feature of genome dashboards is the idea of structural-informatics. Genome dashboards unify the 3D structure and 1D informatics descriptors such that any sequence data can be mapped onto a 3D structure. Likewise, the structure data can be extracted from a 3D model and presented as 1D informatics tracks. Whenever a model is imported or built in G-Dash structural informatics tracks $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ are generated and automatically updated in the genome browser. These tracks are highlighted in green in G-Dash and include helical parameter data, such as “Roll”, “Slide”, “Twist”, energy from the nucleosome energy landscape, and nucleosome occupancy data. The complete set of structural-informatics tracks appear under the Modeling Data tab in Biodalliance’s track management tools. Each of these tracks can be downloaded for analysis against any other track data in an informatics workflow.

It is also useful in the analysis of structure-function relationships to map informatics data onto a 3D model, Figure 6.7 D. G-Dash provides two methods of doing this. Individual nucleosomes can be selected in the Nucleosome Widget and assigned specific colors, or an informatics track can be utilized to color the DNA in a model at base pair resolution. Selecting a track and using the “trackcolor” button in the control panel pushes informatics data onto the 3D model. The “trackcolor” button will appear under the 3D model only when a coarse-grained model is built. JSmol’s command console is fully functional so it can also be used to modify the display and extract structural data. However, it does not currently have direct access to informatics data in the genome browser or the Model.

The “Analyze” tab is intended to provide a collection of analysis metrics. In G-Dash, two metrics are automatically updated every time a model is built: a $\alpha - \beta$ plot and a distance-distance matrix, Figure 6.8. The $\alpha - \beta$ plot is a Woodcock Equivalent (WE) Model, that measures α, the angle between the centers of 3 nucleosomes, and β, the dihedral angle between the centers of 4 nucleosomes[65, 86]. For this analysis, the reported α and β values are computed as if the linkers were straight even if they are not. For this reason they are termed “equivalent” plots. The distance-distance matrix reports the center to center distance between each nucleosome. In the future, additional metric will be added to the standard analysis library such as Linking Number (Lk), Twist (Tw), and Writhe (Wr)[94, 95]. These topological descriptors underscore the ability of genome dashboards to unify data from diverse perspectives.

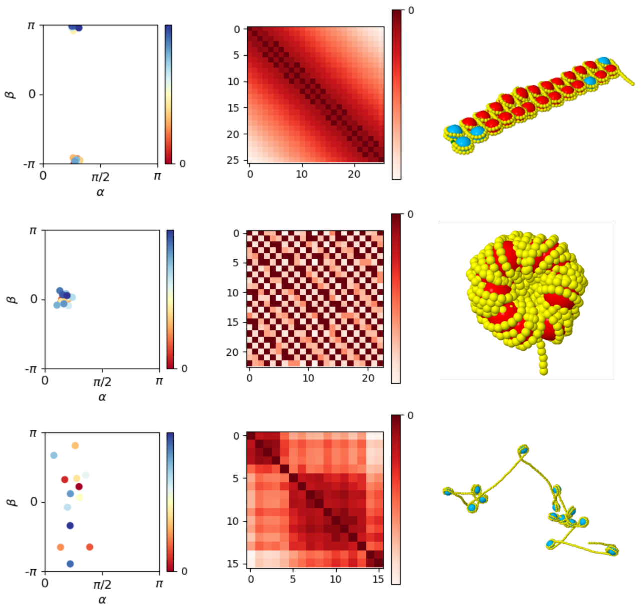


Figure 6.8: Woodcock Equivalent and Distance-Distance Plots: (Left) Woodcock Equivalent Plots provide a two angle representation of the model. (Center) Nucleosome based distance-distance matrices. (Right) 3D models. (Top to bottom): Uniform model with 4 base pairs long linkers, uniform model with 25 base pairs long linkers, and "auto" model with nucleosome positioned at minimal in the Nucleosome Energy Landscape.

Based on the framework proposed above, we have developed a minimal genome dashboard named "G-Dash". Here, we demonstrate two examples using G-Dash to show how genome dashboards contribute to our understanding of biological function by the unification of informatics and physical structures.

6.5.2 Informatics to Physical Structure

A hormone response element (HRE) is a specific sequence of DNA representing 15 base pairs. Selective binding of an activated hormone receptor (HR) to the HRE is a critical component of the hormone response mechanism, see Figure 1-41 of [81]. A variant of this gene regulatory mechanism is employed to control numerous physiologic functions in all higher organisms. To demonstrate the power of data unification achieved with our G-Dash application, we have identified estrogen response elements (ERE) using ERE-Finder [82]. This informatics data is displayed as the ERE Track in Figure 6.9. All experimentally determined nucleosome positions for the human genome [83] are also displayed in Figure 6.9 as the Nuc-Pos Track. These Tracks provide locations for EREs and nucleosomes. The 1D representation in the genome browser is insufficient to determine whether or not the locations are physically realizable. Nonetheless, these tracks are sufficient to identify several regions of interest. One of them is associated with chromosome 6 and coordinate location $168,131,722$ to $168,132,130$. Here, we find three overlapping nucleosomes and an ERE that appears to function as a classic switching mechanism. We explore this hypothesis with G-Dash by generating physical structures.

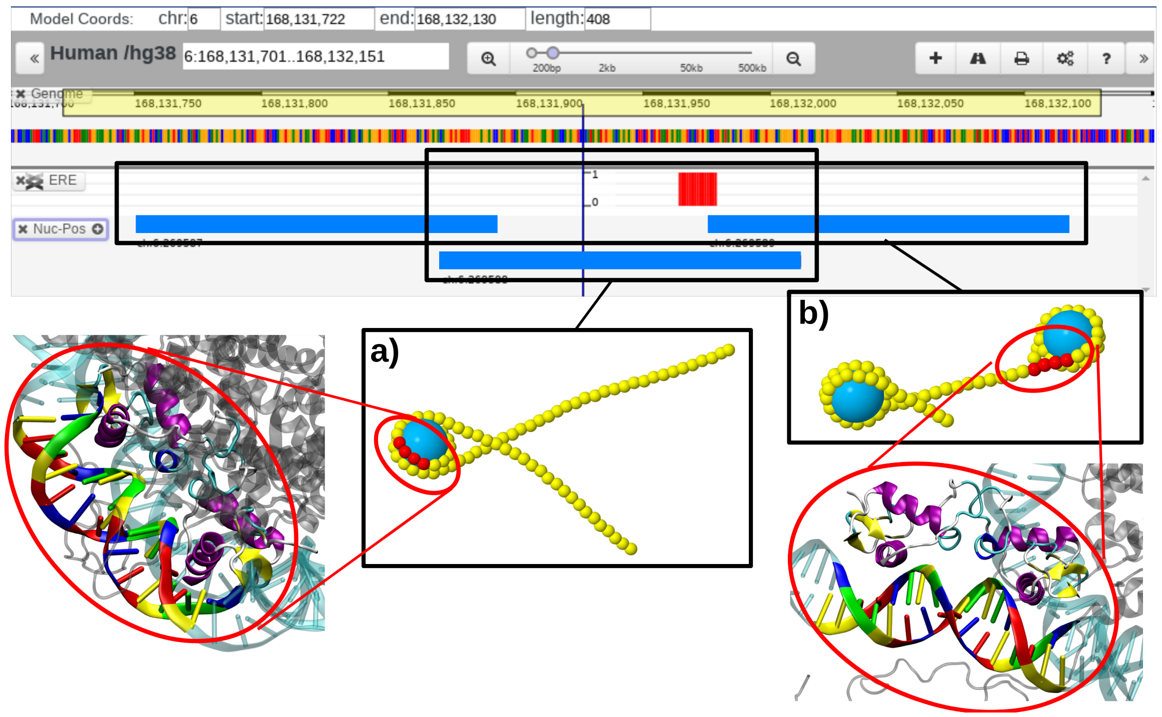


Figure 6.9: Black boxes: $C(s)$ and $T(s)$ representations of two allowed states. Upper boxes are $T(s)$ representations of nucleosome positions (blue bars) and an ERE (red bar). Lower boxes are $C(s)$ representations (small beads represent 5 base pairs large beads represent histone octamers). Red ellipses: The corresponding all-atom structures with the estrogen receptor DNA-binding domain docked to the DNA as in PDB entry 1HCQ. a) The ERE is located within a nucleosome with the major groove facing inward. The receptor is prohibited from binding. b). The ERE is located in a nucleosome free region. Docking 1HCQ indicates that the ERE is physically accessible.

We first selected the single nucleosome shown in the bottom of the Nuc-Pos Track in Figure 6.9 and generated a coarse-grained representation in G-Dash with the compute tools that also drive ICM Web [64]. Mapping the ERE location onto a coarse-grained physical structure, an informatics problem, provided the ERE's location, but without knowledge of major groove orientation, one can still not determine the accessibility of the ERE site for ER binding. We constructed an all-atom model using the base pair step parameter data generated by the ICM Web compute tools in G-Dash. With the all-atom model, we see that the major groove is actually facing towards the histones. This prevents the estrogen receptor DNA-binding domain (1HCQ) from binding to this region of the DNA major groove. To bind 1HCQ to the all-atom model, we downloaded the all-atom model from G-Dash, then loaded it and the 1HCQ into VMD. A simple VMD script fits the DNA in 1HCQ to the DNA in the G-Dash all-atom model.

We used the same approach to model the two nucleosomes shown in the top of the Nuc-Pos Track in Figure 6.9. Mapping the ERE location to the coarse-grained model suggests the ERE may be accessible to ER. Using the same procedure and script as before, we find that 1HCQ can physically access this ERE with the nucleosome present. We point out that 1HCQ is only the DNA binding domain of the estrogen receptor. There exist steric conflicts between the ER-DBD and the histones so this is not the complete story; but it strongly suggests this site as a candidate for a genetic switching mechanism.

With this example, we demonstrate with G-Dash how informatics is used to construct a physical structure that extends and validates the interpretation of the informatics data. We emphasize that any informatics track or combination of tracks can be used to inform the physical structure. All-atom and coarse grained molecular mechanics can be used to further explore this structure. The choice of physical model depends on the exact question being posed.

6.5.3 Physical Structure to Informatics

Models of chromatin are rapidly maturing. As the models develop, there is increasing demand to capture biologic realism, including DNA sequence, experimentally determined nucleosome positions, states of chemical modification etc. Without a genome dashboard, manually curating informatics data to build a “biologically inspired” model is a time-consuming and tedious task that informs the initial model but does not necessarily support the interpretation of modeling results. Genome dashboards enable any available informatics data to be easily associated with an existing physical model or experimentally determined structure to achieve a meaningful biological interpretation. Here, we import a HOXC Mesoscale model generated by the Schlick lab [84] into G-Dash to demonstrate how informatics can be overlayed onto an existing physical model or structure.

In G-Dash we provide an upload function that is specific for DiscoTech based models [85]. We upload and convert the HOXC Mesoscale model into $C(s)$ and $T(s)$ representations, Figure 6.10. The HOXC model utilizes a 9 base pair per bead model that includes both the location and orientation of each bead and each DiscoTech based nucleosome, i.e. $\vec{r}(s)$ and $\mathbf{D}(s)$ are provided for both the ribbon and the Masks. The DiscoTech nucleosomes are represented as Masks consisting of a single director frame and center atom. We calculate $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ based on the $[\vec{r}(s), \mathbf{D}(s)]$ data provided for DNA beads and Masks. The $[\vec{\Gamma}(s), \vec{\Omega}(s)]$ values computed are no longer DNA base pair step parameters; however, they still represent an oriented space curve or ribbon. We refer to them as generalized step parameters (GSP) and utilize the same naming conventions as for the DNA parameters: Tilt, Roll, Twist, Shift, Slide and Rise, and display them as informatics tracks in the genome browser. Twist and Rise are displayed as green structural informatics tracks in Figure 6.10-d. Our generalized parameter values differ significantly from those associated with DNA. Small angle approximations are no longer valid.

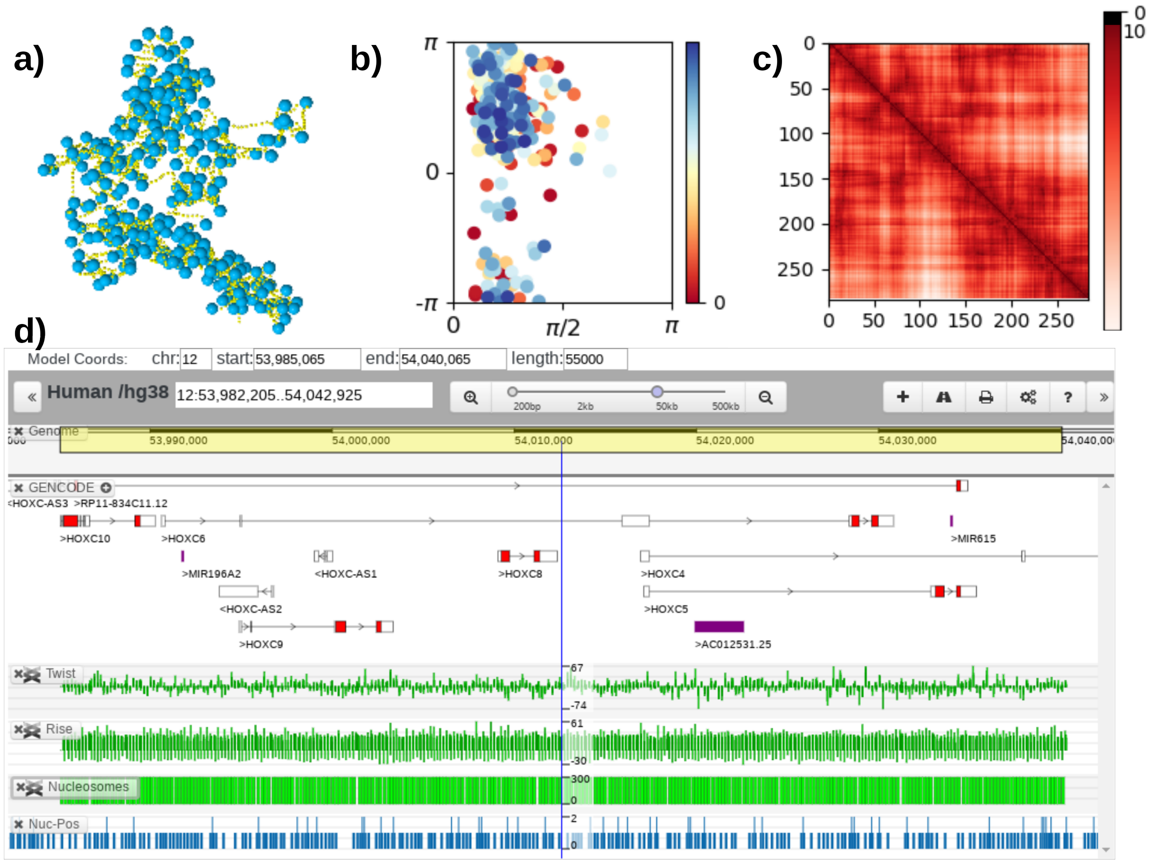


Figure 6.10: a) HOXC coarse-grained model of chromatin containing approximately 55 k base pairs of DNA and 284 nucleosomes. Uploading the HOXC model to G-Dash generates: b) a two-angle representation of the HOXC model, c) a distance-distance matrix based on nucleosome centers of mass, and d) structural informatics data. "Generalized Helical Parameter" ("Twist" and "Rise") and nucleosome position ("Nucleosomes") data are displayed alongside experimentally determined nucleosome positions ("Nuc-Pos") and other informatics data ("Gencode").

To determine if either of the E-R or E-A algorithms is suitable for our GSPs, we have converted all six HOXC models reported in [84] from the $C(s)$ to the $T(s)$ and back to the $C(s)$ representations. Each G-Dash converted model contains approximately 1779 discrete points, represents 55,000 base pairs and 284 nucleosomes and has a unique conformation. The RMSD between the initial and final $C(s)$ structures are computed using VMD's RMSD functions. The E-R approach yields RMSD values ranging from 0.03 Å to 0.15 Å after alignment, and the E-A approach yields RMSD values ranging from 0.002 Å to 0.003 Å after alignment, Table 6.1. Our experience is that both the E-A and E-R methods are acceptable algorithms for implementing "The Model" even when thermal fluctuations or deformations associated with nucleosomes and chromatin, or even generalized step parameters, are modeled.

Table 6.1: RMSD values associated with converting the original $C(s)$ representations of each HOXC model [84] to "generalized step parameters" and then converting back to a $C(s)$ structure using the Euler-Rodrigues (E-R) and Euler-Angle(E-A) methods as described in the text:

Model	Methods	
	E-R	E-A
	(Å)	(Å)
HOXC	0.153	0.002
Life-like	0.154	0.002
Life-like-Ac	0.118	0.002
Life-like-LH	0.050	0.002
uniformNFR	0.059	0.003
uniformNRL	0.028	0.003

The HOXC model was constructed for a specific sequence of DNA: HOXC10 of the annotated human genome assembly 38 that begins at chr12:53,985,065. However, HOXC models provided do not contain sequence information because the model itself is sequence independent. Thus, whenever a DiscoTech model is uploaded into

G-Dash, it must be aligned with a sequence using the yellow sequence selection bar, Figure 6.10. If the sequence information is included in the model, the model can be automatically aligned to the data in a genome browser. Once aligned, the structural informatics tracks enable us to compare the nucleosome positions used in the uploaded model (green Nucleosomes Track in Figure 6.10) to experimentally determined the nucleosome positions (blue Nuc-Pos Track in Figure 6.10). It is clear that the two tracks differ. Resolving these differences promises to advance our understanding of both the experimental and modeling data.

To complete our multi-dimensional representation of chromatin folding, we have incorporated 2D representations into G-Dash. Figure 6.10-b is a two-angle plot and Figure 6.10-c is a distance-distance matrix plot. As with the structure tracks, these representations are automatically generated for structures containing sufficiently many (more than 3) nucleosomes. The two-angle plot is a Woodcock Equivalent (WE) Plot [65, 86]. On these plots, α is the angle between the centers of three adjacent nucleosomes, and β is the dihedral rotation angle obtained from the centers of four adjacent nucleosomes. For this analysis, the reported α and β values are computed as if the linkers were straight, even if the linkers are not. For this reason, we have adopted the label "Woodcock Equivalent" Plot. The distance-distance matrix reports the center-to-center distance between all nucleosomes. Unlike the exchange of data between the informatics (1D) and physical structures (3D), the WE Plot data and distance-distance maps are one directional. The 2D representations are obtained from the 3D model but cannot be used to generate 3D structures. Methods exist for this purpose [63] but have not yet been implemented in G-Dash.

6.5.4 Performance

We have implemented both E-A and E-R methods in python, and record the time to convert n base pairs of **SP** to **RD**, as shown in Table 6.2. We can see that both timing of E-A and E-R methods are linearly increasing with more number of base pairs. The E-A method is faster than the E-R method for it does not require a solution to the square root of the matrix. Both methods are fast and can be optimized with either parallel computing or the choice of programming language (e.g. FORTRAN).

Table 6.2: Timing of E-A and E-R methods. The PC and CPU specification is shown in table 4.3 and table 4.4

Number of base pairs	E-A (s)	E-R (s)
50	0.0036	0.048
100	0.0072	0.052
200	0.015	0.098
500	0.037	0.25
1000	0.072	0.51
2000	0.15	0.96
5000	0.35	2.44
10000	0.76	5.01
20000	1.47	9.50
50000	3.6	24.78
100000	7.45	50.35
200000	14.45	98.82
500000	36.45	242.64
1000000	80.54	490.07

6.6 Conclusions

As a working example of a genome dashboard, G-Dash demonstrates that informatics and physical structures can be unified in a web based application in real time. Our tests demonstrate that interactive usage can be achieved for systems containing 10,000 to 50,000 base pairs or coarse-grained beads. The algorithms for converting informatics to physical structures and physical structures to informatics work in both directions for base pair resolution structures using either the Euler-Rodrigues (Curves+) or Euler-Angle (3DNA) based method. We believe our application to the HOXC model is the first demonstration that both the Euler-Rodrigues and the Euler-Angle algorithms can also be applied to coarse-grained models discretized well beyond the base pair level, Figure 6.11. We label the internal coordinates "generalized step parameters" in this case.

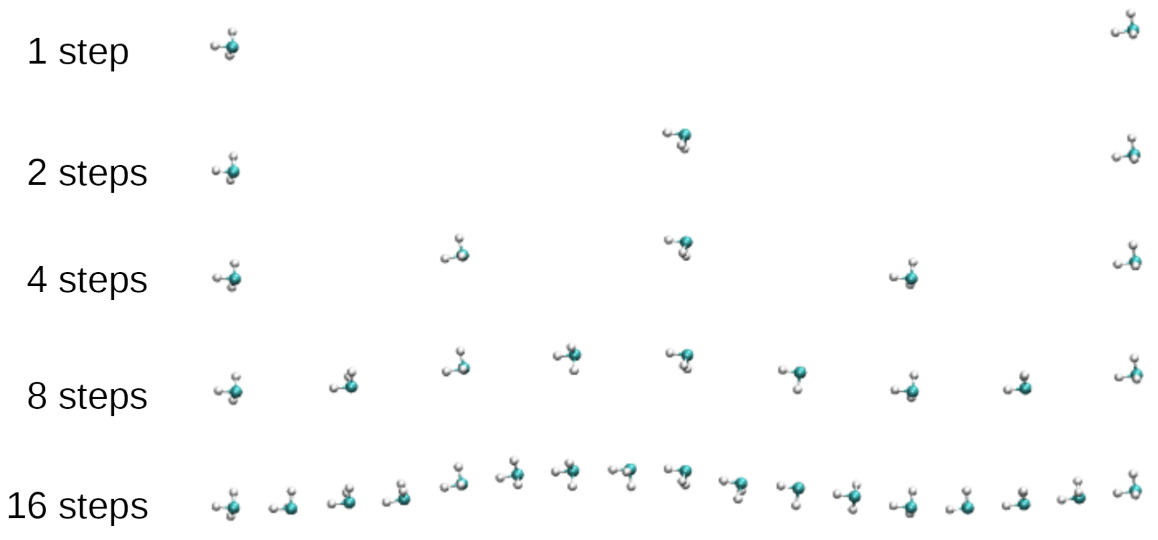


Figure 6.11: Demonstration of "Generalized Step Parameters" (GSP) as a means of reducing 16 base pair steps to 8, 4, 2, or 1 generalized steps. The ribbon represented corresponds to the "AATTAGTCTGCACCGGA" model shown in Figure 6.4, but generalized step parameters can be used to describe any ordered collection of the director frames.

G-Dash also demonstrates that a commodity-off-the-shelf approach coupled with Model-View-Controller principles can be employed to efficiently develop genome dashboards that are customizable, extensible and portable. G-Dash can generate atomic or coarse-grained models of DNA, nucleosomes, and chromatin by combining any experimentally or theoretically determined informatics data. Coupling G-Dash's atomic modeling capabilities with high-performance, high-throughput workflows, and our TMB Library of nucleosome simulations [34] provides a software ecosystem for overnight comparative molecular dynamics simulations of nucleosomes [87]. Such models are necessary for developing designer nucleosomes and assessing protein-DNA interactions in their native context [88].

The genome dashboard framework achieves unification of informatics and physical structures; the challenge is data representation. Genomics data has well-defined data formats, but there are numerous data formats for Cartesian coordinate data and no established conventions for representing the director frame data or step parameters. We have demonstrated that the E-R and E-A algorithms can support multi-scale and multi-dimensional modeling of DNA as the common thread in chromatin using generalized helical parameters. We deliberately avoided energetic considerations and focused on structure. The genome dashboard user or developer must decide which energy model is most appropriate for their particular application. We thus expect many genome dashboards to be developed, each tailored for a specific use.

As described here, genome dashboards are designed to work with chromatin folding, but the concept and framework are not limited to chromatin, eukaryotes or even DNA. The informatics data can be any data associated with a 1D indexing system, e.g. protein sequence or a SMILES string. The structure data can be a slender body or an ordered collection of points in space that are linked to form an oriented space curve. The latter includes observations of nucleosomes with super-resolution microscopy. If the sequence ordering of nucleosomes in a microscopy image can be determined and the director frames assigned to each nucleosome, it will be possible to unify 1D, 2D, 3D, and 4D representations of chromatin as shown in Figure 6.10.

CHAPTER 7

CONCLUSIONS

Multi-dimensional and multi-scale modeling have been studied in this dissertation. Mathematically, the idea of oriented space curve is the essential concept that connecting a 1D internal frame with 3D external frame. We have provided the mathematical integration and differentiation of these conversion, and two discrete method E-A and E-R that can be used to compute the conversions bi-directional. A space curve is at best represents a slender object. SMILES and Masks provide functionalities of extended slender objects into branched objects and any objects. Computationally, we have proposed a framework that unites these representations together by the Model-View-Controller (MVC) concept. Two algorithms aiming to solve problems occur at applying Masks and collision detecting are also described. The zero-one integer programming is seeking for the solution of a NP-complete problem, that placing Masks at the minimal of given informatics with restrict of the length of the Mask and they cannot overlap. This solution gives us a way to identify the global minimal/maximum of this type of problem. Zero-one integer programming statistically increased the accuracy. The GJK algorithm provides a solution for identify collisions of any 3D objects, which computationally reduce the time complexity.

The mathematical analyses of the differences between continuous and discrete construction of oriented space curves are also described. The methods appears for decades, but not much effort has been applied to study the differences between them. However, even though E-A, and E-R methods are robust tools for constructions of DNA, we do identify some properties that E-A or E-R method does not catch with their own set-ups. Particularly, they do not distinguish Shear Helix and Torsion Helix, which mathematically have clear patterns.

Based on this space curve concept and related algorithm, a toy named "Magic Snake" has been studied. We simplified the 1D internal frames into one rotation sequence. With this rotation sequence, we have proven some mathematical properties of the Magic Snake. Together with the idea of line skeleton representation of a Magic Snake, we also designed the Magic Snake to fit into any space curve, and expended a Magic Snake into more complex shapes. Then by applying the SMILES and MVC design strategy, we designed a Nanocar racing web application, that connecting 1D chem-informatics, 2D chemical structure, 3D molecular model, and 4D simulation. It also connected simulations with initial data access, to set ups, to result analyses, to data maintaining and sharing as a whole workflow. G-Dash applies all the related mathematical foundation and computational implementation that unifies the physical structures with bio-informatics, and the genome dashboard framework is then developed. It provides not only the abilities of modeling through a single base pair of DNA to the entire chromatin of coarse-grained and all-atom models, but also provides the interaction between computational modeling with biological informatics.

Recall the purpose for this dissertation is that the bridging between numerous amount of 1D informatics data and 3D physical modeling can be simulated with exponential increasing computation power. We have achieved this by multi-dimensional and multi-scale modeling that unifies different representations of data. We demonstrated the usefulness of our research mathematically with the example of “Magic Snake”, computationally with the example of “Nanocar Racing Web application”, and combined both mathematical and computational into the example of “Genome Dashboard”. As a result, from a mathematical point of view, this research is capable of modeling any object and representing the object into multi-dimensions (1D, 2D, 3D, and 4D), and make coarse-grained or all-atom models multi-scale. From a computational point of view, this research is capable of implementing any such multi-dimensional and multi-scale models into a portable and extensible dashboard, which enables the unification of data representations, as well as the communication between researchers from different research areas.

In summary, we have contributions to:

1. Design a framework to achieve multi-dimensional and multi-scale modeling with space curves, SMILES, and Masks.
2. Evaluate methods (continuous and discrete) for studying space curves.
3. Applications of this framework to study Magic Snake toys, Nanocar Racing, and Genomics.
4. Evaluate algorithms for collision detection in both 1D and 3D.

BIBLIOGRAPHY

[1] Jung J., Nishima W. and et al. (2019) Scaling molecular dynamics beyond 100,000 processor cores for largescale biophysical simulations. *J. Comput. Chem.*, **40**, 1919-1930.

[2] Shiva Singh (2018) The hundred-dollar genome: a health care cart before the genomic horse. *CMAJ.*, **190(16)**, E514.

[3] Simo J.C., Marsden J.E. and Krishnaprasad P.S. (1988) The Hamiltonian structure of nonlinear elasticity: the material and convective representations of solids, rods, and plates. *Arch. Ration. Mech. Anal.*, **104(2)**, 125-183.

[4] Dickerson R.E. (1989) Definitions and nomenclature of nucleic acid structure component. *Nucleic Acids Res.*, **17**, 1797-1803.

[5] Society of Naval Architects and Marine Engineers (1989) Section 3 - Ship Responses to Regular Waves. *Principles of Naval Architecture.*, **III**, 41.

[6] Struik D. J. (1988) Lectures on Classical Differential Geometry: Second Edition *Dover Publications.*

[7] el Hassan M.A. and Calladine C.R. (1995) The assessment of the geometry of dinucleotide steps in double-helical DNA; a new local calculation scheme. *J Mol Biol.*, **251**, 648-64.

[8] Lu Xiang-Jun and Olson Wilma K (2003) 3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. *Nucleic Acids Res.*, **31**, 5108-5121.

[9] Gonzalez O., Petkeviciute D. and Maddocks J.H. (2013) A sequence-dependent rigid-base model of DNA. *J Chem Phys.*, **138**, 055102.

[10] Lavery R., Moakher M. and et al. (2009) Conformational analysis of nucleic acids revisited: Curves+. *Nucleic Acids Res.*, **37**, 5917-5929.

[11] Babcock M.S., Pednault E.P. and Olson W.K. (1994) Nucleic acid structure analysis. Mathematics for local Cartesian and helical structure parameters that are truly comparable between structures. *J Mol Biol.*, **237**, 125-156.

[12] Bishop T.C. (2008) Geometry of the Nucleosomal DNA Superhelix. *Biophys J.*, **95(3)**, 1007-1017.

[13] Albie Fiore (1981) Shaping rubik's snake. *Penguin Books.*

[14] Zeng D., Li M. and et al. (2018) Overview of Rubiks cube and reflections on its application in mechanism. *Chinese Journal of Mechanical Engineering (English Edition)*, **31(4)**.

[15] Charles Fenyvesi (1981) Rubik's snake of "infinite possibilities" *The Washington Post.*

[16] Iguchi Kazumoto (1998) A toy model for understanding the conceptual framework of protein folding: Rubik's Magic Snake Model. *Modern Physics Letters B*, **12(13)**, 499-506.

[17] Iguchi Kazumoto (1999) Exactly solvable model of protein folding: Rubik's magic snake model. *International Journal of Modern Physics B*, **13(4)**, 325-361.

[18] Ding Xilun, Lu Shengnan and et al (2011) Configuration Transformation Theory from a Chain-type Reconfigurable Modular Mechanism-Rubik's Snake. *The 13th World Congress in Mechanism and Machine Science.*

[19] Ding Xilun and Lu Shengnan (2013) Fundamental reconfiguration theory of chain-type modular reconfigurable mechanisms. *Mechanism and Machine Theory*, **70**, 487-507.

[20] Zhang X. and Liu J. (2016) Prototype design of a rubik snake robot. *Mechanisms and Machine Science*, **36**, 581-591.

[21] Liu J. and Zhang X. and et al. (2019) Configuration analysis of a reconfigurable Rubik's snake robot. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, **233(9)**, 3137-3154.

[22] White P. J. and Revzen S. and et al. (2011) A general stiffness model for programmable matter and modular robotic structures. *Robotica*, **29(1)**, 103-121.

[23] H. Hadwiger (1950) Minkowskische Addition und Subtraktion beliebiger Punktmengen und die Theoreme von Erhard Schmidt. *Mathematische Zeitschrift*, **53(3)**, 210-218.

[24] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, **4**, 193203.

[25] Shirai Y., Osgood A. J. and et al. (2005) Directional control in thermally driven single-molecule nanocars. *Nano Letters*, **5(11)**, 2330-2334.

[26] Rapenne G. and Joachim C. (2017) The first nanocar race. *Nature Reviews Materials*, **2**.

[27] Loeve S. (2019) What's new in the world of molecular machines? The incredible adventure of nanocars. *Philosophia Scientiae*, **23(1)**, 73-98.

[28] Shirai Y., Minami K. and et al. (2016) Driving nanocars and nanomachines at interfaces: From concept of nanoarchitectonics to actual use in world wide race and hand operation. *Japanese Journal of Applied Physics*, **55(11)**.

[29] B. Bienfait and P. Ertl (2013) JSME: a free molecule editor in JavaScript. *J. Cheminformatics*, **5(24)**.

[30] Angel Herraez. (2006) Biomolecules in the computer: Jmol to the rescue. *Biochem. Mol. Biol. Educ.*, **34(4)**, 255-261.

[31] D.A. Case, I.Y. Ben-Shalom and et al. (2018) AMBER 2018. *University of California, San Francisco.*

[32] Humphrey W., Dalke A. and Schulten K. (1996) VMD - Visual Molecular Dynamics. *J. Molec. Graphics*, **14**, 33-38.

[33] J.C. Phillips, R. Braun, and et al. (2005) Scalable molecular dynamics with NAMD. *J. Comput. Chem*, **26**, 1781-1802.

[34] Sun R., Li Z. and Bishop T. C. (2019) The TMB Library: A Library of Nucleosome Simulations of DNA Sequence Effects. *J. Chem. Inf. Model.*, **59(10)**, 4289-4299.

[35] Thibault Julien C., Cheatham Thomas E. and Facelli Julio C. (2014) iBIOMES Lite: Summarizing Biomolecular Simulation Data in Limited Settings. *J. Chem. Inf. Model.*, **54**, 1810-1819.

[36] S. Jo, T. Kim, V.G. Iyer, and W. Im (2008) CHARMM-GUI: A Web-based Graphical User Interface for CHARMM. *J. Comput. Chem.* , **29**, 1859-1865.

[37] Ribeiro JV, et al. (2016) QwikMD-integrative molecular dynamics toolkit for novices and experts. *Sci Rep* , **6**, 26536.

[38] E. W. Dijkstra (1974) Programming as a discipline of mathematical nature. *Am. Math. Monthly*, **81(6)**, 608-612.

[39] Chaffer Jonathan and Swedberg Karl (2007) jQuery Reference Guide: A Comprehensive Exploration of the Popular JavaScript Library. *Packt Publishing.*

[40] Burch Carl (2010) Django, a Web Framework Using Python: Tutorial Presentation. *J. Comput. Sci. Coll.* , **25(5)**, 154-155.

[41] Andrio P., Hospital A., Conejero J. and et al. (2019) BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows. *Sci Data*, **6**, 169.

[42] Fussner E., Ching R.W. and Bazett-Jones D.P. (2011) Living without 30 nm chromatin fibers. *Trends Biochem Sci.*, **36**, 1-6.

[43] Auton A., Brooks L.D. and et al. (2015) A global reference for human genetic variation. *Nature.*, **526**, 68-74.

[44] Consortium E.P. (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature.*, **489**, 57-74.

[45] Dekker J., Belmont A.S. and et al. (2017) The 4D nucleome project. *Nature.*, **549**, 219-226.

[46] Cowper-Sallari R., Zhang X. and et al. (2012) Breast cancer risk-associated SNPs modulate the affinity of chromatin for FOXA1 and alter gene expression. *Nat. Genet.*, **44**, 1191-1198.

[47] Sadlon T., Brown C.Y. and et al. (2018) Unravelling the molecular basis for regulatory T-cell plasticity and loss of function in disease. *Clin Transl Immunology.*, **7**, e1011.

[48] Belton J.M., McCord R.P. and et al. (2012) Hi-C: a comprehensive technique to capture the conformation of genomes. *Methods.*, **58**, 268-276.

[49] Belaghzal H., Dekker J. and Gibcus J.H. 2017 Hi-C 2.0: An optimized Hi-C procedure for high-resolution genome-wide mapping of chromosome conformation. *Methods.*, **123**, 56-65.

[50] Hsieh T.S., Fudenberg G. and et al. (2016) Micro-C XL: assaying chromosome conformation from the nucleosome to the entire genome. *Nat Methods.*, **13**, 1009-1011.

[51] Dekker J., Rippe K. and et al. (2002) Capturing chromosome conformation. *Science.*, **295**, 1306-1311.

[52] Sati S. and Cavalli G. (2017) Chromosome conformation capture technologies and their impact in understanding genome function. *Chromosoma*, **126**, 33-44.

[53] Duim W.C., Jiang Y. and et al. (2014) Super-resolution fluorescence of huntingtin reveals growth of globular species into short fibers and coexistence of distinct aggregates. *ACS Chem. Biol.*, **9**, 2767-2778.

[54] Ricci M.A., Cosma M.P. and Lakadamyali M. (2017) Super resolution imaging of chromatin in pluripotency, differentiation, and reprogramming. *Current opinion in genetics & development*, **46**, 186-193.

[55] Wilson M.D. and Costa A. (2017) Cryo-electron microscopy of chromatin biology. *Acta crystallographica. Section D, Structural biology*, **73**, 541-548.

[56] Mlynárik V. (2017) Introduction to nuclear magnetic resonance. *Anal. Biochem.*, **529**, 4-9.

[57] Tan S. and Davey C.A. (2011) Nucleosome structural studies. *Curr. Opin. Struct. Biol.*, **21**, 128-136.

[58] Schlick T., Hayes J. and Grigoryev S. (2012) Toward convergence of experimental studies and theoretical modeling of the chromatin fiber. *J Biol Chem.*, **287**, 5183-5191.

[59] Perišić O. and Schlick T. (2016) Computational strategies to address chromatin structure problems. *Phys Biol.*, **13**, 035006.

[60] Portillo-Ledesma S. and Schlick T. (2019) Bridging chromatin structure and function over a range of experimental spatial and temporal scales by molecular modeling. *WIREs Comput. Mol. Sci.*, e1434.

[61] Ozer G., Luque A. and Schlick T. (2015) The chromatin fiber: multiscale problems and approaches. *Curr Opin Struct Biol.*, **31**, 124-39.

[62] Perkel J.M. (2017) Plot a course through the genome. *Nature.*, **549**, 117-118.

[63] 4DN Software, available at `https://www.4dnucleome.org/software.html/`

[64] Stolz R.C. and Bishop T.C. (2010) ICM web: the interactive chromatin modeling web server. *Nucleic Acids Res.*, **38**, W254-61.

[65] Woodcock C.L., Grigoryev S.A. and et al. (2015) A chromatin folding model that incorporates linker variability generates fibers resembling the native structures. *Proc. Natl. Acad. Sci. U.S.A.*, **90**, 9021-9025.

[66] Simo J.C. and Vu-Quoc L. (1991) A geometrically-exact rod model incorporating shear and torsion-warping deformation. *Int J Solids Struct.*, **27**, 371-393.

[67] Calladine C.R. and Drew H.R. (1986) Principles of sequence-dependent flexure of DNA. *J Mol Biol.*, **192**, 907-918.

[68] Fathizadeh A., Eslami-Mossallam B. and Ejtehadi M.R. (2012) Definition of the persistence length in the coarse-grained models of DNA elasticity. *Phys Rev E Stat Nonlin Soft Matter Phys.*, **86**, 051907.

[69] Olson W.K., Bansal M. and et al. (2001) A standard reference frame for the description of nucleic acid base-pair geometry. *J Mol Biol.*, **313**, 229-37.

[70] Olson W.K., Gorin A.A. and et al. (1998) DNA sequence-dependent deformability deduced from protein-DNA crystal complexes. *Proc Natl Acad Sci U S A.*, **95**, 11163-11168.

[71] Lavery R., Zakrzewska K. and et al. (2010) A systematic molecular dynamics study of nearest-neighbor effects on base pair and base pair step conformations and fluctuations in b-dna. *Nucleic Acids Res.*, **38**, 299-313.

[72] Pasi M., Maddocks H. and et al. (2014) μABC: a systematic microsecond molecular dynamics study of tetranucleotide sequence effects in B-DNA. *Nucleic Acids Res.*, **42**, 12272-83.

[73] Diekmann S. (1989) Definitions and nomenclature of nucleic acid structure parameters. *J Mol Biol.*, **205(4)**, 787-791.

[74] Petkevičiūtė,D., Pasi,M., Gonzalez,O. and Maddocks,J.H. (2014) cgDNA: a software package for the prediction of sequence-dependent coarse-grain free energies of B-form DNA. *Nucleic Acids Res.*, **42**,e153.

[75] Bishop T.C. (2009) VDNA: the virtual DNA plug-in for VMD. *Bioinformatics*, **25**, 3187-3188.

[76] Bishop T.C. and Hearst J.E. (1998) Potential Function Describing the Folding of the 30 nm Fiber. *The Journal of Physical Chemistry B*, **102(33)**, 6433-6439.

[77] Alexander S.R. and Peter W.H. (2015) NGL Viewer: a web application for molecular visualization. *Nucleic Acids Res.*, **43(W1)**, W576-W579.

[78] Schrodinger LLC. (2015) The PyMOL Molecular Graphics System, Version 1.8

[79] Down T.A., Matias P. and Tim J.P.H. (2011) Dalliance: Interactive Genome Viewing on the Web. *Bioinformatics*, **27(6)**, 889-890.

[80] Skinner M.E., Uzilov A.V. and et al. (2009) JBrowse: A next-generation genome browser *Genome Res.*, **19(9)**, 1630-1638.

[81] Anthony W. N. and Gerald L. (1997) *Hormones.* Academic Press.

[82] Andrew P.A. and Adam G.J. (2019) erefinder: Genome-wide detection of oestrogen response elements. *Mol Ecol Resour.*,

[83] Yongbing Z., Jinyue W. and et al. (2018) NucMap: a database of genome-wide nucleosome positioning map across species. *Nucleic Acids Res.*, **47(D1)**, D163-D169.

[84] Bascom G.D., Myers C.G. and Schlick T. (2019) Mesoscale modeling reveals formation of an epigenetically driven HOXC gene hub. *Proc Natl Acad Sci U S A.*, **116(11)**, 4955-4962.

[85] Zhang Q., Beard D.A. and Schlick T. (2003) Constructing irregular surfaces to enclose macromolecular complexes for mesoscale modeling using the discrete surface charge optimization (DISCO) algorithm. *J Comput Chem.*, **24**, 2063-2074.

[86] Helmut S., William M. G. and Robijn B. (2001) DNA Folding: Structural and Mechanical Properties of the Two-Angle Model for Chromatin. *Biophys. J.*, **80(4)**, 1940-1956.

[87] Smith J.A., Romanus M. and et al. (2013) Scalable online comparative genomics of mononucleosomes: a BigJob. *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, **13**, 23:1–23:8.

[88] Bishop T.C., Kosztin D. and Schulten K. (1997) How hormone receptor-DNA binding affects nucleosomal DNA: the role of symmetry. *Biophys J.*, **72**, 2056-2067.

[89] Kent W J, Zweig A S and et al. (2010) BigWig and BigBed: enabling browsing of large distributed datasets. *Biophys J.*, **26**, 2204-2207.

[90] Davey Curt A, Sargent David F and et al. (2002) Solvent mediated interactions in the structure of the nucleosome core particle at 1.9 a resolution. *J. Mol. Biol.*, **319(5)**, 10971113.

[91] Plimpton S. (1995) Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.*, **117**, 1-19.

[92] Nikolay Korolev, Alexander P. Lyubartsev and Lars Nordenskiold (2010) Cation-induced polyelectrolyte–polyelectrolyte attraction in solutions of DNA and nucleosome core particles. *Adv. Colloid Interface Sci.*, **158(1-2)**, 32-47.

[93] Nikolay Korolev, Yongqian Zhao and et al (2012) The effect of salt on oligocation-induced chromatin condensation. *Biochem. Biophys. Res. Commun.*, **418(2)**, 205-210.

[94] Fuller F B (1971) The writhing number of a space curve. *Proc. Natl. Acad. Sci. U.S.A.*, **68**, 815-819.

[95] White J H and Bauer W R (1986) Calculation of the twist and the writhe for representative models of DNA. *J. Mol. Biol.*, **189**, 329-341.