



**HAL**  
open science

# Conception de nanostructures ADN par des méthodes géométriques : Modèles, logiciel et expériences

Nicolas Levy

► **To cite this version:**

Nicolas Levy. Conception de nanostructures ADN par des méthodes géométriques : Modèles, logiciel et expériences. Modeling and Simulation. Ecole normale supérieure de lyon - ENS LYON, 2023. English. NNT : 2023ENSL0025 . tel-04196229

**HAL Id: tel-04196229**

**<https://theses.hal.science/tel-04196229>**

Submitted on 5 Sep 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Numéro National de Thèse : 2023ENSL0025

## THÈSE

en vue de l'obtention du grade de Docteur, délivré par  
l'ÉCOLE NORMALE SUPÉRIEURE DE LYON

**École Doctorale N° 512**  
École Doctorale en Informatique et Mathématiques de Lyon

**Discipline** : Informatique

Soutenue publiquement le 22/06/2023, par :

**Nicolas Levy**

---

### **Geometry-driven design of DNA nanostructures : Models, software and experiments**

**Conception de nanostructures ADN par des méthodes géométriques : Modèles,  
logiciel et expériences.**

---

Devant le jury composé de :

DIETZ, Hendrik	Professeur, TUM Munich	Rapporteur
SIEGEL, Anne	Directrice de Recherche, Université de Rennes1	Rapporteuse
GALAS, Jean-Christophe	Chargé de Recherche HDR, UPMC	Examineur
PONTY, Yann	Directeur de Recherche, École Polytechnique	Président
QIAN, Lulu	Professeure, Caltech	Examinatrice
SCHABANEL, Nicolas	Directeur de Recherche, ENS de Lyon	Directeur de thèse





*Dans l'univers soudain rendu à son silence, les milles petites voix émerveillées de la terre s'élèvent. Appels inconscients et secrets, invitations de tous les visages, ils sont l'envers nécessaire et le prix de la victoire. Il n'y a pas de soleil sans ombre, et il faut connaître la nuit. L'homme absurde dit oui et son effort n'aura plus cesse.*

Albert Camus, *Le mythe de Sisyphe*



# Remerciements/Acknowledgements

I was first introduced to DNA nanotechnology and molecular computing by Tristan, a former classmate of mine who presented me the implementation by Woods, Doty, *et al.* of 6-bits algorithms with DNA self-assembly. Later, I was given the chance to work in Damien Wood's group, under the supervision of Pierre-Étienne Meunier. I would like to express my gratitude to all the people thanks to whom my stay in Ireland was such a great experience.

My adventures into the world of nucleic acids continued under the supervision of Nicolas Schabanel, who supervised me during my last internship and then during my whole PhD. Thank you, Nicolas, for your trust, your support, and your patience. Thank you also for having produced some of the experimental results that are presented in this manuscript. It is certainly not often that a supervisor runs the experiments that accompany their student's work.

I am also extremely grateful to Gaëtan, Allan and especially Julie for their tremendous work and efforts that allowed me to illustrate the last chapters of this manuscript with beautiful images of DNA origami.

Finally, I would like to thank Hendrik Dietz and Anne Siegel for taking the time to review this manuscript and for their instructive feedbacks; as well as Jean-Christophe Galas, Yann Ponty, and Lulu Qian for agreeing to take part to the evaluation of my thesis as members of the jury.

Si j'ai eu la chance de pouvoir compter sur le soutien de nombreuses personnes pour mon développement en tant que scientifique. Je souhaite également remercier toutes celles et ceux qui ont contribué d'une manière plus personnelle à mon épanouissement pendant mes années d'études. À ces personnes-là, je souhaite formuler mes remerciements en français, langue qui me permet naturellement de m'exprimer de manière plus intime.

Mon année la plus marquante à l'ENS fut celle où j'ai découvert que les cellules animales en culture ne s'organisaient pas naturellement en un joli maillage ; que c'est trop la honte de ne pas connaître la méthode de Sanger ; et que l'on pouvait tenir pour vraie toute hypothèse dont il a été établi que les alternatives ont moins de 5% de crédibilité. Au cours de cette année j'ai rencontré de nombreuses personnes formidables que j'ai l'immense privilège de pouvoir toujours compter parmi mes amis.

Merci, Paul, de m'avoir démontré que l'on pouvait apprécier n'importe quelle plante potagère, pourvu que celle-ci soit accompagnée d'une dose suffisante de crème fraîche. Merci, Alexandra, de m'avoir ouvert les yeux sur le scandale culinaire que constituerait la consommation de certains fruits lors d'une fondue au chocolat. Merci, Armelle, de m'avoir expliqué que les photos de vacances paraissent plus authentiques lorsque l'on ne peut pas y voir de câbles électriques ou autres remontées mécaniques. Merci enfin, Nicolas, de t'être

débarrassé de ces dangereux mini-Mu et de la menace infectieuse qu'ils représentaient pour nos cultures de bactéries.

Je voudrais remercier ensuite toutes les personnes qui ont fait de mes années de master et de thèses des années extrêmement riches.

Tout d'abord, merci Nicolas pour le travail que nous avons accompli ensemble pendant ces trois ans. La thèse n'est pas un long fleuve tranquille, et nos styles de navigation sont loin d'être identiques, mais je suis heureux que nous ayons réussi à nous accorder pour amener la barque à bon port.

Merci Pierre pour le temps que nous avons passé ensemble à arpenter les salles d'escalade et les restaurants de Lyon. J'espère que tu continueras à t'inspirer des civilisations qui ont précédé la notre et, en particulier, à ne pas faire de suppositions 🐼

Merci au groupe des gones d'avoir rythmés nos années de thèse. Merci Adèle et Octave pour le rôle important que vous avez joué dans l'organisation des sorties en dehors de Lyon. Merci Valentine d'avoir supervisé la réalisation des nombreux desserts que je me réjouissais de déguster lors des brunchs et soirées qui ont fait vivre notre groupe. Merci à toutes et tous, en particulier à Alice et Nicolas pour votre précieux soutien pendant les moments difficiles ayant marqués la rédaction de ce manuscrit.

Merci enfin, Justine, pour le bout de chemin que nous avons parcouru ensemble et au long duquel j'espère m'être imprégné d'une fraction de ta bienveillance et de ta sollicitude.

Le dernier paragraphe de ces remerciements s'adresse à mes parents et à mes sœurs. Je le rédige depuis le bout de la Pointe de la Garde peu de temps après avoir terminé le corps de ce manuscrit. Je sais que toute ma vie, cet endroit me rappellera que j'ai eu la chance de grandir dans une famille formidable et grâce à laquelle j'ai eu une enfance heureuse.

Tout au long de mon voyage académique, j'ai pu m'appuyer sur vos encouragements et sur votre présence. Votre confiance en moi et votre soutien inconditionnel m'ont permis de trouver le courage d'explorer des domaines scientifiques bien plus vastes que ceux auxquels j'aurais pu croire être prédestiné. Ce document n'existerait pas sans tout cela, merci.





# Abstract

DNA origami is a technique used to build bio-compatible structures at the nanometer scale with remarkable ease and precision. The technique consists in letting self-assemble a carefully designed network of DNA strands.

The key step in DNA origami design is the positioning of crossovers, which constrains the structure geometrically to adopt the desired shape. This important task is made difficult by the peculiar geometry of the DNA double helix. Software and empirical patterns have been proposed to assist the designer, yielding to increasingly complex structures. However, the limited precision of these rules forces the DNA origami design process to involve several passes of validation using simulation software, as well as several experimental trials of assembly and characterization (by atomic force or electron microscopy) before obtaining the desired shape with a satisfying precision and yield.

In this work, we lead a reflection on what constitutes a good software for designing DNA origami. Our goal is to provide an edition interface that allows crossovers locations to be directly deduced from the geometry of the intended shape. To that end, we develop a model of DNA nanostructures that is simple enough to be manipulated through intuitive interfaces, without compromising on precision and fine-tuning. We also develop a new geometrical model for curved DNA helices allowing a new nature-inspired spiral-based method for routing helices along curved surfaces. We have implemented all these new methods in our new software, ENSnano.

We provide experimental evidence of their efficiency by designing and accurately assembling origami with unprecedentedly complex 3D curvatures.





# Résumé en Français

La technique de l'origami ADN est utilisée pour construire des structures bio-compatibles avec une facilité et une précision remarquables à l'échelle du nanomètre. La technique consiste à faire s'auto-assembler un ensemble de brins d'ADN dont les séquences sont soigneusement choisies dans ce but.

La difficulté principale de la technique est le positionnement des crossing-overs qui imposent à la structure des contraintes géométriques dont résulte sa forme. Cette tâche essentielle est compliquée par la géométrie singulière de la double hélice d'ADN. Des logiciels facilitant le travail de conception et des règles empiriques ont permis la production d'objets de plus en plus complexes. Cependant, du fait de la précision limitée de ces outils, la conception d'un origami ADN implique le plus souvent plusieurs passes de validation par des logiciels de simulations physiques, ainsi que plusieurs essais d'assemblage et de caractérisation (par microscopie à force atomique ou par microscopie électronique) avant d'obtenir la forme souhaitée avec une précision et un rendement satisfaisants.

Dans cette thèse, nous menons une réflexion sur ce qui constitue un bon logiciel pour la conception d'origamis ADN. Notre objectif est de proposer une interface permettant de déduire la position des crossing-overs directement de la géométrie de la forme souhaitée. Nous développons un modèle des nanostructures ADN qui est suffisamment simple pour pouvoir être manipulé via des interfaces utilisateur intuitives, sans compromis sur la précision nécessaire pour affiner les designs. Nous développons en particulier un nouveau modèle géométrique pour les hélices d'ADN courbes. Ce modèle nous a permis de développer une nouvelle méthode, inspirée de structures naturelles, pour couvrir, avec des hélices d'ADN, des surfaces courbes auparavant inaccessibles. Nous avons implémenté toutes ces méthodes dans notre nouveau logiciel : ENSnano.

Nous démontrons expérimentalement l'efficacité de nos méthodes en concevant et en assemblant des origamis ADN présentant des courbures tridimensionnelles d'une complexité sans précédent.



# Résumé long en Français

L'ADN est la molécule biologique qui sert de support de l'information génétique des êtres vivants. Sa structure moléculaire consiste en deux brins enroulés l'un autour de l'autre pour former une double hélice. Ces brins sont une succession de monomères appelés nucléotides et contenant chacun une base azotée. La cohésion des deux brins de la double hélice est permise par l'existence d'une affinité chimique entre les bases dites complémentaires. La double hélice d'ADN peut donc aussi être vue comme une succession de paires de bases.

En 1982, Nadrian Seeman a introduit l'idée d'utiliser cette affinité chimique entre bases complémentaires pour créer des structures artificielles faites de brins d'ADN. Dans ces structures, chaque brin est spécialement conçu pour s'appareiller à plusieurs autres et former ainsi un réseau cristallin. L'applicabilité de l'idée de Seeman fut démontrée expérimentalement quelques années plus tard ce qui marqua la fondation d'un nouveau champs de recherche scientifique appelé nanotechnologie ADN.

En 2006, Paul Rothemund publia une méthode appelée "Origami ADN" permettant de créer des nanostructures ADN à partir d'une longue molécule circulaire d'ADN viral et de plusieurs petits brins synthétiques. Du fait de sa robustesse et la simplicité de sa mise en place expérimentale, cette technique est devenue centrale dans les nanotechnologies ADN.

L'étape cruciale dans la conception d'un origami ADN est le choix des positions auxquelles les brins d'ADN passent d'une hélice à une autre. Ces changements d'hélices, appelés *crossing-overs*, sont la source des contraintes géométriques imposées à la structure et dont résulte sa forme. Le positionnement des crossing-overs est donc une tâche essentielle. Cette tâche est cependant rendue difficile par la géométrie particulière de la double-hélice d'ADN. En effet, les crossing-overs doivent être positionnés de sorte à relier deux nucléotides suffisamment proches sur chacune des deux hélices impliquées. Les liaisons covalentes entre deux nucléotides étant rigides, essayer de relier deux nucléotides distants provoquerait des contraintes physiques non désirables pouvant diminuer les chances que la structure s'assemble avec la forme voulue.

Pour assister les concepteurs et conceptrices d'origami ADN dans le positionnement des crossing-overs, il existe des règles basées sur des approximations de la géométrie de la double hélice, ainsi que des logiciels de conception assistée par ordinateurs (CAO). Avec le temps, ces outils ont permis la réalisation d'objets de plus en plus complexes. Toutefois, du fait de leur précision limitée, la conception d'un origami ADN implique généralement plusieurs passes de validations par des logiciels de simulations physiques, ainsi que plusieurs tentatives d'assemblage et de caractérisation (par microscopie à force atomique ou par microscopie électronique) avant d'obtenir la forme souhaitée avec une précision et un rendement satisfaisant.

Dans cette thèse, nous menons une réflexion sur ce qui constitue un bon logiciel pour la conception d'origamis ADN, et nous proposons une méthode de conception dans laquelle

le positionnement des crossing-overs se fait directement à partir d'un modèle géométrique des doubles hélices d'ADN. Notre modèle s'applique également aux doubles-hélices d'ADN courbes à l'aide de nouvelles méthodes que nous introduisons dans ce document. Le développement mathématique de ces méthodes s'accompagne d'une implémentation dans notre logiciel ENSnano, ainsi que de validations expérimentales démontrant l'efficacité de notre approche.

La première partie de cette thèse constitue une introduction aux nanotechnologies ADN.

Après avoir introduit quelques notions fondamentales de biologie moléculaire, nous présentons l'histoire des nanotechnologies ADN ainsi que certaines de leurs applications.

Nous présentons ensuite en détails la technique de l'Origami ADN ainsi que les méthodes de conceptions les plus fréquemment utilisées.

Enfin, nous nous penchons sur l'état de l'art en matière de logiciels de CAO pour les nanotechnologies ADN. Sur la base de cet état de l'art, nous établissons nos besoins en terme d'interface utilisateur et de modèle géométrique pour notre logiciel ENSnano.

La deuxième partie de ce document établit les fondations de notre logiciel.

Nous nous penchons d'abord sur les structures de données existant dans d'autres logiciels afin de déterminer une structure de donnée adaptée à la représentation de designs de nanostructures ADN. Nous proposons alors une structure hiérarchique de ces objets, basées sur la notion de grilles pour positionner dans l'espace les hélices portant la structure. À l'instar de logiciels pré-existants, nous utilisons une structure de donnée dans laquelle la topologie du design est représentée séparément de sa forme 3D. La structure que nous proposons se démarque en permettant de représenter facilement les différents composants d'une structure et de les positionner librement dans l'espace. Nous présentons ensuite comment déduire de notre structure de donnée les positions dans l'espace des nucléotides le long des doubles hélices droites.

La connaissance des positions dans l'espace des nucléotides nous permet de développer plusieurs moteurs physiques pouvant être intégrés directement au logiciel. Nous proposons ainsi un outil permettant d'ajuster automatiquement la rotation des hélices autour de leur axe afin de minimiser les distances entre nucléotides reliés par des crossing-overs. Nous développons également un moteur de simulation de physique du solide que nous appliquons aux doubles hélices du design afin de tester sa stabilité. Le but de ce moteur n'est pas de déduire la forme de la structure à partir de sa topologie, mais de vérifier si la forme telle que représentée dans le logiciel a de bonnes chances d'être obtenue expérimentalement ou non. Enfin, nous offrons un troisième outil, également basé sur des simulations de physique du solide, permettant de positionner dans l'espace les différents composants de designs réalisés dans des logiciels n'incorporant pas d'informations sur la géométrie dans leur format de fichier.

Nous terminons la deuxième partie par une présentation de la première version publique de notre logiciel. Une particularité d'ENSnano est d'utiliser simultanément deux interfaces pour visualiser et éditer le design. ENSnano offre ainsi une vue 3D dans laquelle il est possible d'observer et d'interagir avec l'organisation spatiale des éléments de la structure, ainsi qu'une interface 2D inspirée de celle du logiciel cadnano et permettant d'interagir avec le design au travers d'une représentation abstraite de sa topologie. Cette interface 2D "à la cadnano" est, en général, la plus ergonomique pour les opérations d'éditions les plus courantes. L'interface 3D d'ENSnano est quant à elle, particulièrement utile pour certaines opérations spécifiques

(comme la création d'un crossing-over entre deux hélices non parallèles) pour lesquelles une visualisation en 3D est indispensable. Nous terminons la présentation de notre logiciel par une démonstration de son efficacité en concevant dans ENSnano puis en assemblant expérimentalement avec succès un origami ADN fait de deux couches d'hélices non parallèles. La facilité et la précision avec laquelle un tel design peut être conçu dans ENSnano permet à notre logiciel de se démarquer des autres solutions existantes.

Dans la troisième et dernière partie, nous développons des méthodes permettant de concevoir dans ENSnano des origamis avec des formes courbes.

Nous commençons par présenter les polynômes de Tchebychev. Ces objets mathématiques fréquemment utilisés en analyse numérique nous permettent d'interpoler, représenter et stocker efficacement des chemins courbes.

Nous présentons ensuite notre nouveau modèle géométrique pour les doubles-hélices d'ADN courbes. Nous développons un algorithme pour équiper des lignes courbes d'un repère "paresseusement adaptatif", que nous utilisons pour générer les positions dans l'espace des nucléotides d'une double-hélice d'ADN dont l'axe suit une trajectoire courbe arbitraire en 3D.

Nous nous servons ensuite de ce modèle pour concevoir un origami ADN fait de six hélices suivant une courbe de Bézier fermée et décrivant un carré avec des angles bouclés ☞.

Nous développons enfin une nouvelle méthode pour recouvrir avec des hélices d'ADN des surfaces de révolution torsadées. L'innovation de cette méthode est d'utiliser des chemins en forme de spirales ce qui permet une meilleure flexibilité que les méthodes précédentes basées sur des anneaux concentriques. À l'aide de cette nouvelle technique nous assemblons une série d'origamis ADN présentant des formes d'une complexité sans précédent : des tores torsadées, dont la section elliptique effectue plusieurs rotations sur elle-même au cours d'une révolution ; ainsi que deux designs chacun réalisant deux sphères concentriques reliées par deux tunnels dont le rayon est plus petit que le plus petit rayon de courbure jamais observé dans la nature pour une double hélice d'ADN.



## Table of symbols

Symbol	Unit/Space	Meaning
$\mathbf{a}$		$\mathbf{a}$ is a vector (lowercase)
$\mathbf{A}$		$\mathbf{A}$ is a matrix (uppercase)
$\mathbf{a}^T$		Transpose of vector $\mathbf{a}$
$\bar{x}$		$\bar{x}$ is an Approximation of an integral by a Riemann sum
$\tilde{f}$	$\mathbb{R} \rightarrow \mathbb{R}$ or $\mathbb{R}$	$\tilde{f}$ is a Chebyshev interpolation of $f$ , or is a value that was computed using a Chebyshev interpolation
$f^{\leftrightarrow}$	$\mathbb{R} \rightarrow f(\mathbb{R})$	A function obtained by performing a change of variable on $f$
$\dot{f}$		The derivative of $f$ with respect to time $\frac{df}{dt}$
$\Re(z)$	$\mathbb{R}$	Real part of the complex number $z$
$\ \cdot\ _2$		Euclidean norm
$[x]$	$\mathbb{Z}$	$x$ rounded to the nearest integer
$\lfloor x \rfloor$	$\mathbb{Z}$	Largest integer $\leq x$
$\{x\}$	$[0, 1[$	fractional part of $x$ : $x - \lfloor x \rfloor$
$\mathbb{R}_n[X]$		The set of polynomials with real coefficients and degree $\leq n$
$\text{SO}_3(\mathbb{R})$		The set of right-handed, orthonormal real $3 \times 3$ matrices
$\mathcal{M}_n(\mathbb{R})$		The set of real $n \times n$ matrices
$\mathcal{O}$	$\mathbb{R}^3$	Origin of the axis of a helix/ Origin of a grid
$\mathcal{F}$	$\text{SO}_3(\mathbb{R})$	Frame associated to a helix/grid
$\mathbf{Q}_{\mathbf{a}}(\theta)$	$\text{SO}_3(\mathbb{R})$	Matrix associated to the rotation of angle $\theta$ around axis $\mathbf{a}$ .
$\Gamma$	$\mathbb{Z}^2 \rightarrow \mathbb{R}^2$	The lattice associated to a grid
$\Delta$	nanometers	Rise of a DNA helix
$\alpha$	radians	Twist of a DNA helix
$\rho$	radians	Roll of a DNA helix
$\mathcal{H}$	nanometers	Inter-helix-axis distance

## Table of abbreviations

AFM	Atomic Force Microscope
TEM	Transmission Electron Microscopy
TAE	Buffer solution with <u>T</u> ris, <u>A</u> cetic acid and <u>E</u> DTA
Mg	Magnesium
Ni	Nickel
mM/ $\mu$ M/nM	millimole/micromole/nanomole per liter
min	minute
bp	base pair
ODE	Ordinary Differential Equation
SST	Single-Stranded Tiles





# Contents

<b>Remerciements/Acknowledgments</b>	<b>i</b>
<b>Abstract</b>	<b>v</b>
<b>Résumé en Français</b>	<b>vii</b>
<b>Résumé long en Français</b>	<b>ix</b>
<b>I Introduction</b>	<b>1</b>
<b>1 DNA and DNA nanotechnologies</b>	<b>3</b>
1.1 Molecular structure of DNA . . . . .	3
1.2 DNA nanotechnologies . . . . .	8
1.3 Applications of DNA nanotechnologies . . . . .	13
<b>2 DNA origami</b>	<b>17</b>
2.1 General Presentation of the technique . . . . .	17
2.2 Design and shapes of DNA origami . . . . .	20
<b>3 Computer assisted design of DNA nanostructures</b>	<b>25</b>
3.1 State of the art . . . . .	25
3.2 Objectives of our work . . . . .	29
<b>II Sane foundations for a DNA nanostructure design software</b>	<b>31</b>
<b>4 Data structure and geometric model for DNA nano-objects</b>	<b>33</b>
4.1 Example of existing data-structures . . . . .	33
4.2 ENSnano’s data structure. . . . .	35
4.3 P-stick model for straight DNA helices . . . . .	36
<b>5 Physical model for automatic roll, stability testing and design import</b>	<b>41</b>
5.1 Motivations and scope . . . . .	41
5.2 Automatic roll optimization . . . . .	42
5.3 Simulating rigid body dynamics . . . . .	44
5.4 Rigid body simulations in ENSnano . . . . .	46

<b>6</b>	<b>An efficient user interface for the conception of DNA nanostructures</b>	<b>51</b>
6.1	Overview of ENSnano’s interface . . . . .	51
6.2	The main 2D and 3D views . . . . .	53
6.3	Geometry-based features . . . . .	60
6.4	Experimental validation: rocket origami with two non-parallel layers . . . . .	63
<b>III</b>	<b>Design of curved DNA nanostructures</b>	<b>65</b>
<b>7</b>	<b>Chebyshev Polynomials</b>	<b>67</b>
7.1	Mathematical properties . . . . .	67
7.2	Implementation details . . . . .	71
<b>8</b>	<b>A model for curved DNA double helices</b>	<b>75</b>
8.1	P-stick model for curved DNA helices . . . . .	75
8.2	Discretization of the curve . . . . .	76
8.3	Construction of a moving right-handed orthonormal frame along a curved path	78
8.4	Implementation of the curved P-stick model and <i>in silico</i> validation . . . . .	80
<b>9</b>	<b>Design of curved bundle of helices</b>	<b>85</b>
9.1	Design principles . . . . .	85
9.2	Design of curved bundles in ENSnano . . . . .	87
9.3	Experimental validation . . . . .	92
<b>10</b>	<b>Routing spiraling helices around shapes with 3D curvature</b>	<b>101</b>
10.1	Motivations and overview of the pipeline . . . . .	102
10.2	Initial Routing . . . . .	107
10.3	Balancing the inter-helix distances . . . . .	109
10.4	Adjustments of the revolution radius to the desired scaffold length . . . . .	116
<b>11</b>	<b>Experimental validations: design and assembly of origami with complex 3D curvatures</b>	<b>119</b>
11.1	Design of twisted revolution surfaces in ENSnano . . . . .	119
11.2	Experimental validations . . . . .	124
<b>Conclusion and perspectives</b>		<b>139</b>
	Summary of contributions . . . . .	139
	Future work . . . . .	140

**Part I**

**Introduction**



# Chapter 1

## DNA and DNA nanotechnologies

### 1.1 Molecular structure of DNA

It is known since Frederick Griffith's experiment in 1928 [Gri28] that DNA is the molecular support of genetic information. The molecular structure of DNA was however discovered much later, in 1953. That year, the iconic X-ray photograph known as "Photo 51" (Figure 1.1) taken by Raymond Gosling as he was working under the direction of Rosalind Franklin [FG53], allowed James Watson and Francis Crick to publish the first ever correct modeling of the structure of B-form DNA [WC53].

**DNA strands** are polymers made of repeating monomeric units called *nucleotides*. Each nucleotide contains a nitrogenous base called *nucleobase* or simply *base* linked to a sugar (more precisely a *deoxyribose*) and a phosphate group. Together, the sugar and phosphate group form the segment of the strand's *backbone* (Figure 1.2).

On the backbone, the third carbon atom of the sugar of a nucleotide connects to the phosphate group of the next nucleotide, and the phosphate group of a nucleotide connects to the fifth carbon atom of its own sugar.

In non-cyclic strands, one end of the strand, called the *5' end of the strand*, is terminated by the phosphate group of the nucleotide, while the other, called the *3' end of the strand*,

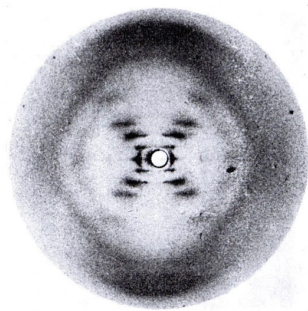


Figure 1.1: **X-ray diagram of a salt of B-form of DNA as published by Franklin and Gosling in [FG53].** This iconic image known as "Photo 51" was a key element leading to the characterization of the structure of B-form DNA.

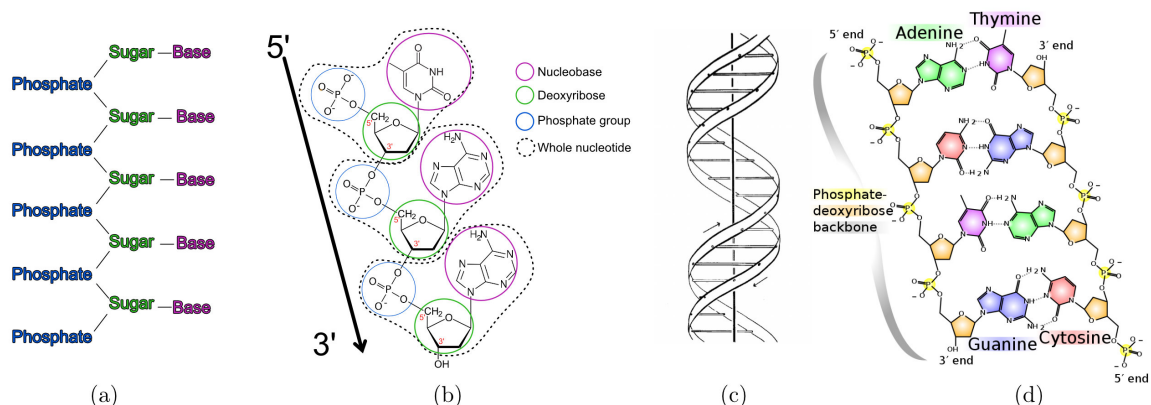


Figure 1.2: **Structure of the DNA molecule.** (a): Schematic representation of the structure of a DNA strand, adapted from [WC53]. (b): The molecular structure of a DNA strand. (c): Drawing of a DNA double helix made of two hybridized strands, as published in [WC53] (d): Molecular structure of a DNA double helix, showing two reverse complementary strands and the hydrogen bonds (dotted lines) between them. Panel extracted from Wikipedia [https://commons.wikimedia.org/wiki/File:DNA\\_chemical\\_structure-1-.fr.svg](https://commons.wikimedia.org/wiki/File:DNA_chemical_structure-1-.fr.svg).

is terminated by a hydroxyl group connected to the third carbon of the nucleotide's sugar. A consequence is that the two ends of DNA strands are distinguishable which means that strands are oriented. By convention, the orientation of a strand goes from the 5' end to the 3' end, which corresponds to the order in which strands are assembled by DNA polymerases in living organisms.

**Base pairs.** The nitrogenous base of a nucleotide can either be Adenine (A), Thymine (T), Cytosine (C) or Guanine (G). There exists chemical affinities between Adenine and Thymine, and between Cytosine or Guanine, that allow bases to bind to each other by hydrogen bonds, forming what is called a “Watson-Crick” base pair.

In its most common natural form, the DNA molecule is made of two strands that bind to each other, forming a double helix with a radius of 1 nm, a rise of 0.334 nm per base pair and making a full turn every 10.44 base pairs on average. The two strands of the helix go in opposite directions. We say that the two strands of the helix are the *reverse complement* of one another (Figure 1.2d).

It is important to note that in B-form DNA, the two strands are not diametrically opposed. For this reason the spaces between the two strands, called *grooves*, are not equally wide. The ratio between the width of the major and the minor groove (measured along the axis of the helix) is almost exactly 7:4 [WDT<sup>+</sup>80], meaning that the major groove is 2.2 nm wide while the minor groove is 1.2 nm wide (Figure 1.3b).

In addition to hydrogen bonds, the stability of the DNA double helix is ensured by *stacking interactions* between consecutive base. These stacking interactions also account for the greater rigidity of the double helix [Por91] compared to single-stranded DNA.

**Melting and hybridization of DNA strands.** In a hot enough medium, the two strands of the double helix will separate in a process called *denaturation* or *melting* of DNA. This is

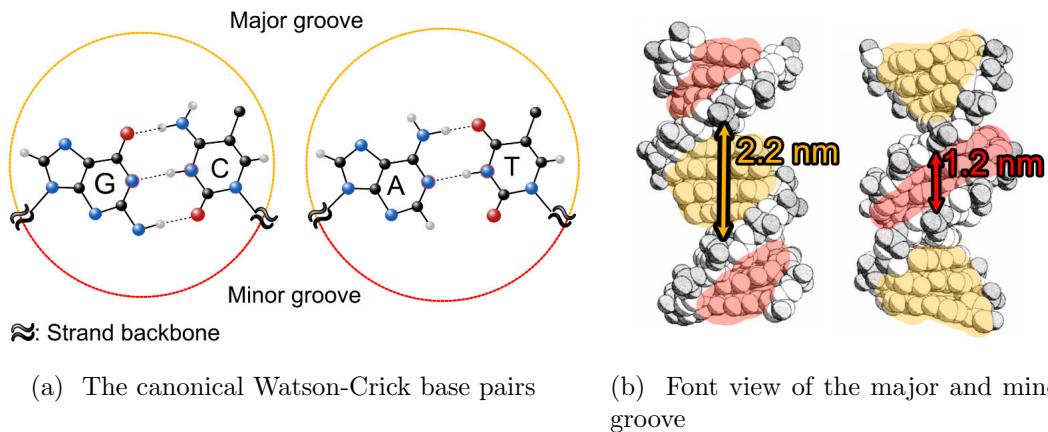


Figure 1.3: **The major and minor grooves of the DNA double helix.** a: The two Watson-Crick base pairs. Notice that the backbone of the nucleotides are not diametrically opposed. b: Space filling representations of the DNA double helix from a point of view facing the major (Left) or minor (Right) groove. The bases are highlighted in orange in the major groove and in red in the minor groove. Panel b is adapted from a figure in [WDT<sup>+</sup>80].

because at high temperature, the chemical energy of the hydrogen bonds do not compensate for the entropy gained by separating the two strands. The temperature at which the strands separate depends on salt concentration in the solution [SL65]. At low salt concentrations, the melting temperature of DNA is around 65°C. Conversely, if two complementary single strands of DNA are present in solution at a low enough temperature, they will spontaneously hybridize to form a double helix.

**Gel electrophoresis.** Because of the presence of phosphate groups on the backbone, the DNA molecule is negatively charged in non-acidic solutions. As a consequence, DNA molecules will move when submitted to an electric field. This can be used to sort DNA strands in a process called *gel electrophoresis*.

The DNA fragments to be sorted are placed in well on one side of an agarose gel and an electric field is run through the gel that will make the fragments migrate through the gel. DNA can then be localized on the gel by using a fluorescent molecule that binds to it. Placing the gel under a UV light will then reveal the positions of the DNA fragments. When the strands present in the solution can be sorted in groups of distinct sizes, clear separated “bands” will appear on the gel (Figure 1.4).

The speed at which the fragments move along the gel is usually strongly correlated to their lengths and the process can be used to discriminate the strands by size. Gel electrophoresis can for example be used to test the presence or absence in a solution of a DNA fragment with a known length. In that case a *molecular ladder* is used. The molecular ladder is a solution containing a range of DNA fragments of known lengths. By comparing the positions of the fragments in the tested solution with those of the molecular scale, their lengths can be inferred.



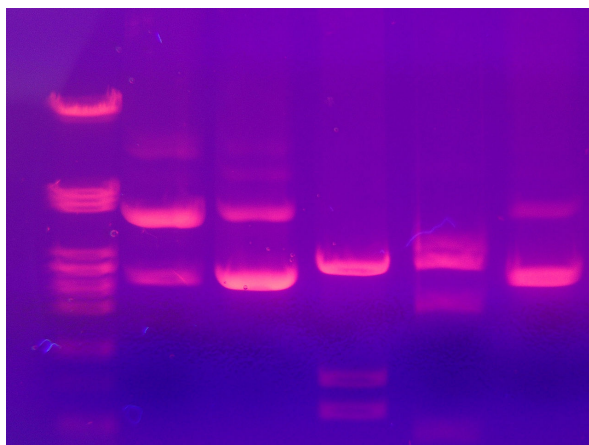


Figure 1.4: **An electrophoresis gel under a UV light.** Ethidium Bromide (EtBr) has been added to the gel. EtBr binds to DNA and fluoresces in red-orange under UV. Photographs of electrophoresis gel usually show the well on top, meaning that lower bands correspond to fragment that migrate faster in the gel. Photo by Mnolf - Photo taken in Innsbruck, Austria, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=1131449>

**DNA replication.** As it was famously intuited by Watson and Crick, the double-stranded structure of DNA “immediately suggests a mechanism for replication” [WC53]. Indeed, the sequence of each strand of the double helix can be deduced from the other by bases complementarity. The DNA replication mechanism starts with the two strands of the Double-helix being separated, allowing a *DNA polymerase* to bind to each of them. The DNA polymerase is the principal actor of the replication. It reads the strand to which it is attached and simultaneously synthesizes its reverse complement (Figure 1.5).

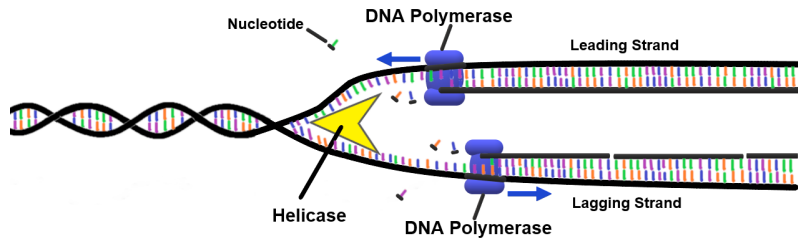
In living cells, DNA replication involves a complex molecular machinery[PO07]. However, the DNA polymerase alone can be used to replicate DNA strands *in vitro* by a method called Polymerase Chain Reaction (PCR) invented by Kary Mullis in 1983 (Figure 1.5b).

The PCR technique is fundamental in molecular biology. It can be used to detect the presence of- and/or duplicate specific regions on large DNA molecule, which has a wide range of applications, including diagnostics and forensic science [ZZX<sup>+</sup>20].

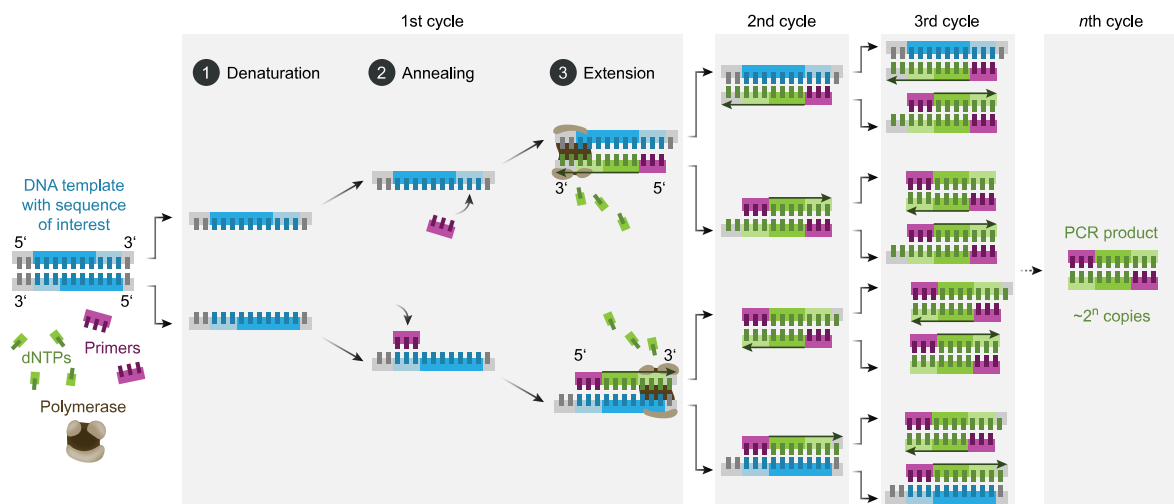
**Holliday junctions.** In 1964, Robin Holliday proposed a model for explaining the mechanism of gene exchanges between distinct DNA molecules [Hol64]. Holliday postulated that these exchanges were made possible by the existence of an intermediate structure involving the four strands of two double DNA helices with similar sequences. In the postulated structure, the strands of the two helices are separated and recombined into a cross-shaped structure (Figure 1.6). In the 1970s, the structure was proven to exist by electron microscopy evidence (Figure 1.6c), and was named *Holliday junction*.

The Holliday junction is an example of a complex where each strand is connected to several others. To think about this kind of structure, it helps to divide the strands into *domains*. A domain is a region of a strand that binds to a region of another strand. This other region is called the *complementary domain*.

It is also simpler to make abstraction of the geometry of the DNA double helix and to simply represent the DNA strands as broken lines, where each segment of the line represents



(a) Simplified representation of DNA replication in the cell.



(b) Principle of the PCR

**Figure 1.5: DNA replication by DNA polymerases.** a: Schematic representation of the mechanism of DNA replication in the cell. The helicase separate the two strands of the double helix, and both strands are separately duplicated by a DNA polymerase. b Exponential amplification of DNA by PCR. DNA is put in solution with DNA polymerases, single nucleotides (dNTPs) and primers that are complementary to the limit of the region of interest. The solution is then heated to separate the two DNA strands (denaturation), and temperature is lowered so that the primers can attach to the target DNA template (annealing). The DNA polymerases attach to the primer/template duplex and duplicate the template (extensions). Several cycles of this reaction can be made sequentially, leading to an exponential duplication of the DNA material. Panel a by Christinellmiller, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0/>, via Wikimedia Commons Panel b by Enzoklop - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=96042657>

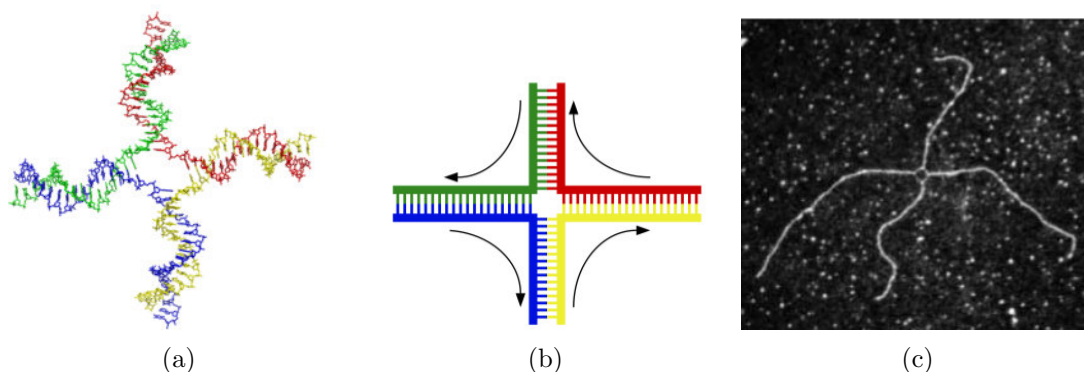


Figure 1.6: **Structure of the Holliday junction.** a: Molecular structure of the Holliday junction. b: Schematic representation of the Holliday junction. The strands are drawn as colored broken lines, each segment representing a domain of the strand. Base pairs are represented as short parallel segments between the complementary domains. The black arrows around the strands indicate the  $5' \rightarrow 3'$  direction. c: Electron microscopy imaging of a Holliday junction, from [PD79].

Panel a By Zephyris on en.wikipedia.org, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=2426907> Panel b from Public Domain: <https://commons.wikimedia.org/w/index.php?curid=11289732>

a domain of the strand, as in Figure 1.6b.

Finally, when domains are named, the star symbol ‘\*’ is used to designate the reverse complement: The domain  $A^*$  is the reverse complement of domain  $A$ .

## 1.2 DNA nanotechnologies

**Foundations.** In 1982, Ned Seeman introduced the idea that using Watson-Crick base pairing could be used to design synthetic DNA strands that would assemble into lattices with crystalline structure [See82] (Figure 1.7a). Seeman’s motivations at the time was to use DNA as a template for protein crystallization (Figure 1.7b).

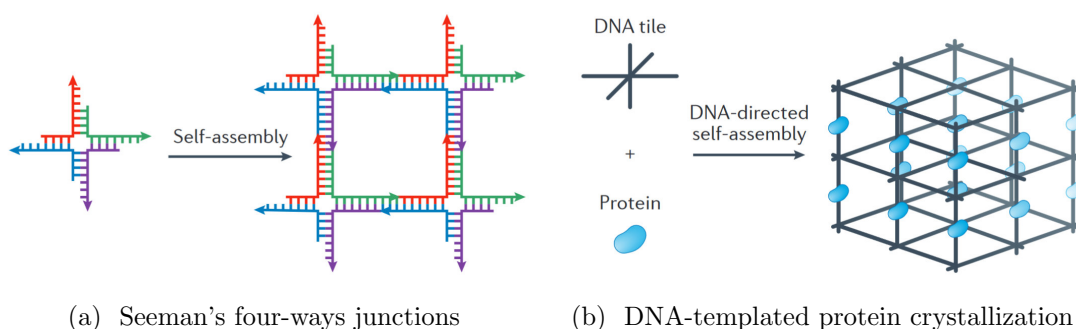


Figure 1.7: **The beginning of DNA nanotechnology.** a: A four way junction that can assemble into a lattice as described by Ned Seeman [See82]. b: Seeman’s motivation was to use DNA as a template for protein crystallization.

In order for the strands to assemble into a lattice, the sequence of the strands would be designed so that each strand binds to several others forming a structure similar to the

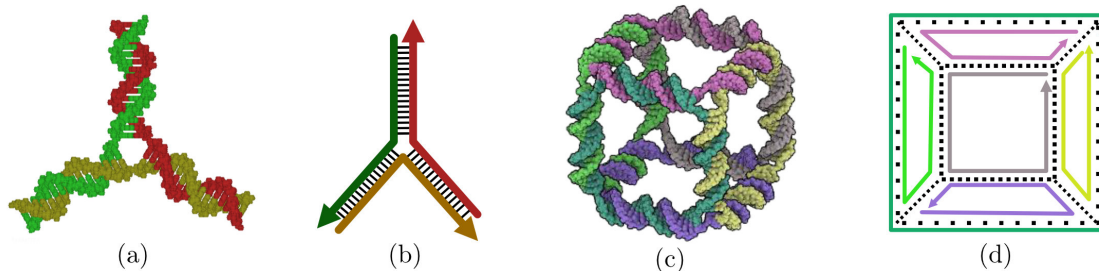


Figure 1.8: **Early DNA nanostructures.** a: A three arm junction, as imagined and assembled by Ned Seeman [See82, KMS83]. b: Schematic representation of a three-way junction. The strands are represented as broken lines where each segment represents a domain of the strand. The black lines represent H-bond between complementary domains. c: A DNA cube made of three arm junctions assembled in 1991 [CS91]. d: Schematic representation of the structure of the DNA cube using a stereographic projection. The dotted black lines separate complementary domains. Panels a and c are extracted from a presentation by Lulu Qian (2016).

Holliday junction (Figure 1.7a).

The year after, Seeman experimentally demonstrated the realizability of his idea by designing strands that assemble into stable four-arms junctions [KMS83], thus founding the field of DNA nanotechnologies.

In 1991, the first artificial 3D DNA nanostructure was assembled [CS91]. It was a cube made of 6 strands, one per face, that each had 4 domains, one per edge of the face (Figure 1.8). Domains that were associated to the same edge were designed to be complementary to each other.

**Molecular computing.** In 1993, Leonard Adleman, as he was studying the fundamentals of molecular biology, was amazed by what he learned about the DNA polymerase. He felt that its behavior was analogous to that of a Turing machine, an abstract machine that computes by reading and writing to a linear tape [Adl98]. This motivated him to explore the computational capabilities of DNA and enzymes.

To do so, he decided to tackle the problem of finding a Hamiltonian path in a graph (*i.e.* a path that goes through every vertex exactly once), a problem of notoriously high combinatorics. His strategy [Adl94] was as follows:

- To encode a graph  $G = (V, E)$  with DNA strands, each vertex  $X \in V$  would be associated to a strand with two domains  $X_-$  and  $X_+$ , and each edge between two vertices  $A$  and  $B$  would be encoded by a strand with domains  $A_+^*$  and  $B_-^*$ .
- In solution, the strands would then randomly attach by base complementarity, forming random valid paths. The path would then be ligated by a Ligase, amplified by PCR, and filtered by length using gel electrophoresis, selecting only the paths of length  $|V|$ .
- For each vertex  $X \in V$ , the selected paths would be filtered by a process using labelled probe that would filter-out any paths that does not contain the sequence corresponding to  $X$ . In the end, the remaining paths would contain all the vertices and have a length equal to  $|V|$ , which means that they would all be Hamiltonian paths.

Using this procedure, Adleman was able to find a Hamiltonian path on a graph with 7 vertices and 14 edges, proving the applicability of his method [Adl94].

**DNA tiles and Algorithmic self-assembly.** Among the achievements of the early years of DNA nanotechnology, is the double-crossover DNA tiles [FS93] that could be used as a building block for larger self-assembling structure.

In 1998, Eric Winfree designed a self assembling 2D lattice of DNA strands [WLWS98]. The lattice was made of a repeating pattern of *tiles*. Each tile was made of four strands, and terminated on both sides with *sticky ends*: domains that are complementary to the sticky ends of another tile. This allows the tiles to attach, growing into a self assembling lattice (Figure 1.9).

The behavior of DNA tiles in this assembly is reminiscent of Wang’s tiles [Wan61] and cellular automata theory. This is not a coincidence: In his Thesis [Win98], Winfree developed a computational model similar to Wang’s tile and called *abstract Tile Assembly Model* (aTAM). He proved that the aTAM was able to simulate any algorithm, and used the 2D self-assembly as a proof that this model of computation could be implemented with molecular tiles made of DNA.

This was not the first time that the computational power of DNA was investigated since Adleman’s work (*e.g.* [BDLS96, Rot96]), but Winfree’s approach stands out by the simplicity of its experimental implementations. In the other approaches, use of enzymes and many experimental steps are required. In contrast, in Winfree’s model, the computation is done by the attachment of the DNA tiles, and the result of the computation can be read in the final assembly. This allows the computation to be driven by a “one-pot” reaction [Win98].

In 2004, Paul Rothmund and Winfree designed a set of DNA tiles that assembles into a fractal pattern: a Sierpinski triangle [RPW04]. This was done by designing a set of abstract tile that implemented a Cellular Automaton whose cell update according to the exclusive or (XOR) pattern (Figure 1.10). This constituted the first experimental proof that the aTAM could be used to assemble complex pattern out of simple DNA tiles.

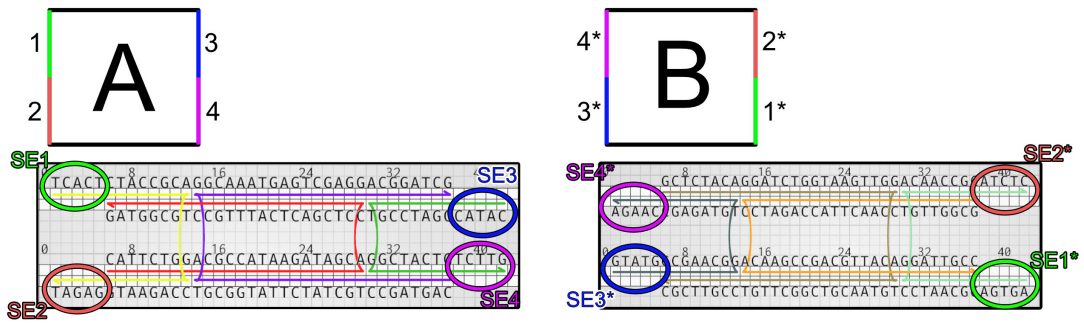
Remarkable latter successes in the field of molecular computing include: a cargo-sorting DNA robot [TLJ<sup>+</sup>17]; implementations of neural network computations with DNA [QWB11, CQ18]; the realization of a molecular 5-bits counter [Eva14] (Figure 1.11b); and the realization of a single programmable set of DNA tiles that implements a wide variety of 6-bits algorithms [WDM<sup>+</sup>19] (Figure 1.11c).

**Other elementary structures.** In addition to DNA tiles, other elementary structures have been developed that can be used as a basis for building larger objects.

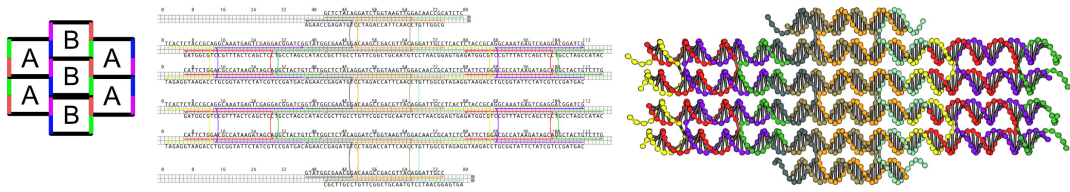
*DNA origami* [Rot06] is a technique invented by Rothmund in 2006 that allows to easily create DNA nanostructures with a great variety of shapes, and a size that is typically between 7,000 and 8,000 base pairs (bp). The details of this technique that “transformed the landscape of DNA nanotechnology” [SS17], will be the subject of the next chapter.

A DNA origami can be used together with DNA tiles to serve as a *nucleation seed* from which the growth of the structure is initiated [LYJ<sup>+</sup>14, MVC<sup>+</sup>17, WDM<sup>+</sup>19]. It can also be used as an elementary component of much larger structures, up to the micrometer scale (Figure 1.13). To that end, the origami components can be bound by Watson-Crick base pairing of sticky ends [TPQ17a, WME<sup>+</sup>22], or by shape complementarity [GWND15, WSD17, MAM<sup>+</sup>22].

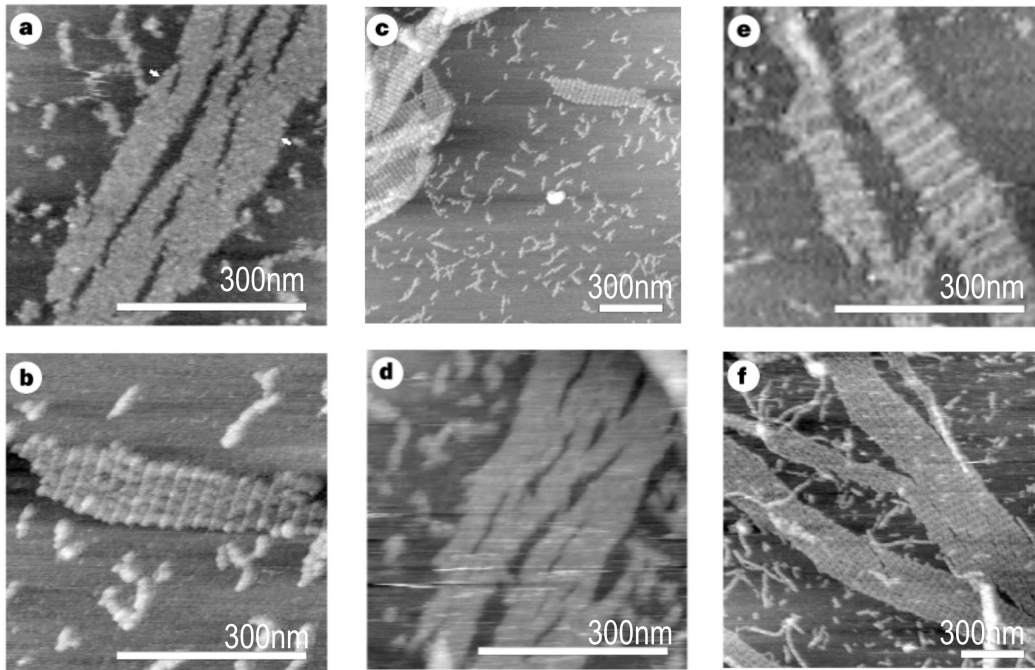
Finally, *DNA bricks* [KOSY12], also called *single-stranded tiles* (SST) are elementary blocks made of only four domains that can attach to each other like Lego bricks to form large and complex structure (Figure 1.12).



(a) Abstract tiles and their DNA implementations



(b) Theoretical assembly



(c) Experimental assembly

Figure 1.9: **Design and assembly self-assembling 2D lattice made of DNA tiles [WLWS98].** a: The two different class of tiles and their implementation as DNA strands. Each tile has four sticky ends (SE) that are associated to specific domains. b: Assembly of the tiles into a lattice in the aTAM, and as DNA strands. c: Atomic Force Microscopy (AFM) images of the experimental assembly of the lattice with DNA tiles as shown in [WLWS98].



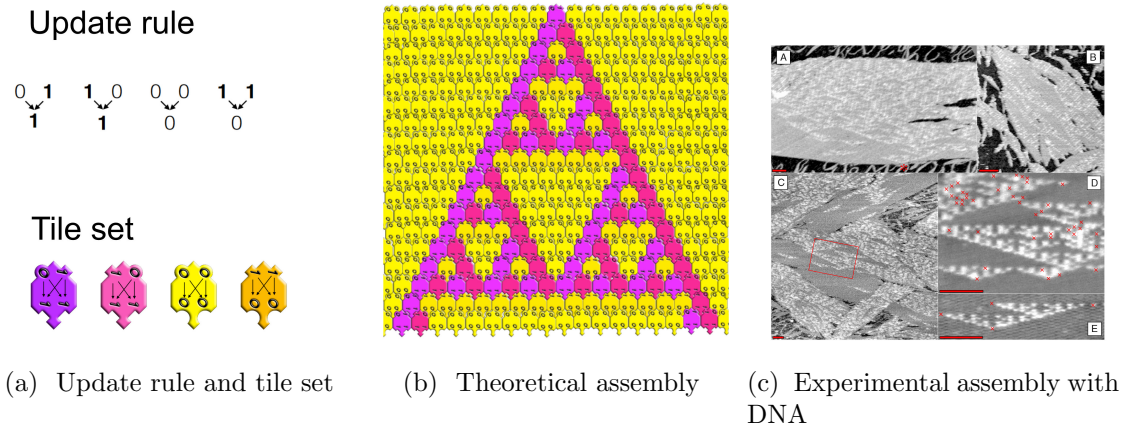


Figure 1.10: **A set of DNA tiles that grow into a Sierpinski triangle.** a: Update rule and tile set of the cellular automaton. The cell is updated with the value of the XOR of its neighbors. Each tile has two inputs and one duplicated output equal to the XOR of its inputs. b: Theoretical self-assembly of the set of tiles, as predicted by the aTAM. c: AFM images of a realization of the cellular automaton with DNA tiles by Rothemund and Winfree. Illustrations on Panel a and b were drawn by Nicolas Schabanel. Panel c is extracted from [RPW04].

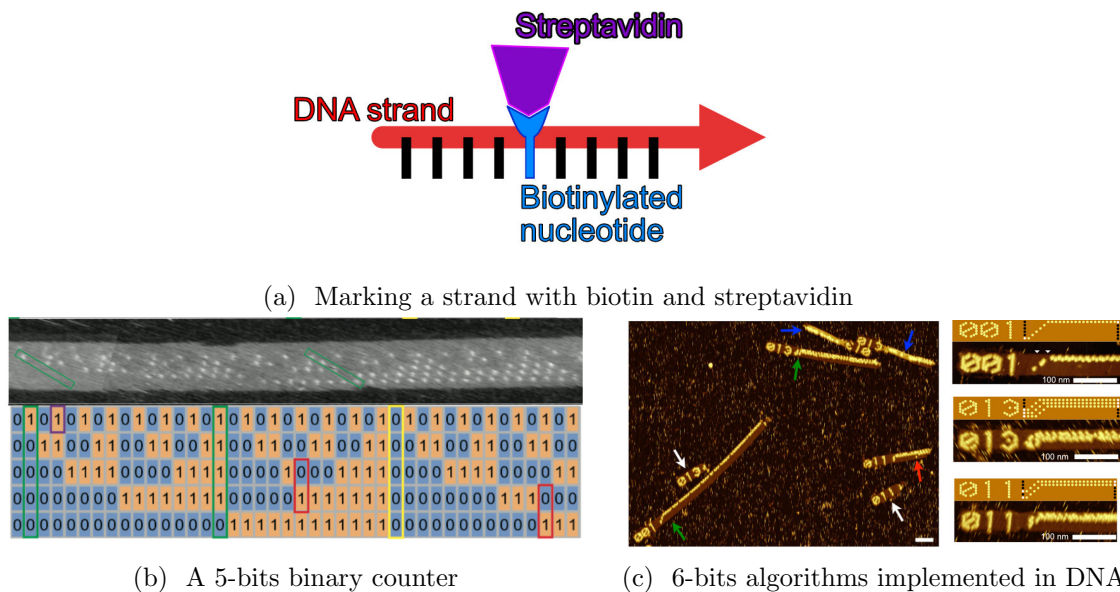
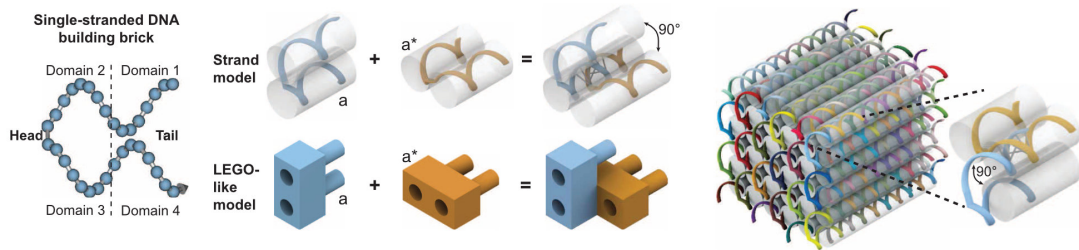
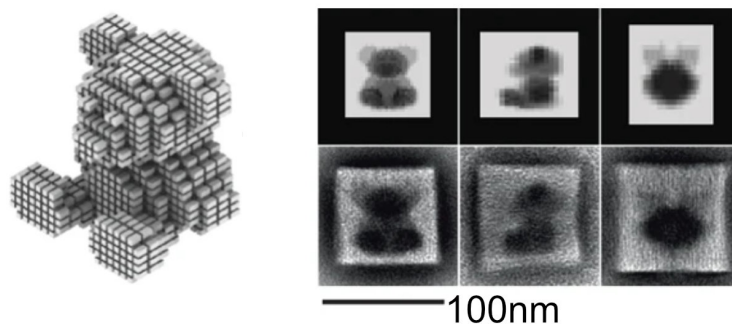


Figure 1.11: **Computing with DNA.** a: Marking strands with biotin and streptavidin. Strands can be marked by having them incorporating a special biotinylated nucleotide. The biotin will strongly attract streptavidin molecules which will make the strand stand out during AFM imaging. b: A 5-bits binary counter. The 1-valued bits are identified using special DNA strands with biotin attached to the backbone. Panel extracted from [Eva14]. c: DNA ribbons that compute the execution steps of various 6-bits algorithms as they self-assemble. Again, 1-valued bits are identified by biotin/streptavidin marking. Panel extracted from [WDM<sup>+</sup>19].



(a) Principle of SST assembly



(b) A Teddy bear made with DNA using SST

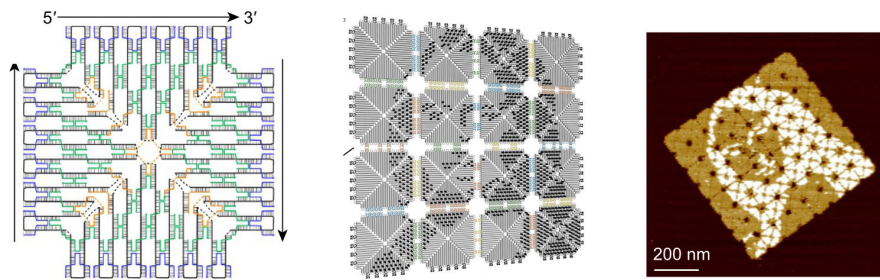
Figure 1.12: **Building DNA nanostructures with SST.** a: Principles of the SST method [KOSY12]. Each tile consists in a single strand with four unique domains, allowing it to bind to four different other tiles, forming a  $90^\circ$  angle with each connection. b: A DNA nanostructure with the shape of a Teddy bear assembled using the SST technique [OHY<sup>+</sup>17]. Illustrations extracted from the referenced source.

### 1.3 Applications of DNA nanotechnologies

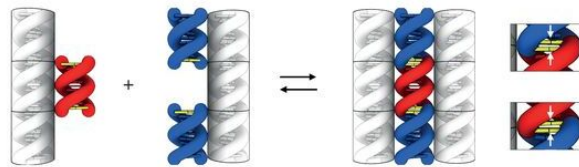
**In engineering.** As we have seen, DNA nanotechnologies allow the construction of nanoscopic objects with 2 nm precision. This has found use in engineering fields such as nanophotonics or nanoelectronics where devices must be designed with nanometer-scale precision. In that context, DNA nanostructures can be used as geometric guides for the assembly. They can for example be used as a template for gold nanoparticles arrays [LCP<sup>+</sup>99, KSF<sup>+</sup>12], which can be used to create sensing devices [THS<sup>+</sup>14] based of the plasmonic coupling phenomenon [JHES07]. DNA nanostructures can also be used as frame for lithography in nanoelectronics [DM04, DBG<sup>+</sup>16].

**In structural and molecular biology.** As it was already well-known in the 1980s, the biological function of proteins is strongly related to their molecular structure [Whi13]. For that reason, the examination of proteins structure is essential to the understanding of many biological processes. X-ray crystallography is one of the most prominent techniques of structural biology, but solving the structure of a protein via X-ray crystallography requires to be able to crystallize the said protein [DF96]. This is why Seeman's original motivations for designing DNA junctions that would self assemble into lattices was to use these structures to

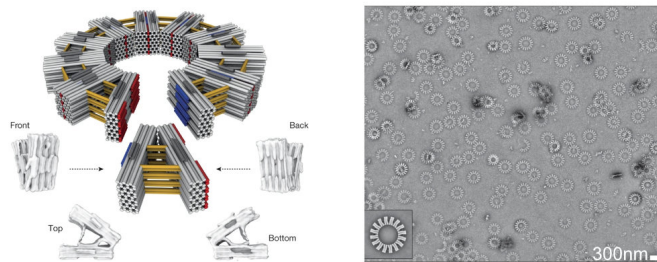




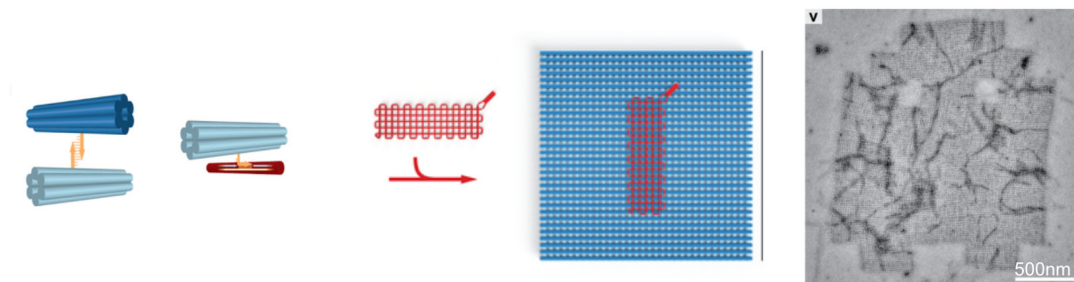
(a) Large canvas made of DNA origami tiles



(b) Principle of shape complementarity



(c) V-shaped origami assembled into a ring by shape complementarity



(d) Criss-cross polymerization of DNA origami

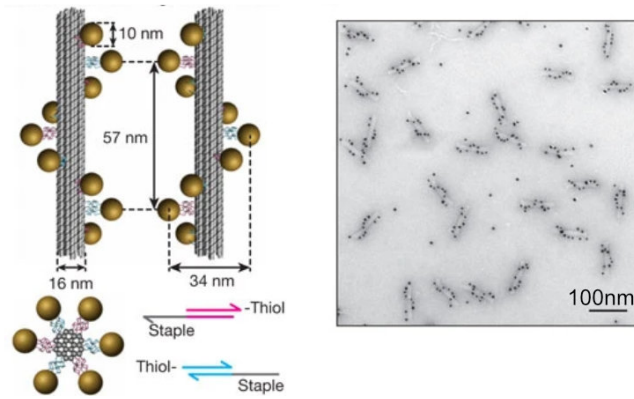
Figure 1.13: **Using DNA origami as elementary blocks for building large DNA objects.** a: Large canvas made of DNA origami tiles [TPQ17a]. DNA origami tiles are assembled into origami arrays with arbitrary patterns. b: Principle of shape complementarity in DNA nanostructures [GWND15]. Stacking interaction will make the red helix attach to the docking site in the blue helix. This attachment will only happen at high enough Magnesium concentration and can be reversed by lowering the salt concentration. c: Large ring produced by programmable oligomerization of DNA origami [WSD17]. Several V-shaped origami are assembled into one large structure by shape-complementarity. d: Crisscross polymerization of DNA-origami salts [WME<sup>+</sup>22]. Cylinder-shaped DNA origamis are assembled into grids that can be used as uniquely identified pixel to grow large structures up to the micrometer scale. All the illustrations on this Figure were extracted from the referenced sources.

control protein crystallization. Since then, DNA has effectively been used as a template to form protein array [SAA<sup>+</sup>11], and a DNA-nanotube based method was described to determine the structure of membrane proteins [DCS07].

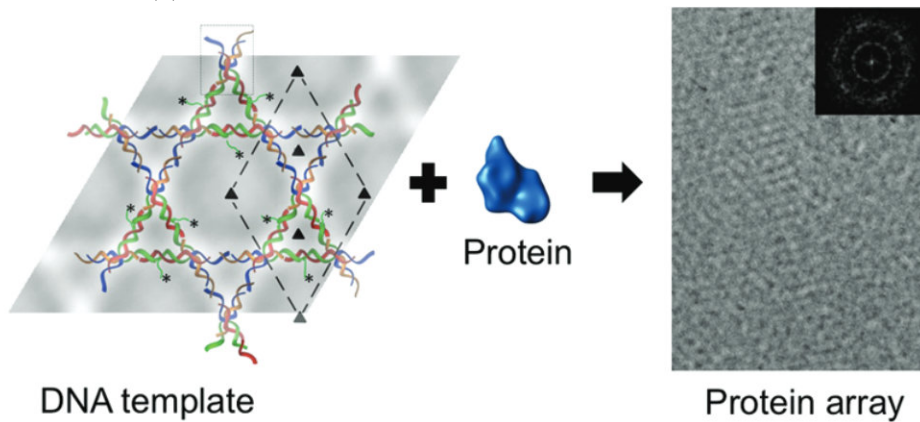
DNA nanotechnologies can even be used in sensing devices for *in vivo* applications. In 2011, a DNA-based molecular device was designed that would sense pH change in a living organism [SBKK11].

**Medical applications.** The success of DNA as a material for life-compatible devices also spread to area like drug delivery. In 2014 DNA origami were used to deliver antitumoral drugs in targeted cancerous cells [ZJL<sup>+</sup>14].

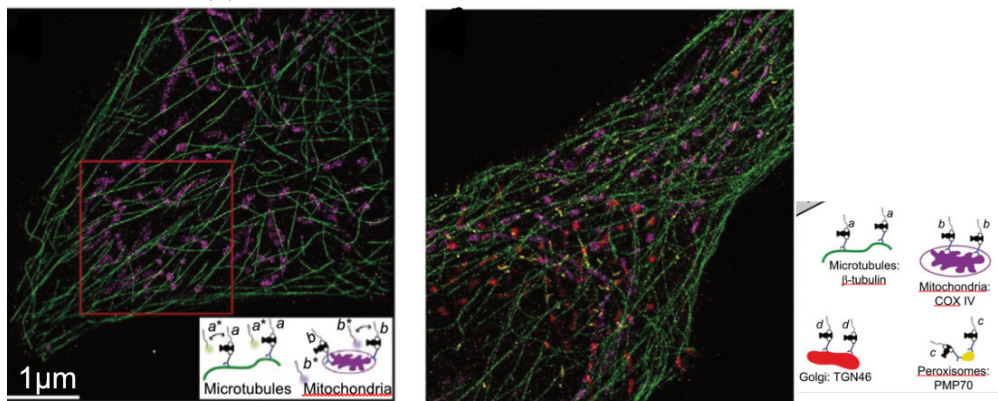
There are also promising recent results that suggest that DNA nanostructures could be used in vaccine as antigen presenting platforms [VMS<sup>+</sup>20], or in therapeutic devices as virus trapping mechanisms [MKS<sup>+</sup>22].



(a) Using DNA as a template for gold nanoparticles



(b) Using DNA as a template for protein crystallization



(c) Using DNA for super-resolution imaging

Figure 1.14: **Example of applications of DNA nanotechnologies.** a: Using DNA as a template to build nanohelices with gold nanoparticles [KSF<sup>+</sup>12]. Gold nanoparticles carry DNA strands that are incorporated at specific position in the origami, allowing the gold particles to be attached the surface. b: Using DNA as a template for protein crystallization for single-molecule imaging [SAA<sup>+</sup>11]. DNA strands are assembled into a lattice. These strands carry modified nucleotides for protein attachment. Fixation of the protein on the lattice allows its observation by electron microscopy. c: Super-resolution imaging of fixed cell with the DNA-PAINt technique [ZJL<sup>+</sup>14]. All images were taken from the referenced sources.

## Chapter 2

# DNA origami

### 2.1 General Presentation of the technique

DNA origami [Rot06] is a technique invented by Paul Rothemund in which a natural single strand of DNA called the *scaffold*, is hybridized with many short artificial strands, called the *staples*, leading it to fold into a controlled shape.

The scaffold is typically extracted from the M13 bacteriophage whose genome is supported by a circular single strand of DNA that is typically about 7000 nucleotides long. Working with this natural DNA strand is a way to overcome the technical difficulties of synthesizing DNA strands with a length over a few hundreds of nucleotides [BI92].

Each staple is designed to be made of several domains that are complementary to distinct regions of the scaffold. The binding of a staple to the scaffold will make these regions co-localize, thus folding the scaffold (Figure 2.1).

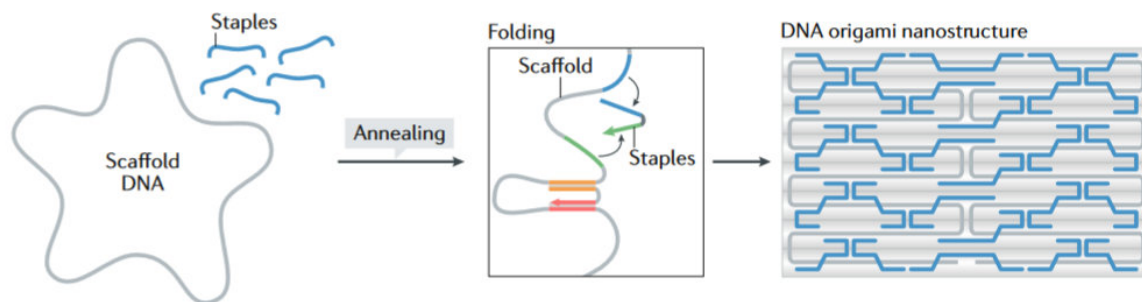


Figure 2.1: **Principle of the DNA origami method.** A long single stranded DNA scaffold is mixed in solutions with several short strands called the staples. The staples to be complementary to domains that lie on distinct regions of the scaffold. When a single staple attach to all its complementary domains, it makes them co-localize, thus folding the scaffold strand. Illustration taken from [DFG<sup>+</sup>21]

The production of DNA origami is a five-steps process (Figure 2.2).

- 1. Design of the shape.** The designer chose the desired shape for the folded origami. As we will see in Section 2.2, refinement in design techniques have allowed to broaden the variety of shapes in which it is possible to fold an origami.

**2. Dividing the shape into double DNA helices.** The designer must come up with a covering of the desired shape with virtual double DNA helices. These helices will support the scaffold and staple strands.

**3. Scaffold routing and stapling.** A path for the scaffold must be routed in the virtual helices. Often, the scaffold is cyclic and in that case the routed path must of course also be cyclic. Staples are then positioned on the complementary strands on the virtual helices.

When designing the staples and the scaffold's route, it is essential to choose carefully the nucleotides positions at which the strands jump from one helix to another. The spacing between these jumps, called *crossover*, must be adjusted so that the connected helices make approximately an integer number of full turns between consecutive crossovers. Failing to do so can lead to physical stress in the structure and hinders its assembly. For that reason, designers frequently resort to using simulation software to assess the feasibility of their design.

**4. Experimental assembly of the origami.** At this point, the nucleotide sequences of the staples can be deduced from the sequence of their complementary domains on the scaffold. There exists specialized commercial platforms that sell scaffold strands and will synthesize staple strands with custom sequences on demand.

To assemble the origami, all the staple strands are put in with the scaffold in a PCR tube containing a folding buffer solution. The choice of folding buffer and staples/scaffold concentrations is flexible and sometimes needs to be adjusted with trial and error. In Rothemund's original protocol [Rot06] the folding buffer contains 1x TAE, 12.5 mM Mg, 1.6 nM scaffold and 160 nM staple strands. This corresponds to a 100-fold excess of staple strands. Nowadays, some authors (e.g [CKK<sup>+</sup>11]) recommend a 5-fold excess of staple strands instead.

The origami is then annealed in a PCR machine. Again, the precise temperature ramp is flexible and may need to be adjusted to the specificity of the design. A good starting point is to start at 80°C and do a fast descent to 60 °C at 5 min per degree, then a slow descent from 60 °C to 25 °C at 300 min per degree [CKK<sup>+</sup>11].

**5. Evaluation of the folding's quality.** The quality of the folding can be evaluated by electrophoresis on a 1% agarose gel.

When running an electrophoresis on DNA origami, all the objects that migrate have the same molecular weight. However, the migration speed of a DNA nanostructure is also affected by its shape: compact objects progress faster in the gel than linear ones. This is why a distinct band is usually the sign of a good, consistent folding, while a "smear" may indicate that a significant portion of the objects present structural defects and that the design/protocol might need to be further optimized (Figure 2.2e).

Optionally, the electrophoresis gel can also be used to purify the sample before imaging with AFM or TEM to observe the shape

The optimization of the experimental conditions in the assembly step is an important subject, but is beyond the scope of the present work. However, we think that design quality, in terms of scaffold routing and staple crossover locations, is key to a successful assembly.



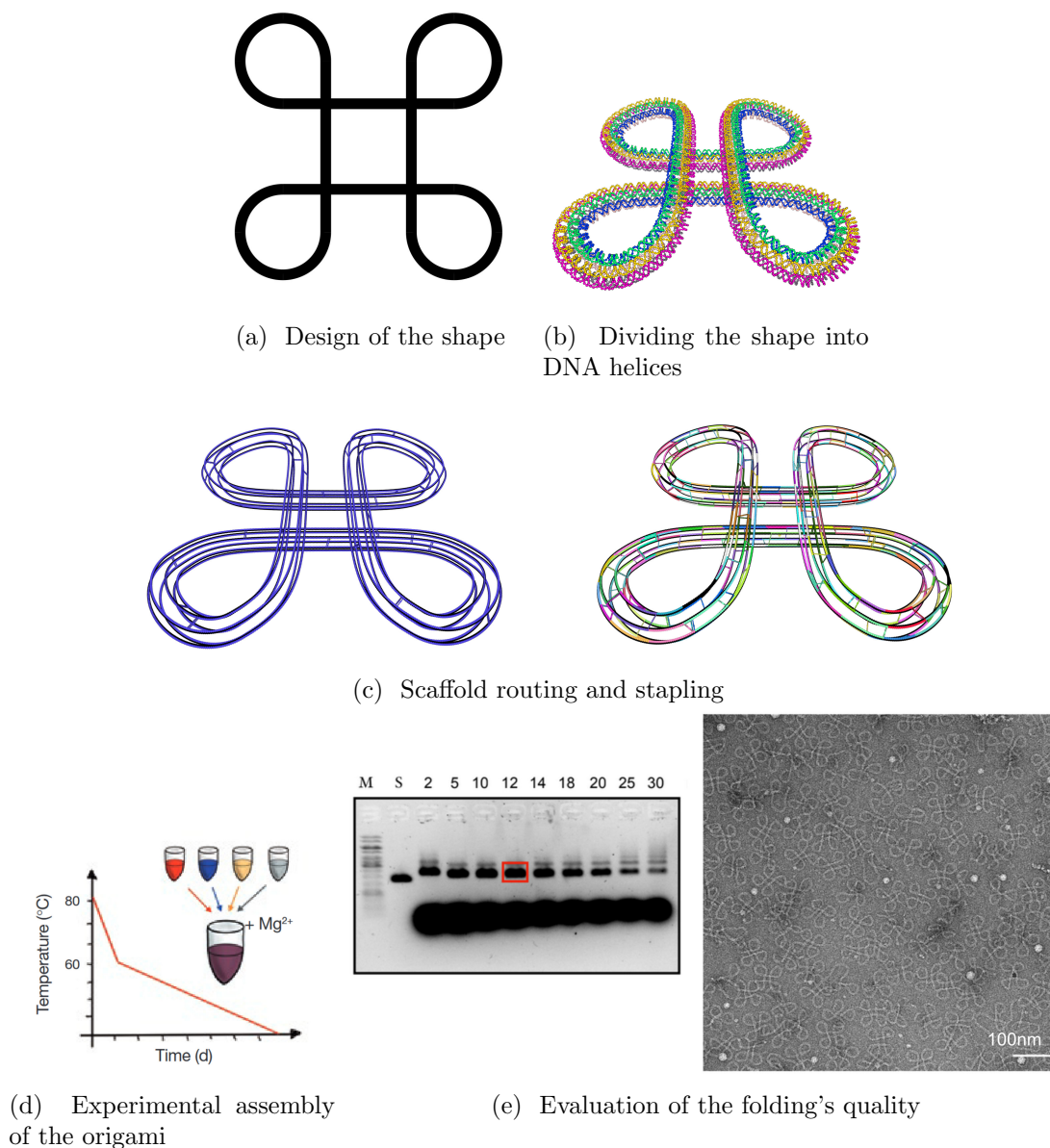


Figure 2.2: **The five steps of the DNA origami production.** The process is illustrated by an origami that we design and assemble in Section 9.3 using the new design techniques that we develop in Part III. a: Design of the shape. Here we chose to design an origami with the shape of a looped square. b: Filling the shape with DNA double helices. Here we use a hexagonal bundle of helices and make it follow the desired shape. c: Scaffold routing and stapling. The scaffold (on the left) is routed along all the helices of the bundle. Each staple (on the right) go through two or three helices and hold them together. d: Experimental assembly of the origami. The strands and staples are put together in solution in a PCR tube and go through an annealing ramp. The solution is brought to a temperature where all DNA molecules are single-stranded. The temperature is then progressively lowered, allowing the staples to bind the scaffold and fold the origami. e: Evaluation of the folding quality. Gel electrophoresis allows to quickly assess the folding quality. Here, the 'M' well is the molecular scale, the 'S' well is the scaffold strand alone, and the numbered wells correspond to origami annealed with  $x$  mM Mg. At magnesium concentration between 5 and 20mM, a distinct band is visible which is often sign of good folding quality. At higher magnesium concentration, a smear is visible which indicates poorer folding quality. Imaging of the origami (in that case via TEM) provide more precise structural information. Panel d is extracted from [CKK<sup>+</sup>11].

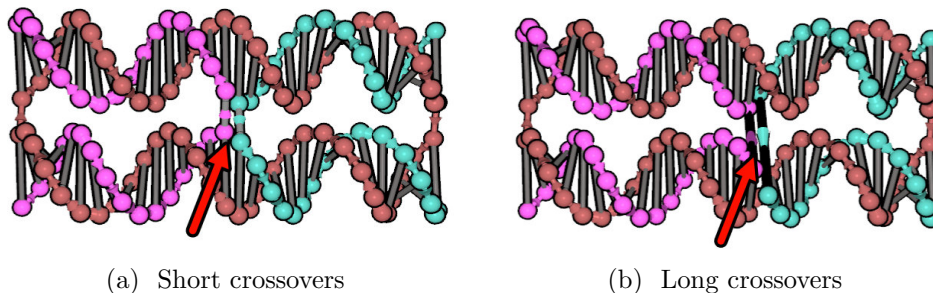


Figure 2.3: **Influence of the position of crossovers on their length.** Panels a and b both present a design with circular DNA strand (brown) bound by two staples (purple and blue) to go through two DNA helices. On Panel a, the position of the staples’ crossovers is well-chosen, so the corresponding covalent bonds are not too stretched. On Panel b however, the crossovers connect nucleotides that are further apart, which may result in undesirable strain in the actual structure.

**Importance of geometry in origami design.** As we already mentioned, special care must be given to the positioning of crossovers in a DNA origami design. This is because crossovers are a succession of covalent molecular bonds, and therefore have a constrained length. The distance between the nucleotides of neighboring helices varies greatly due to the shape of the helix (Figure 2.3).

Attempts to connect distant nucleotides in a design will result in a physical stress that might cause the assembly to adopt a shape different from the one intended by the designer. Undesired shapes can be the result of even subtle mistakes in the modeling of the geometry of the helix. A classical example was given in [WR11] where an origami with an undesirably twisted shape could be fixed by slightly adjusting the geometric parameters of DNA that were used to choose crossover positions in the design: The design was fixed by assuming a number of base pairs per turn of 10.44 instead of 10.67 (Figure 2.4).

## 2.2 Design and shapes of DNA origami

DNA origami is a flexible method that can be used to create nanostructures with a great variety of shapes.

These shapes can be organized into several classes, and several design techniques exist. Some of these techniques are general while some are specific to certain classes of shapes.

**Flat, single layer DNA origami (Figure 2.5).** Rothemund’s first origamis [Rot06] were flat structures (Figure 2.5). The desired shape was filled with stacked double helices. In the simplest designs, all the helices were parallel, but Rothemund also designed triangular shapes with rectangular and trapezoidal edges.

Despite the apparent simplicity of designing flat single layer structures, care is still required. For example, if the crossover positions are chosen using a model that approximate the geometry of the double helix, for example by assuming that the number of base pairs per turn of a DNA double helix is a fraction of small integers, the resulting error accumulate and can lead to twisted structures [WR11] (Figure 2.4).

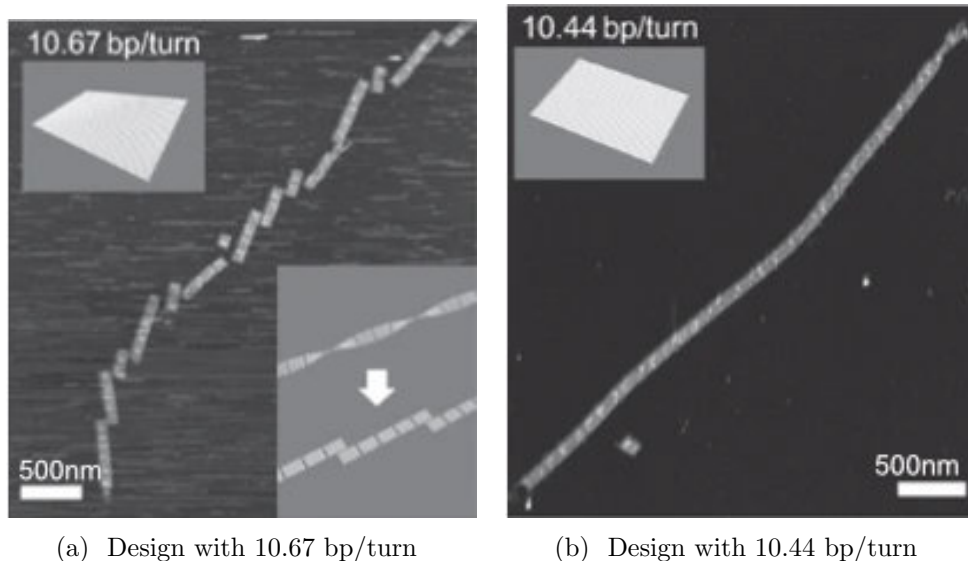


Figure 2.4: **Importance of the value used for the number of base-pair (bp) per turn [WR11].** Authors attempted to design DNA origami with the shape of flat rectangles that would assemble into long chains because of stacking interactions. On Panel a, the rectangles were designed using a software called cadnano that uses a model of DNA in which the double helix makes a full turn every 10.67 bp. This design resulted in twisted rectangles that could not stack well. On Panel b, the design were fixed by adjusting the crossover positions assuming that the double helix makes a full turn every 10.44 bp. In practice this was done by removing a nucleotide position once every 48 bp. The resulting rectangles were flat and could stack as intended. Since the publication of this work, a value of 10.44 bp/turn is commonly used when designing DNA origami.

**3D origamis (Figure 2.6).** The first 3D origami were assembled in [DDL<sup>+</sup>09]. They were built using helices organized in honeycomb lattices, mimicking some crystalline structures.

**Twisted and curved 3D origami (Figure 2.7)** In many designs, the DNA helices are thought of as straight cylinders. In that case it is not possible to create curved shapes without a certain degree of rasterization. It is however possible to design crossover patterns that induce a curvature in the DNA helices, as proven in [DDS09].

The method consists in creating structures with several layers, and to chose crossover positions that are further apart on the innermost helices than on the outermost ones. This leads to the helices on the outer layers being shorter than those of the inner layers, and therefore causes the structure to bend.

A similar method, also presented in [DDS09], can be used to create twisted bundles of helices. As in [DDL<sup>+</sup>09], the bundles are designed using helices organized in a lattice, but the crossover positions are purposely chosen to have a frequency slightly lower or higher than the helicity of DNA. This creates a stress in the structure that is compensated by a global twist.

In [HPN<sup>+</sup>11], DNA origami with curved 3D shapes were created by stacking concentric



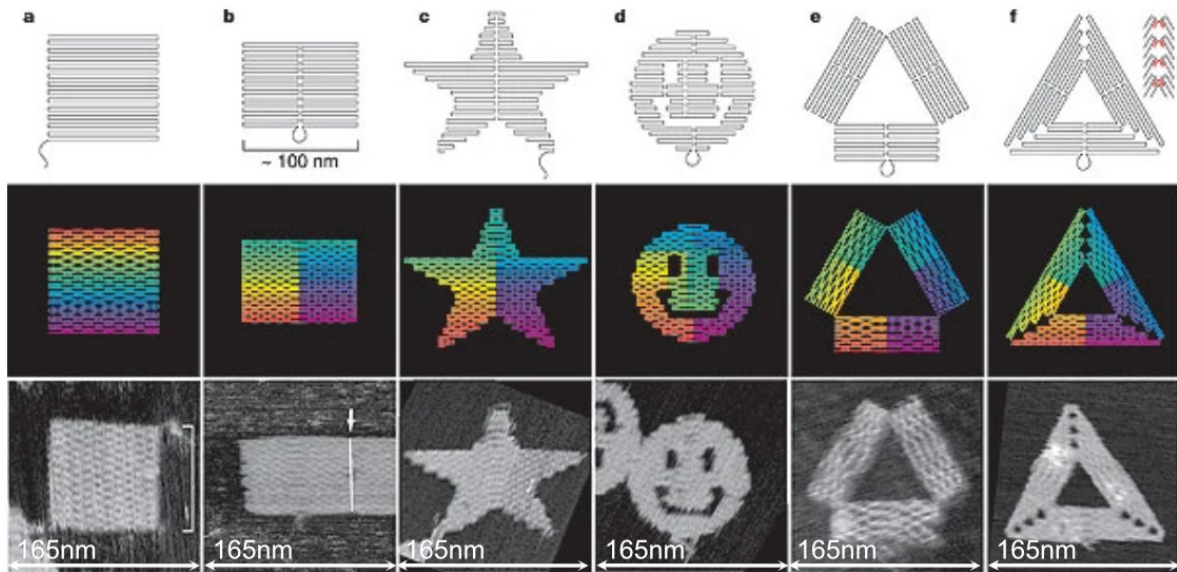


Figure 2.5: **Flat DNA origami [Rot06]**. Flat surfaces are filled by stacking parallel DNA helices. These structures are the first DNA origami ever published.

rings of DNA with varying radii. Again, the circular shapes of the helices that compose the structure were obtained by using a carefully designed crossover pattern, and a correspondence table between crossover period and radius of curvature was established.

In Part III, we will present our own original method for designing curved 3D origami.

**Wireframe origami (Figure 2.8).** The wireframe method for designing DNA nanostructures [HYS<sup>+</sup>08, HPY<sup>+</sup>13] (not necessarily DNA origami) consist in using Seeman's junction as vertices in a network of DNA helices. DNA origami with polyhedral shapes can be assembled using this technique [HPY<sup>+</sup>13, BMG<sup>+</sup>15].

The special structure of wireframe designs allows the use of graph theory algorithms to automate scaffold routing and stapling, as well as tessellation algorithms to convert surfaces into network of DNA helices.

**Reconfigurable and dynamic DNA origami (Figure 2.9).** Various method exist to reconfigure annealed DNA origami. It is for example possible to introduce DNA strands that will displace staples on the origami and thus modify its structure [ZNL12].

Strands can also be displaced by proteins. This is used as an unlocking mechanism in a targeted drug delivery device in [DBC12] (Figure 2.9b). It is also possible to design structures that will move in reaction to an electric field [KLM<sup>+</sup>18] or a modification of the salt concentration [GWND15] (Figure 2.9a).

Specific design techniques have been developed to design objects with mechanical joints such as kinks and hinges [WES<sup>+</sup>17]. These elementary joints can then be integrated in devices that achieve more complex motions [MZSC15].

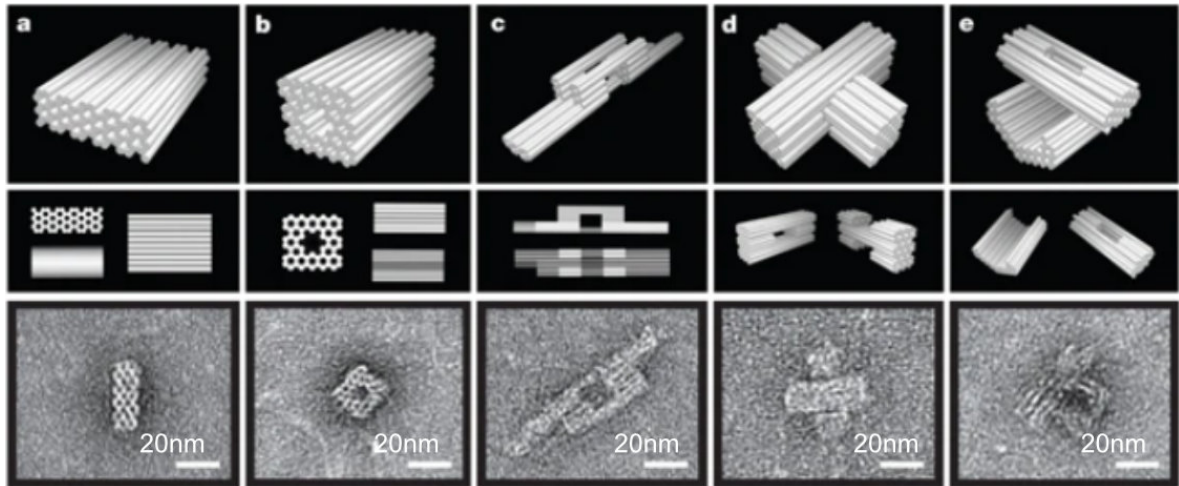


Figure 2.6: **3D DNA origami [DDL<sup>+</sup>09]**. Objects are built by arranging DNA helices in a honeycomb lattice.

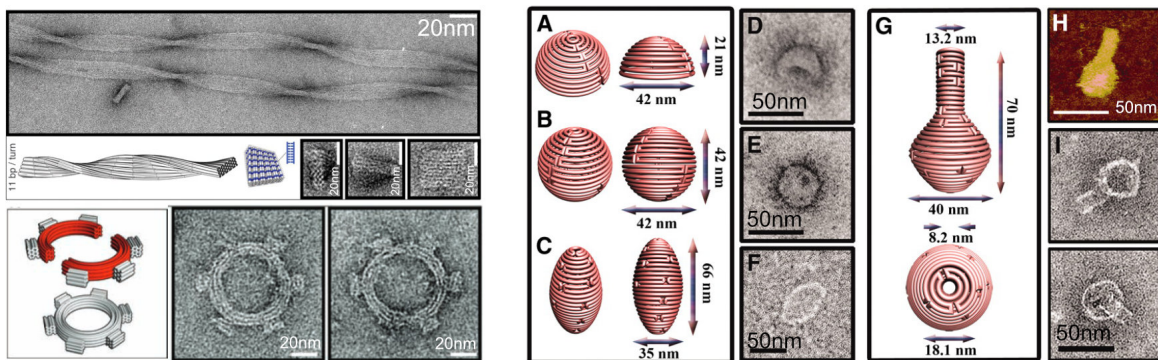


Figure 2.7: **Left: Twisted and curved DNA origami bundle [DDS09]**. The design are made using a design technique similar to that employed for the 3D origamis in Figure 2.6, but the frequency of the crossover is adjusted to induce twist and curvature in the structure. **Right: DNA origami with 3D curvature [HPN<sup>+</sup>11]**. These curved origami are made by stacking concentric DNA rings.

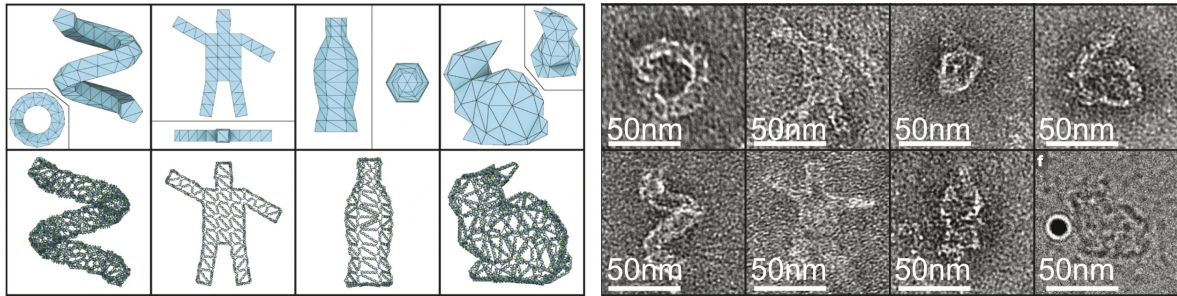


Figure 2.8: **Wireframe DNA origami [BMG<sup>+</sup>15]**. These origamis were designed using an automatic method that converts 3D meshes into lattices of DNA helices and automatically creates the staples and scaffold path.

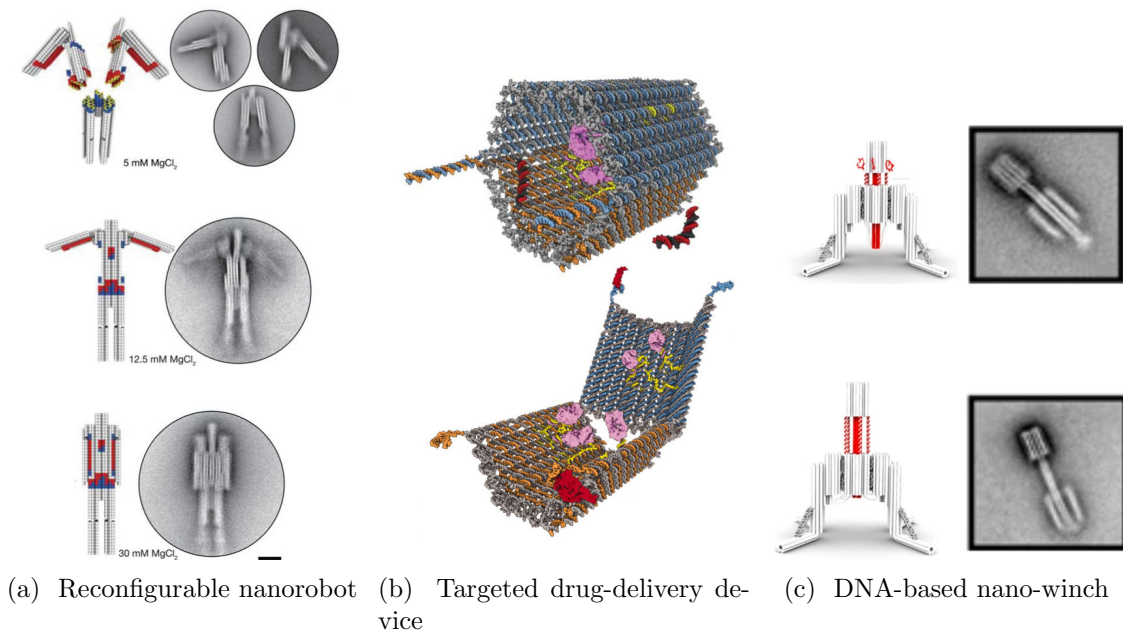


Figure 2.9: **Dynamic and reconfigurable DNA origami**. a: Reconfigurable DNA nanorobot that reacts to change in salt concentration [GWND15]. The robot is made of three origami that are designed to have complementary shapes. The strength of stacking interactions varies with magnesium concentration, and each level of magnesium concentration corresponds to a configuration of the meta assembly. b: DNA nanorobot with protein payload for targeted drug delivery [DBC12]. The device incorporate a lock mechanism that opens upon interaction with a targeted protein. c: Modular spring-loaded actuator for mechanical activation of membrane proteins [MAM<sup>+</sup>22]. The winch can be extended by adding staples complementary to the strands between the head and the body of the winch. Since double-stranded DNA is more rigid than single strands, the hybridization of the staples elevates the head of the winch.

## Chapter 3

# Computer assisted design of DNA nanostructures

### 3.1 State of the art

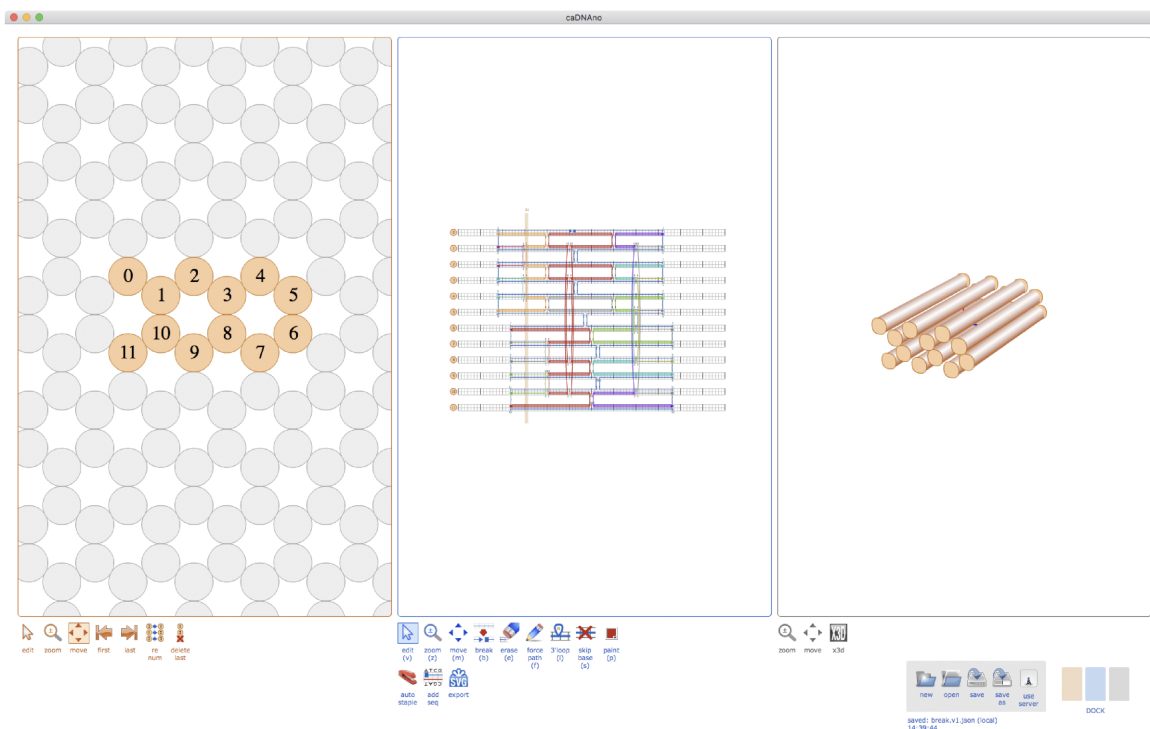


Figure 3.1: **Cadnano's user interface.** The leftmost panel is used to visualize and position the DNA helices in the chosen lattice. The middle panel is used to create the scaffold and staple strands. Each DNA helix is represented as a horizontal grid with two rows, one for each strand, where each square represents a nucleotide. Indications around the helices show possible positions for crossing over to another helix. The rightmost panel shows a 3D representation of the design. Figure taken from [GDS<sup>+</sup>21].

A key step in the design of DNA origami is the positioning of crossovers which constrain the structure geometrically to adopt the desired shape. Various software have been developed to help the designer in this crucial task. DNA nanostructure design software essentially provide four kinds of tools:

**1. Abstract representation of the design.** Some characteristics of the structure of the DNA double helix, such as the existence of a major and minor groove, and the fact that its periodicity (10.44 bp per turn) is not a fraction of small integers, make DNA geometry difficult to grasp. Software such as *cadnano* [DMT<sup>+</sup>09] and *SARSE* [ADN<sup>+</sup>08], propose to make abstraction of the geometry and present the user with only the relevant aspect of the DNA helices.

*Cadnano* provides a framework to position DNA helices next to each other in a square or hexagonal lattice, as well as guides to connect them using crossovers (Figure 3.1). *Cadnano* has an internal representation of the geometry of the double helix and uses it to generate *crossovers suggestions*, that is to say, positions where the nucleotides of neighboring helices are close enough to be connected by a crossover.

*Cadnano* seldom exposes DNA geometry to the user: it offers a minimalistic, non-editable, 3D representation of the design, and visual cues of the nucleotide position in base pair plane.

It should be noted that *cadnano* always assume that all the helices in the design are parallel. Crossover suggestions can therefore not be used for crossovers between two non-parallel helices.

Moreover, *cadnano*'s internal representation of DNA geometry assumes that the DNA double helix makes a full turn every 10.5 (for squared grids) or 10.67 (for honeycomb grids) bp. This value is slightly off, which can lead to errors for design using long helices [WR11]. To overcome this problem, user can use *deletions* in the design. These deletions can be used to shift and correct *cadnano*'s crossover suggestion. For flat designs, it is recommended to add a deletion once every 48 nucleotides [WR11]. Deletions can also be used to design curved and twisted structures using the method described in [DDS09].

**2. 3D editable view of the design.** Various software propose, instead, a 3D rendering of the design, where the user can see the 3D positions of the nucleotides to directly create crossovers at appropriate positions. A notable software in this category is *Tiamat* [WLL<sup>+</sup>09] which introduced a 3D interface capable of fine-tuning and editing of the design in a stand-alone software (Figure 3.2). There also exist tools for designing DNA nanostructures that are distributed as plugins for existing software. Examples include *vHelix* [BMG<sup>+</sup>15] that is a Maya plugin, and *Adenita* [dLMA<sup>+</sup>20] that integrates in the molecular design software *SAMSON*.

Unfortunately, while 3D views are useful for positioning non-parallel DNA helices and creating crossover between them, they are not the better interface for design fine-tuning [GDS<sup>+</sup>21]. Indeed, 3D views are often jammed, even for simple design, and the 3D→2D parallax effect makes it hard to evaluate the length of a crossover, especially because its direction, and thus its apparent length, changes when it is moved around.

**3. Automated and scriptable tools.** When designing a DNA origami, many tasks are repetitive and dull, such as designing the staples strands in a rectangle. On the opposite, some tasks require a high technical level such as routing the scaffold in a wireframe origami,



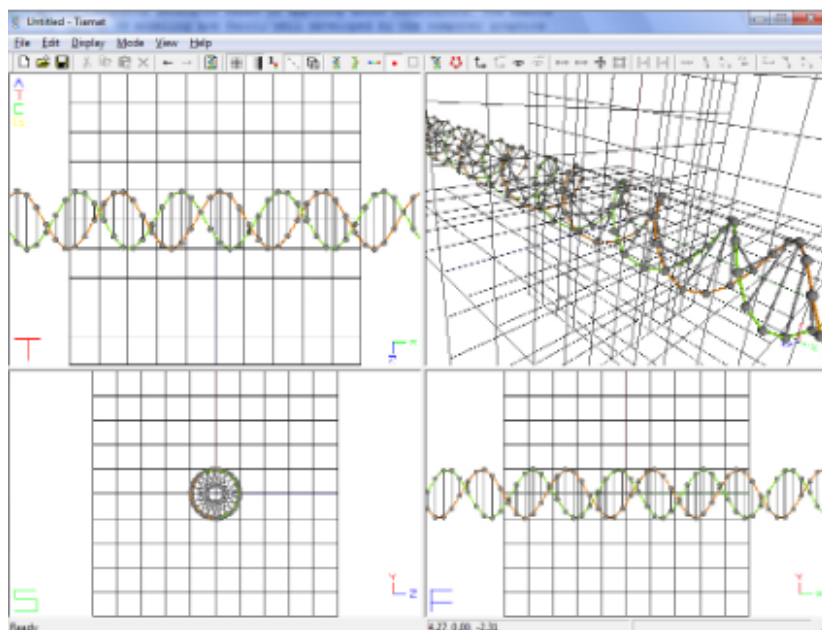


Figure 3.2: **Screenshot of the Tiamat software.** Tiamat offers four 3D editable views of the design. In the top right panel, the camera can be moved freely while the three other panels offer a view of the design from a fixed angle.

or creating staples in a 3D bulk. Software with various levels of automation exist to simplify both these kinds of tasks.

Codenano [LMD] is a browser-based software that allows to design DNA nanostructures using a Rust API, and to visualize these structures in 3D. Codenano also incorporates a finite elements simulator to assess the stability of the design.

Scadnano [DLS20] is a software that offers both a cadnano-inspired graphical user interface, and a Python library to write scripts that output a design. Scadnano defines a programming framework consisting of various function calls to describe helices' and strands' positions.

Other software such as BScOR [BMG<sup>+</sup>15], and the suite from Bathe Bionanolab [JSZ<sup>+</sup>19, JZS<sup>+</sup>19, JWP<sup>+</sup>21] offer fully automated design of various classes of wireframe origami. MagicDNA [HKJ<sup>+</sup>21] (Figure 3.3) proposes an interesting, user-friendlier, alternative approach, generalizing the wireframe technique, which consists in describing the design as graph embedded in space, whose edges are replaced by configurable bundles of parallel helices, along which the routing of the scaffold and its staples is algorithmically computed. Finally, the recent DNAxis [FPNP<sup>+</sup>22] provides an automatic pipeline for designing curved 3D DNA origami based on the method described in [HPN<sup>+</sup>11].

**4. Coarse grain or thermodynamic physics simulation.** Three-dimensional views of a design only represent the desired shape for the design, and strands may not self-assemble as foreseen by the designer. While 3D views are essential to choose the right crossovers to bind strands together in order to achieve the desired assembly, they offer no guarantee in the resulting shape. Physical simulation software such as CanDO [KKDB12] (Figure 3.4),

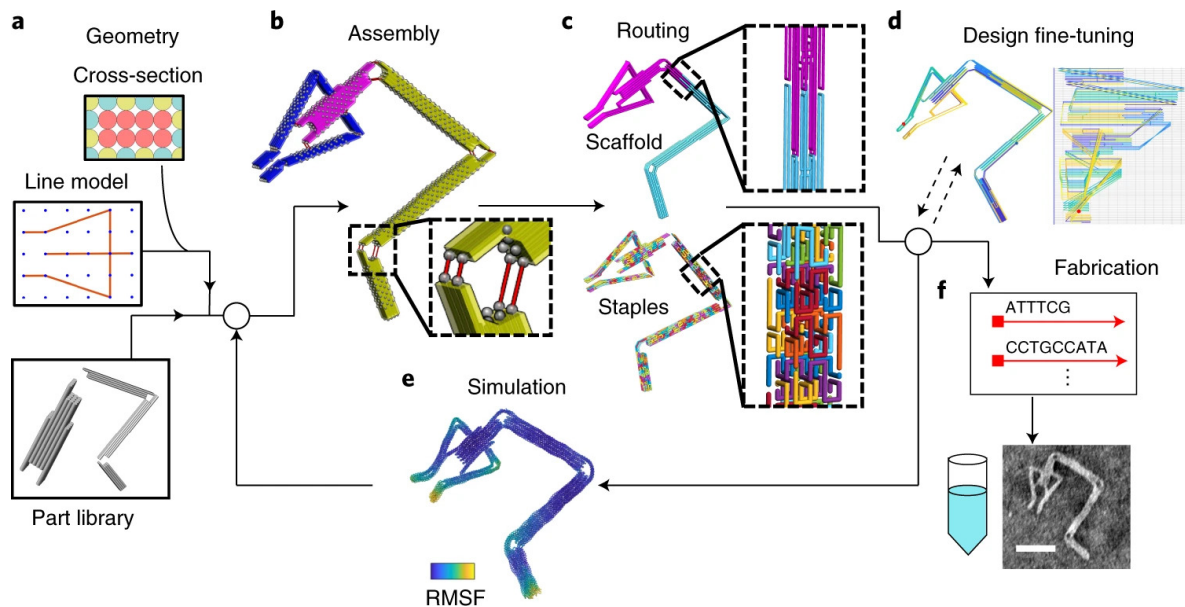


Figure 3.3: **MagicDNA's automated pipeline for designing DNA nanostructures made of several DNA origami [HKJ<sup>+</sup>21]**. Users input a geometric description of the desired assembly including mechanical linkage between components, and MagicDNA automatically generates designs with the corresponding shape. The designs can be exported to cadnano for fine-tuning.

OxDNA [DOL<sup>+</sup>13], MrDNA [MA20] or SNUPI [JWP<sup>+</sup>21], and thermodynamic binding estimation software such as NUPACK [ZSB<sup>+</sup>11] or ViennaRNA [GLB<sup>+</sup>08] allow a much more precise feedback on the feasibility and stability of a design.

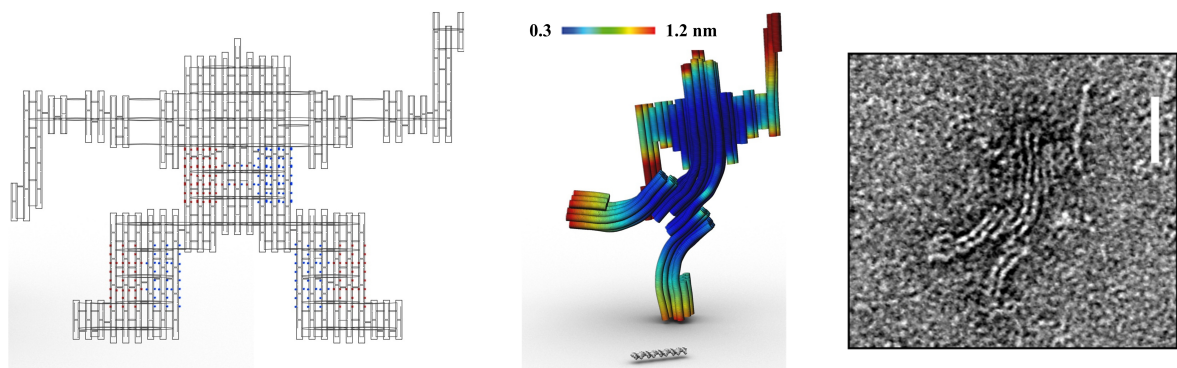


Figure 3.4: **Design of a nanorobot with a curved body with cadnano and CanDO**. Left panel: Blueprint of the origami showing use of insertions (blue dots) and deletions (red crosses) to induce a curvature in the body and legs. Middle panel: 3D structure of the robot as predicted by the CanDO software. The coloring indicates the rigidity of the structure, the blue parts being the most rigid and the red parts being the most flexible. Right panel: TEM imaging of the structure. Extracted from <https://cando-dna-origami.org/examples/>

Simulation software are however computer-intensive and furthermore require a high level

of competency to interact with. They can hardly produce a fast-enough feedback to be use during the design process. They are thus usually used at the end of the design chain. Design software usually offer to export the design into the sophisticated file format of simulation software for validation.

## 3.2 Objectives of our work

**Identification of a good interface for DNA origami.** Three-dimensional, editable views of the design are essential when it comes to design complex 3D structures involving non-parallel, or even curved, DNA helices. However, as we have seen in the previous section, 3D views are not well adapted for fine-tuning. This is especially true for tools that are plugged into software which were not designed for the specific needs of DNA nanotechnology.

As it turns out, almost all software, including the all-automated ones, recommend to export to cadnano for checking and fine-tuning of their designs. As a matter of fact, the 2D representation of a helix as a double array, popularized by cadnano, remains the most practical for editing. Being able to rotate the helices' representations as in scadnano allows an even more convenient rendering of the design. It is however not possible to faithfully map 3D complex designs to 2D, and, no matter what, some helices that are close to each other in 3D will be mapped at distant locations in a 2D representation. As a result, many crossovers overlay the design and make it confusing to read and edit.

A good compromise would then be to complement a cadnano-like 2D representation of the helices with a 3D editable view of the design. This would allow most of the editing work to be efficiently performed in the 2D interface, while being able to refer to the 3D view to position crossovers between non-parallel helices and evaluate their length.

**Geometry based method for crossover design.** Designs of noticeably high complexity have been made in cadnano. Examples include curved objects [DDS09, HPN<sup>+</sup>11], an origami with two non-parallel layers that assemble in the shape of a perfect square [TLJ<sup>+</sup>17], and structures with corners and kinks [PKZ<sup>+</sup>14, IKJ<sup>+</sup>14].

As we have seen, various methods, software and rules have been developed that allows the design of such complex structures. However, these techniques are still lacking in precision, and going back and forth between design, simulation software and experiment is still the norm.

What we propose here is to deviate from the pattern based approach and instead, to rely on geometry for the positioning of crossovers. To that end, we introduce our new software ENSnano. ENSnano's interface is built upon a sound (as proven by experimental validations throughout the present manuscript) geometric model of DNA. Our goal is to propose a model of DNA that is simple enough to be manipulated through intuitive interfaces, without compromise on precision and fine-tuning.

In Part II, we present ENSnano's interface and how it represents designs made of straight helices. We present ENSnano's features that facilitate the design process and validate our approach by successfully assembling a design made of two non-parallel layers.

In Part III, we develop a new geometric model for curved DNA that allows to generate the positions of nucleotides along curved helices. We also present our new methods for routing curved DNA helices around curved shapes. Our methods, together with ENSnano's interface



that allows crossover locations to be directly deduced from the 3D positions of the nucleotides, enables us to assemble origami with unprecedentedly complex curved shapes.

## Part II

# Sane foundations for a DNA nanostructure design software



## Chapter 4

# Data structure and geometric model for DNA nano-objects

The choice of a data structure to represent a design is essential when it comes to developing a computer assisted design software. In this chapter, we review the data structures used in some other DNA nanostructure design software, and present how designs are represented in ENSnano.

### 4.1 Example of existing data-structures

**Cadnano.** In cadnano, the strands are seen as successions of nucleotides that are located on *virtual double helices*. Each virtual helix consists of two *virtual strands* that serve as support for the real strands of the design. Each virtual strand is a set of positions that is indexed by positive integers. A position on a virtual strand is a potential nucleotide for a real strand. If the position is used by a nucleotide from a real strand, it contains the coordinates (helix index and position on the helix) of the successor (if any) and predecessor (if any) of the nucleotide on that strand.

More precisely, a cadnano design is given by:

- A grid type (squared or honeycomb) that defines a lattice  $\Lambda : \mathbb{N}^2 \rightarrow \mathbb{R}^2$ .
- A list of helices, where each helix is identified by an integer and its position on the grid's lattice.
- For each helix, an upper bound  $N$  on the last used position on the helix, and two arrays called **scaf** (short for “scaffold”) and **stap** (short for “staple”). These arrays correspond to the strands of the helix. If the identifier of the helix is even, the **scaf** array corresponds to the *forward strand* of the helix, *i.e.* the strand where nucleotide index increases along the  $5' \rightarrow 3'$  direction, and the **stap** array corresponds to the *backward strand*, *i.e.* the strand where nucleotide index increases along the  $3' \rightarrow 5'$  direction. For helices with an odd identifier, the correspondence is reversed.

Each cell of the **scaf** and **stap** arrays represent an available position on the virtual strand. For  $0 \leq i \leq N - 1$ , **scaf**[ $i$ ] (*resp.* **stap**[ $i$ ]) is a quadruplet of integers  $(h_5, n_5, h_3, n_3)$ . If the position  $i$  on the scaffold strand (*resp.* staple strand) is not used then all these values are set to  $-1$ . Otherwise,  $(h_5, n_5)$  (*resp.*  $(h_3, n_3)$ ) are the

coordinates (helix index and nucleotide index) of the 5' (*resp.* 3') neighbor of the nucleotide, or  $(-1, -1)$  if the nucleotide is at the 5' (*resp.* 3') end of the strand. Helices have two strands, so this convention may seem ambiguous at first since the strand is not specified. This ambiguity is resolved by the following convention: A nucleotide on a scaffold strand (*resp.* staple strand) is always connected to another nucleotide on a scaffold strand (*resp.* staple strand), even in the case of crossovers.

Figure 4.1 illustrates how the strands of a simple design are represented in cadnano.

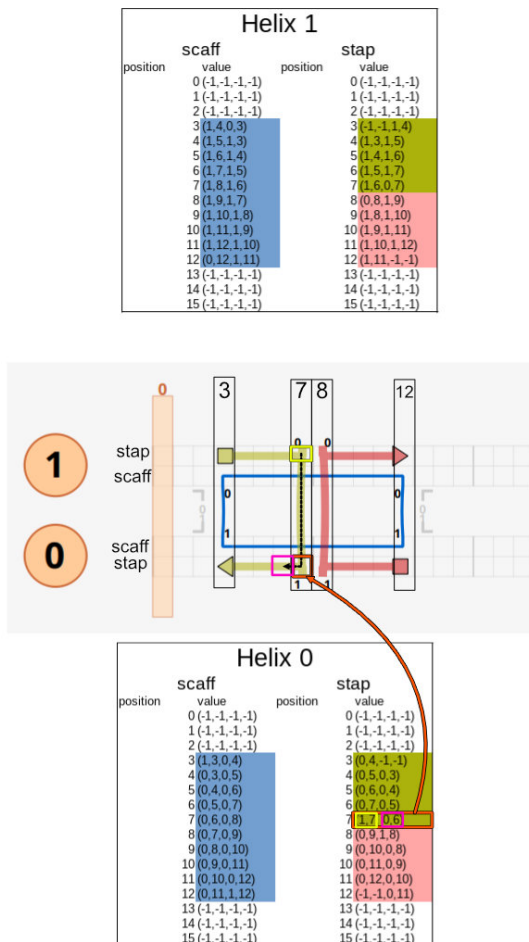


Figure 4.1: **Representation of the strands of a simple design in cadnano.** The design consists in a circular scaffold strand (blue) and two staple strands (yellow and red). The scaffold strand uses nucleotides that are positioned on the `scaff` virtual strands and the staples use nucleotides on the `stap` virtual strands. For a nucleotide at position  $i$  on a virtual strand, the corresponding cell contains the coordinates of the successor and predecessor of the nucleotide on its strand. For example the nucleotide at position 7 on the `stap` virtual strand of helix 1, has its predecessor at position 7 on helix 0 and its successor at position 8 on helix 1.

This data-structure has the advantage of separating the topology of the strands from their geometry. However, it contains a lot of undesirable redundancy (for example, each connection

between two nucleotides is encoded twice), and the convention that maps the `stap/scaf` arrays to forward/backward strands of the helices imposes undesirable limitations. It is for example impossible to create a cyclic strand that goes through an odd number of helices.

**Tiamat.** Tiamat (see Figure 3.2 in Chapter 3) represents designs as an oriented graph. Each vertex of the graph corresponds to a nucleotide in the design, and edges connect vertices that are linked by covalent bonds (neighbors on a strand) or H-bonds (forming a base pair). Each vertex has its own 3D coordinates and can be moved around freely.

This low-level representation is extremely versatile but lacks higher level abstractions that make “bulk” modifications easier.

**Scadnano.** Like `cadnano`, `scadnano` can position helices on a grid. A `scadnano` design also contains a grid type, and a list of helices whose positions may be given by coordinates in the grid’s lattice. `Scadnano` also supports free-positioned helices and the position of a helix can be specified by a point in  $\mathbb{R}^3$  and three angles that define its orientation and tilt.

`Scadnano` also separates the strands’ topology from their geometry. Strands are stored in a separate field and are represented as a list of “domains” that correspond to intervals of indices on a strand of a helix. This notion of domain is a bit different from the one presented in Part I, as there is no notion of a strand’s domain being the exact reverse complement of another one.

`Scadnano`’s data structure offers the same advantages as `cadnano`’s, while fixing some of its limitations. Note that the above description of `scadnano`’s data structure is a simplified one. A more comprehensive description can be found in [DLS20].

## 4.2 ENSnano’s data structure.

As discussed in Section 3.2, we want `ENSnano` to offer both a 3D and a “`cadnano`-like” 2D representation of the design. The 3D view helps to visualize and arrange the elements composing the design in space, while the 2D view allows manipulating and reading the topology of the design. The 2D view is also more readable.

It is therefore natural to opt for a data structure that separates the geometry and the topology of the design. Our approach is similar to (s)`cadnano`’s: the strands composing a nanostructure are positioned on helices that are assumed to have rigid shapes.

**Helices** are virtual support for double-stranded DNA. In `ENSnano`, each helix has two bi-infinite strands, one forward and one backward, that map indices in  $\mathbb{Z}$  to nucleotides. On the *forward strand* (*resp. backward strand*), the nucleotide index increases along the  $5' \rightarrow 3'$  (*resp.  $3' \rightarrow 5'$* ) direction so that facing nucleotides have the same index. For each index  $i \in \mathbb{Z}$ , it is assumed that, if there exist nucleotides at position  $i$  on both the forward and backward strand, these nucleotides are meant to form a base pair.

In this part, we will deal with helices that have the shape of rigid, straight cylinders. We will see in Part III how `ENSnano` extends this data structure to curved DNA helices.

**Grids.** As we have seen, all the helices in a `cadnano` design are positioned on a single grid, allowing the user to create structures made of parallel helices organized in a lattice. This approach can however only be applied to designs where all helices are meant to be parallel.

Scadnano overcomes this limitation by allowing the helices to be freely positioned in space. It should however be noted that scadnano only *stores* this information and does not make use of it for crossover recommendations, nor does it expose the geometry of the design to the user.

While the possibility to position helices freely in space is essential to allow designs to contain non-parallel helices, it is also useful to be able to group helices that are *meant* to be parallel. ENSnano’s approach is to position all straight helices on a grid, but to allow several grids to exist. All the grids can have different orientations which makes it possible to create designs in which helices point in different directions.

When designing a 3D origami, a classic technique consists in dividing the design into components made of parallel helices (e.g. [GWND15, TLJ<sup>+</sup>17, MAM<sup>+</sup>22]). In ENSnano, this technique can be implemented by grouping together on a grid the helices that constitute a component. Each component attached to a grid can be thought of as a separated cadnano design. As discussed in Chapter 3, an abstract 2D representation of these groups of parallel helices is often the most ergonomic to edit these components. However, when connecting together multiple components, a 3D editable view of the design becomes useful. To provide such an interface, it is necessary to be able to generate 3D positions for the nucleotides of the strands constituting the design. These positions will depend on the origin and orientation of the helix on which the nucleotides lie.

**Origin and orientation of grids and helices.** Each grid  $\mathcal{G}$  is given:

- An origin  $\mathcal{O}_{\mathcal{G}}$ .
- A frame  $\mathcal{F}_{\mathcal{G}} \in \text{SO}_3(\mathbb{R})$ . Its  $\mathbf{x}$  vector defines the orientation of the helices, and its  $YZ$  plane its tilt and where the helices will be positioned.
- A lattice  $\lambda_{\mathcal{G}} : \mathbb{Z}^2 \rightarrow \mathbb{R}^2$ . For convenience, since this lattice serves to position the helices in the  $ZY$  plane, we embed this lattice in space by defining

$$\Lambda_{\mathcal{G}} : \mathbb{Z}^2 \rightarrow \mathbb{R}^3$$

$$(i, j) \mapsto \begin{pmatrix} 0 \\ (\lambda_{\mathcal{G}}(i, j))_y \\ (\lambda_{\mathcal{G}}(i, j))_x \end{pmatrix}.$$

Together these attributes define a function  $\mathcal{G} : \mathbb{Z}^2 \rightarrow \mathbb{R}^3 \times \text{SO}_3(\mathbb{R})$  that maps a position on the grid’s lattice to a position and orientation (Figure 4.2). This function is defined by

$$\mathcal{G}(i, j) = (\mathcal{O}_{\mathcal{G}} + \mathcal{F}_{\mathcal{G}} \cdot \Lambda_{\mathcal{G}}(i, j), \mathcal{F}_{\mathcal{G}}). \quad (4.1)$$

### 4.3 P-stick model for straight DNA helices

A straight DNA double helix  $H$  can be seen as an infinite cylinder characterized by the position  $\mathcal{O}_H$  of the origin of its axis and a frame  $\mathcal{F}_H = (X_H, Y_H, Z_H) \in \text{SO}_3(\mathbb{R})$  that defines its orientation. For a helix at position  $(i, j)$  on a grid  $\mathcal{G}$ , its position and orientation are given by

$$(\mathcal{O}_{\mathcal{H}}, \mathcal{F}_{\mathcal{H}}) = \mathcal{G}(i, j).$$

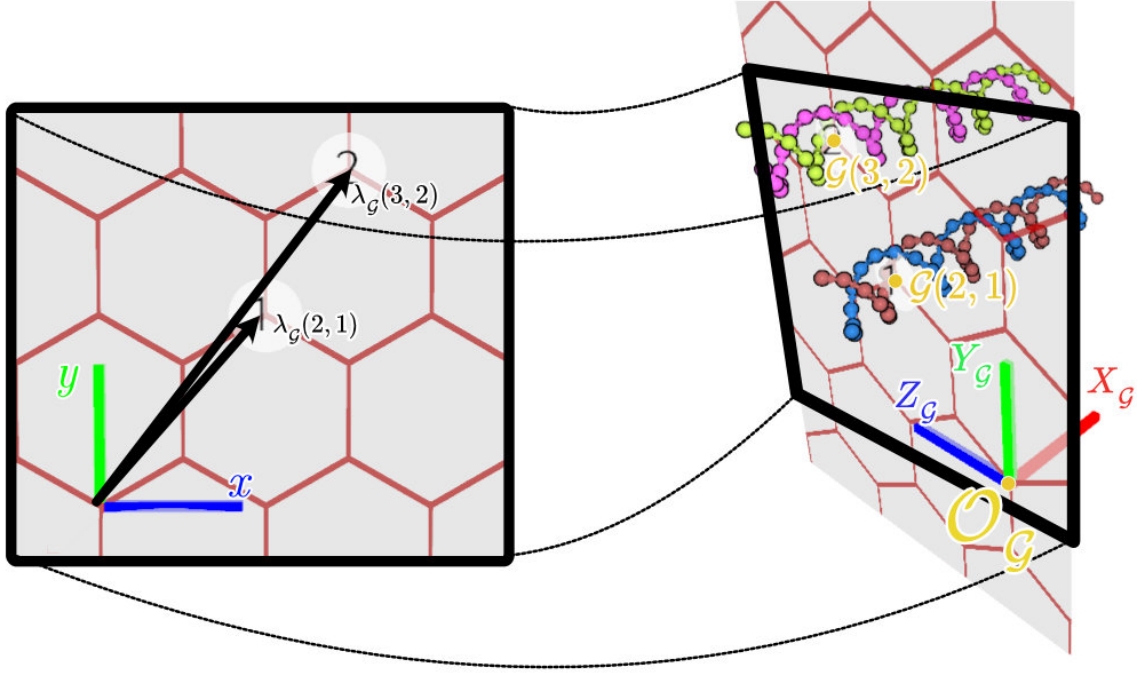


Figure 4.2: **Mapping positions on a grid lattice to positions and orientations in space.** The grid is equipped with a lattice  $\lambda_G$ , in that case a honeycomb lattice. When the lattice is instantiated as a grid  $\mathcal{G}$  with origin  $\mathcal{O}_G$  and a frame  $(X_G, Y_G, Z_G)$ , the lattice is embedded in the  $Z_G Y_G$  plane (the  $X_G$  axis corresponds to the axis of the helices on the grid).

The helix is also given an additional *Roll* parameter  $\rho$  that is the value of the angle  $\widehat{X_H O_H N_0^f}$  (Figure 4.3). Where  $N_0^f$  is the position of the phosphate of the nucleotide at position 0 on the forward strand of the helix. This additional parameter  $\rho$  allows different helices that belong to the same grid  $\mathcal{G}$  to share a single frame  $F_G$  while having distinct roll values.

The *P-stick* model [GA14] generates the positions of the Phosphate atoms (hence its name) on a straight DNA double helix. The parameters of the model are:

- The *rise*  $\Delta$  that is the distance between the projections of two successive nucleotides on  $H$ 's axis.
- The *helicity*  $\hbar$  that is expressed in (fractional) number of base pairs in a full turn of the double helix. Alternatively we can use the *twist* parameter  $\beta$  that is the angle between two consecutive nucleotides. The twist and helicity are connected by the formula  $\beta = \frac{2\pi}{\hbar}$ .
- The *radius*  $R$  of the helix.
- The *minor groove angle*  $\alpha$  that is the oriented angle between the nucleotide of the forward strand and the nucleotide of the backward strand. More precisely, it is the angle between their projection in a plane orthogonal to the axis of the helix.



- The *inclination*  $I$  that is the distance between the projections on  $H$ 's axis of the nucleotide from the forward strand and the nucleotide of the backward strand at a given position.

In the P-stick model, each nucleotide lies on a circle of radius  $R$ , parallel to the plane  $X_H Y_H$  and centered on the axis of the helix (Figure 4.3). In other words, its position is of the form

$$\mathcal{O}_H + \mathcal{F}_H \begin{pmatrix} R \cos(\theta) \\ R \sin(\theta) \\ \text{total rise} \end{pmatrix}$$

Where the total rise increases by steps of  $\Delta$  at each position and is shifted by  $I$  for nucleotides of the backward strand, while the angle  $\theta$  is initialized at  $\rho$ , increases by steps of  $\beta$  and is shifted by  $\alpha$  for nucleotides of the backward strand.

In summary, in the P-stick model, the positions  $N_i^f$  (resp.  $N_i^b$ ) of the  $i$ -th nucleotide of the forward (resp. backward) strand of the helix are identified with the positions of their Phosphate atom and given by the following formulas.

$$N_i^f = \mathcal{O}_H + \Delta \cdot i Z_H + R X_H \cos(\rho + \beta \cdot i) - R Y_H \sin(\rho + \beta \cdot i) \quad (4.2)$$

$$N_i^b = \mathcal{O}_H + (\Delta \cdot i + I) Z_H + R X_H \cos(\rho + \beta \cdot i + \alpha) - R Y_H \sin(\rho + \beta \cdot i + \alpha) \quad (4.3)$$

In ENSnano, custom values of the geometric parameters of the P-stick model can be specified in the design's `.ens` file. The default values for the parameters of the P-stick model are given in Table 4.1 and illustrated in Figure 4.3.

Radius	$R = 0.93 \text{ nm}$
Rise	$\Delta = 0.332 \text{ nm}$
Inclination	$I = 0.375 \text{ nm}$
Narrow groove angle	$\alpha = 170.4^\circ$
Twist	$\beta = 34.48^\circ$
Helicity	$\hbar = 10.44 \text{ bp/turn}$

Table 4.1: Default values for the geometric parameters of the P-stick model in ENSnano. These values are taken from [GA14, WR11]

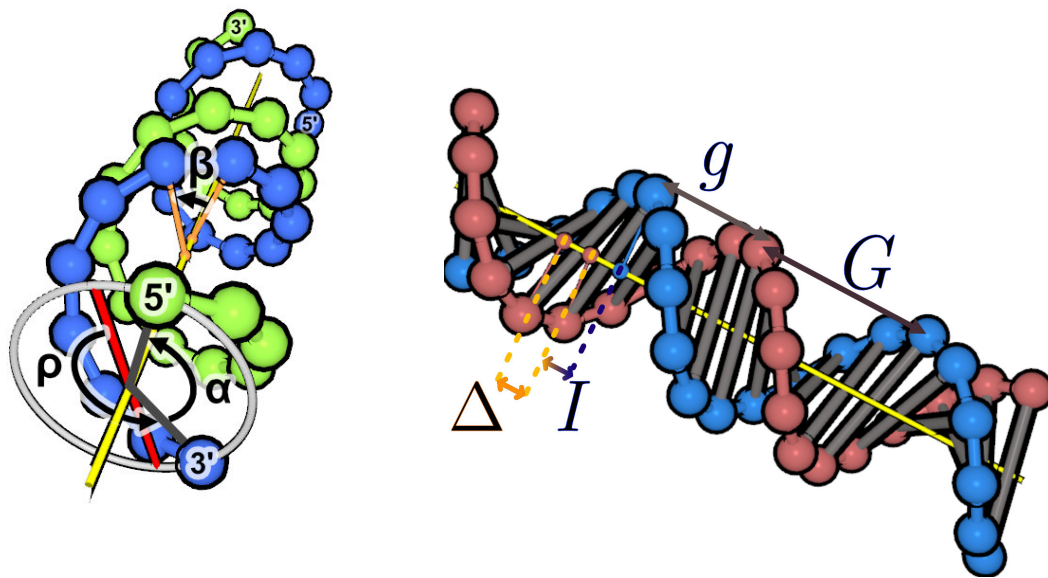


Figure 4.3: **The geometric parameters of the P-stick model**  $\alpha$ : Narrow groove angle  $\beta$ : Angle between two consecutive base pairs. This angle can be obtained by the formula  $\beta = \frac{2\pi}{h}$ ;  $\Delta$ : Distance in nanometers between two consecutive bases along the axis;  $G$ : Length of the major groove;  $g$ : Length of the minor groove;  $G$ ,  $g$  and  $\alpha$  are related to each other by the formula  $\alpha = 2\pi \frac{g}{G+g}$ .



## Chapter 5

# Physical model for automatic roll, stability testing and design import

### 5.1 Motivations and scope

**Automatic roll optimization.** Rolling a helix around its axis shifts its strands forward or backward and thus has a huge impact on the positions at which crossovers can be made with neighboring helices. Reciprocally, placing some crossovers on a helix will have an impact on its optimal roll which will in turn impact all the surrounding crossovers.

Choosing the roll of the helices gives more freedom in a design, and improves its feasibility. It may happen that a design with seemingly poorly-placed crossover can be “fixed” by simply modifying the roll of the helices, without modifying the design’s topology. Note that, in that case, the design is not really modified: its 3D representation is simply made more faithful.

An automatic optimization of the roll parameters of all the helices of the design can constitute a quick check of the feasibility of the design: If all the crossovers are well-placed, there should exist a set of roll parameters for which all the crossovers have reasonable apparent lengths, and this set of parameters should be found by a roll optimization process.

Conversely, if optimizing the rolls fails to reduce the length of some crossovers, this may indicate that some crossovers are not coherently positioned.

**Rigid body simulations.** As we saw in Chapter 3, there exist physical simulation software that are used to validate designs by assessing their feasibility. These software typically simulate intricate physical processes and are computationally expensive to run. While they are able of providing pretty accurate structure predictions, they are usually used only at the end of the design process. Moreover, because each simulation requires a long computation time, going back-and-forth between design and simulation software is time-consuming.

In ENSnano, we want to provide an integrated, lightweight physics engine that can be used regularly to get quick feedbacks during the design process. The goal is not to provide realistic thermodynamics simulations or to predict the shape of the design, but rather to evaluate the stability of the structure as it is currently displayed in the 3D view.

We also want to use our physics engine to facilitate the import of designs from cadnano, which have no explicit 3D embedding. The goal is to recover the relative positions of the different components of a design just from its topology (*i.e.* crossover pattern). Note that other software such as OxView [BMP<sup>+</sup>22] also rely on a physics engine to infer the geometry

of cadnano designs before running the simulation.

## 5.2 Automatic roll optimization

In our automatic roll optimization system, DNA helices are treated as rigid objects and each crossover is treated as an ideal spring with equilibrium length  $\ell_0 = 0.7\text{nm}$ .

If there are  $n$  helices in the design, the state of our system is simply given by the rolls of all the helices:  $X = (\rho_1, \dots, \rho_n)$ .

For each crossover that connects two nucleotides  $N_1$  and  $N_2$  we write  $X(N_1)$  and  $X(N_2)$  the positions of  $N_1$  and  $N_2$  in state  $X$ , and  $X(N_1, N_2)$  the distance between  $X(N_1)$  and  $X(N_2)$ . Our goal is to minimize the value

$$\phi(X) = \sum_{(N_1, N_2)} (X(N_1, N_2) - \ell_0)^2.$$

To do so, we simulate a physics system were the linear springs are only allowed to act on the roll of the helices. We consider that any force  $F$  applied by a crossover to a helix is compensated by a reaction force  $-F$  exerted on the helix by its own axis. Together, the crossover's force and the reaction result in a *force couple* that induces an angular acceleration of the helix.

A crossover that connects the nucleotides  $N_a$  and  $N_b$  on the helices  $H_i$  and  $H_j$  applies on  $H_i$  a force

$$F_i^{N_a, N_b} = k_{\text{crossover}} \left( 1 - \frac{\ell_0}{X(N_a, N_b)} \right) \overrightarrow{X(N_a)X(N_b)}, \quad (5.1)$$

where  $k_{\text{crossover}}$  is a customizable parameter of the simulation that can be set by the user.

This force exerts on  $H_i$  a torque

$$\tilde{\tau}_i^{N_a, N_b} = \overrightarrow{X(N_a)O_a} \wedge F_i^{N_a, N_b}, \quad (5.2)$$

where  $O_a$  is the projection of  $X(N_a)$  on  $H_i$ 's axis and  $\wedge$  is the vector product in  $\mathbb{R}^3$ :

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \wedge \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_1 z_2 - z_1 y_2 \\ z_1 x_2 - x_1 z_2 \\ x_1 y_2 - y_1 x_2 \end{pmatrix}.$$

The helix is only allowed to rotate around its own axis, we model this by saying that the reaction of the axis induces a torque that cancels out the part of  $\tilde{\tau}_i^{N_a, N_b}$  that is not parallel to the axis of the helix. To sum up, the action of the crossover  $(N_a, N_b)$  on helix  $H_i$  together with the axis' reaction to that action induce a torque equal to

$$\begin{aligned} \tau_i^{N_a, N_b} &= \langle \tilde{\tau}_i^{N_a, N_b} | \mathbf{x}_i \rangle \mathbf{x}_i \\ &= R k_{\text{crossover}} \left( 1 - \frac{\ell_0}{\ell_{X(N_a, N_b)}} \right) \langle \overrightarrow{X(N_a)X(N_b)} | \mathbf{x}_i \rangle \mathbf{x}_i, \end{aligned} \quad (5.3)$$

where  $\mathbf{x}_i$  is the direction of the axis of helix  $H_i$ , and  $R$  is the radius of the helix. Note that Equation (5.3) holds because  $\mathbf{x}_i$  is orthogonal to  $\overrightarrow{X(N_a)O_a}$ .

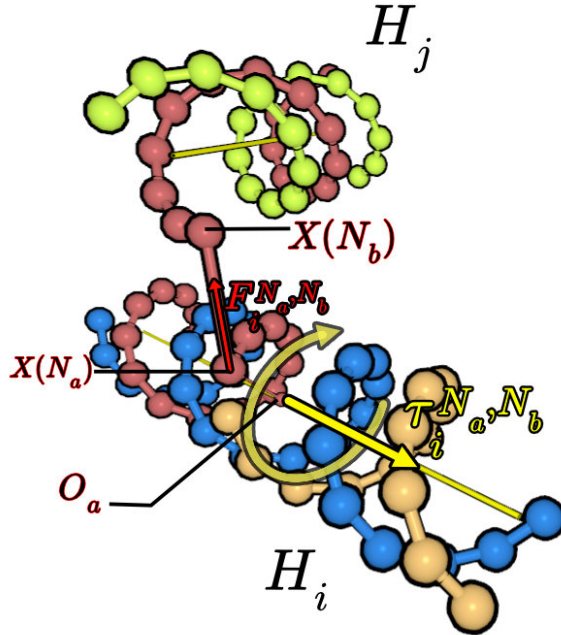


Figure 5.1: **Torque induced on a helix by a crossover.** The crossover  $(N_a, N_b)$  exerts a force  $F_i^{N_a, N_b}$  on helix  $H_i$ . This force is compensated by the reaction of the axis of the helix. Together, the force induced by the crossover and the reaction of the axis induce a torque  $\tau_i^{N_a, N_b}$  that “pulls”  $N_a$  towards  $N_b$  by rotating the helix.

In order to prevent oscillations, we consider that, in addition to the forces exerted by the crossovers, fluid friction is also applied to the helices. The friction applied to helix  $H_i$  induces a torque equal to

$$\tau_i^{\text{frict}} = -k_{\text{frict}} \dot{\rho}_i \mathbf{x}_i,$$

where  $k_{\text{frict}}$  is a customizable parameter of the simulation that can be set by the user.

If we denote by  $C_i$  the set of crossovers that involve helix  $i$ , the dynamics of our system is given by

$$\ddot{\rho}_i = \frac{1}{m} \sum_{N_a, N_b \in C_i} \left( \langle \tau_i^{N_a, N_b} | \mathbf{x}_i \rangle \right) + \langle \tau_i^{\text{frict}} | \mathbf{x}_i \rangle, \quad (5.4)$$

where  $m$  is a mass that we assume to be the same for all helices and that can be set by the user. To compute the trajectory of our system we simply apply Euler’s method: We chose a time step  $dt$  and iterate the following procedure:

- For all  $i$ ,  $\rho_i \leftarrow \rho_i + dt \dot{\rho}_i$
- For all  $i$ ,  $\dot{\rho}_i \leftarrow \dot{\rho}_i + dt \ddot{\rho}_i$
- For all  $i$ , compute  $\ddot{\rho}_i$  according to Equation (5.4).

**Results.** In our day-to-day usage, we find that the automatic roll optimization is one of ENSnano’s most useful features. The simulation reaches equilibrium almost instantaneously and is therefore a convenient way to run a “sanity check” on a design.

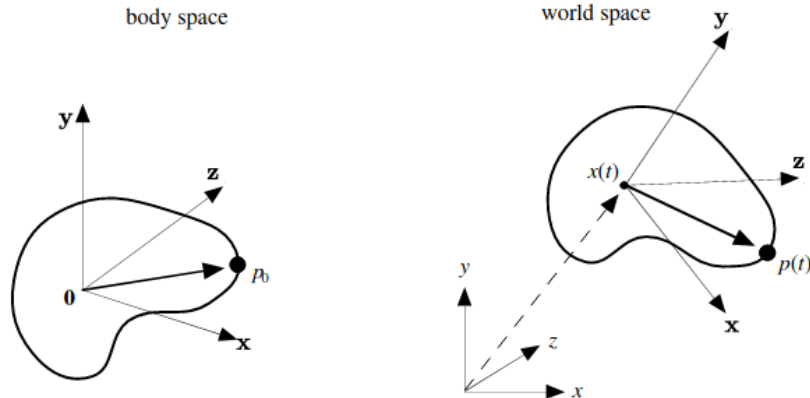


Figure 5.2: **Description of the position of a point in a rigid body.** All point  $p$  of the rigid body has a fixed position  $p_0$  in the body’s referential. The whole rigid body is given a moving referential  $(x(t), R(t))$  (with  $R(t) = (x'(t), y'(t), z'(t))$ ) and the position of a point  $p$  at time  $t$  in the world’s referential is given by  $p(t) = p_0R(t) + x(t)$ . Figure extracted from [Bar01].

This feature is also invaluable when importing a design made in cadnano (Figures 5.4a and 5.5a).

### 5.3 Simulating rigid body dynamics

The implementation of our physics engine is based on a course by David Baraff on rigid body dynamics simulation [Bar01].

**Position of the points of a rigid body.** A rigid body  $B$  can be defined as a set of points  $X_B \subset \mathbb{R}^3$ , and can be equipped with a moving referential  $(x(t), R(t)) \in (\mathbb{R}^3, \text{SO}_3(\mathbb{R}))$  so that  $x(t)$  is the position of the body’s center of mass, and for all points  $p \in X_B$ , there exists a vector  $p_0 \in \mathbb{R}$  so that the position  $p(t)$  at time  $t$  is given by

$$p(t) = p_0R(t) + x(t), \quad (5.5)$$

as illustrated by Figure 5.2. We say that  $p_0$  is the position of  $p$  in the body’s referential. Equation (5.5), along with the constraint that for all time  $t$ ,  $R(t) \in \text{SO}_3(\mathbb{R})$ , is what defines a *rigid* body.

Our goal is to compute the trajectory of a rigid body. That is to say, to compute the functions  $x(t)$  and  $R(t)$ . To do so, our strategy is to express  $x$  and  $R$  as solutions of an ordinary differential equation (ODE), and to compute their trajectory using a third party ODE solver.

The goal of this subsection is to establish these equations.

**Velocities of a rigid body** The *linear velocity*  $v$  of a rigid body is the derivative of its position with respect to time:

$$v(t) = \dot{x}(t). \quad (5.6)$$

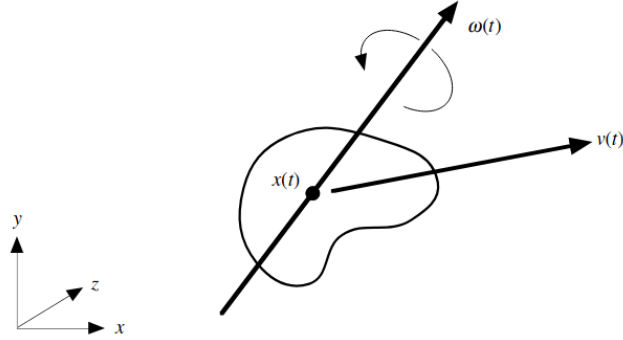


Figure 5.3: **Linear velocity and spin of a rigid body.** The linear velocity  $v$  corresponds to the translation movement of the center of mass. The direction of the spin vector  $\omega$  gives the axis around which the body is rotating. The length of  $\omega$  corresponds to the speed of rotation. Figure extracted from [Bar01].

The dynamics of the body's orientation is described by its *angular velocity* or *spin*. The spin of a rigid body can be described by a vector  $\omega(t)$ . When  $\omega(t)$  is drawn as passing by the center of mass (Figure 5.3), its direction corresponds to the axis around which the body is spinning, while its length corresponds to the speed of the body's rotation.

More formally, for a given point  $p$  of the body, if we define  $r_p(t)$  to be the vector from the center of mass of  $B$  to  $p$ :

$$r_p(t) := p(t) - x(t) = p_0 R(t), \quad (5.7)$$

then the derivative of  $r_p$  is related to  $\omega(t)$  by the formula

$$\dot{r}_p(t) = \omega(t) \wedge r_p(t). \quad (5.8)$$

The relation between the derivative of the matrix  $R$  and  $\omega$  can be derived by applying Equation (5.8) to the axes of  $R = (\mathbf{x}, \mathbf{y}, \mathbf{z})$ :

$$\dot{R}(t) = (\omega(t) \wedge \mathbf{x}(t), \omega(t) \wedge \mathbf{y}(t), \omega(t) \wedge \mathbf{z}(t)) \quad (5.9)$$

**Inertia and momentums of a rigid body.** The linear and angular momentums of a rigid body are important quantities because they are the ones whose derivatives can be directly computed by applying the laws of motion.

The *linear momentum* of a rigid body is the quantity  $\rho(t) = mv(t)$  where  $v(t)$  is the linear velocity and  $m$  is the mass of the body.

In order to define the angular momentum of a rigid body, one first needs to define the inertia matrix. The inertia matrix  $I \in \mathcal{M}_3(\mathbb{R})$  of  $B$  represents how its mass is distributed around its center of mass and is given by

$$I(t) = m \int_{p \in X_B} \begin{pmatrix} r_p(t)_y^2 + r_p(t)_z^2 & -r_p(t)_x r_p(t)_y & -r_p(t)_x r_p(t)_z \\ -r_p(t)_x r_p(t)_y & r_p(t)_x^2 + r_p(t)_z^2 & -r_p(t)_y r_p(t)_z \\ -r_p(t)_x r_p(t)_z & -r_p(t)_y r_p(t)_z & r_p(t)_x^2 + r_p(t)_y^2 \end{pmatrix} d^3 p. \quad (5.10)$$

The *angular momentum*  $L$  of a rigid body is then given by

$$L(t) = I(t)\omega(t). \quad (5.11)$$



**Laws of motion and dynamics of a rigid body.** Let  $(F_i(t))_i$  be the forces that are applied on  $B$ . If the force  $F_i$  is applied on the point  $p_i$ , then the associated *torque* is given by

$$\tau_i(t) = r_{p_i}(t) \wedge F_i(t). \quad (5.12)$$

We are now ready to state the laws of motions:

**Theorem 1** (Laws of motion). The derivative of  $\rho$  and  $L$  are given by

$$\dot{\rho}(t) = \sum_i F_i(t) \quad (5.13)$$

$$\dot{L}(t) = \sum_i \tau_i(t). \quad (5.14)$$

This means that we can now fully express the dynamics of the body with a differential equation by writing

$$\mathbf{X}(t) = \begin{pmatrix} x(t) \\ R(t) \\ \rho(t) \\ L(t) \end{pmatrix},$$

which gives

$$\begin{aligned} \dot{\mathbf{X}}(t) &= \begin{pmatrix} \frac{\rho}{m} \\ \omega(t)^* R(t) \\ \sum_i F_i(t) \\ \sum_i \tau_i(t) \end{pmatrix} \\ &= \begin{pmatrix} \frac{\rho}{m} \\ (I^{-1}(t)L(t))^* R(t) \\ \sum_i F_i(t) \\ \sum_i \tau_i(t) \end{pmatrix}. \end{aligned} \quad (5.15)$$

Where, for a vector  $\mathbf{a} \in \mathbb{R}^3$ ,  $\mathbf{a}^* = \begin{pmatrix} 0 & -\mathbf{a}_z & \mathbf{a}_y \\ \mathbf{a}_z & 0 & -\mathbf{a}_x \\ -\mathbf{a}_y & \mathbf{a}_x & 0 \end{pmatrix}$  is the matrix so that for all  $\mathbf{b} \in \mathbb{R}^3$ ,  $\mathbf{a}^* \mathbf{b} = \mathbf{a} \wedge \mathbf{b}$ .

In order to compute efficiently  $I^{-1}(t)$ , we notice that

$$I(t) = R(t)I(0)R^{-1}(t),$$

which gives

$$I^{-1}(t) = R(t)I^{-1}(0)R^{-1}(t).$$

Therefore, we only need to compute the first inertia matrix  $I(0)$  and its inverse. Moreover, inverting  $R(t)$  is easy: since  $R(t) \in \text{SO}_3(\mathbb{R})$ ,  $R^{-1}(t) = R^T(t)$ .

## 5.4 Rigid body simulations in ENSnano

ENSnano offers two kinds of rigid body simulations. In the first system, the rigid bodies are the double DNA helices used in the design. Each of them is treated as an independent rigid

cylinder, and we treat crossovers between helices as linear springs. This system is meant to be used for quickly assessing the stability of the design.

In the second system, the helices are fixed on their grid positions and only the grids are allowed to move. The rigid bodies in that system are therefore the sets of all the helices attached to each grid. By running a simulation of this system, we can guess the intended relative positions of the different grids of a design. This is useful for importing designs from software like cadnano that do not incorporate information about geometry in their file format.

**System of independent rigid helices.** To apply the method described in Section 5.3, we need to compute the inertia matrices of the elements of our system. Fortunately, this is easy to do for rigid cylinders: The inertia matrix of a cylinder with length  $\ell$ , radius  $r$ , volumetric density  $m$ , centered at the origin and whose axis is parallel to the  $x$  axis is

$$I_{\text{cylinder}} = \begin{pmatrix} \frac{r^4 h}{2} & 0 & 0 \\ 0 & r^2 h \left( \frac{r^2}{4} + \frac{h^2}{12} \right) & 0 \\ 0 & 0 & r^2 h \left( \frac{r^2}{4} + \frac{h^2}{12} \right) \end{pmatrix}. \quad (5.16)$$

The forces and torque that are applied to the helices are the same as in the automatic roll optimization system, and their values are given by Equations (5.1) and (5.2).

The difference is that in this system, the movement of the helices is unconstrained and therefore the influences of the crossovers are not compensated by reaction forces from the axes.

We tested our rigid helices simulation engine on both versions of the rectangle design of [WR11] (see Figure 5.4 for the simulation results and Figure 2.4 in Chapter 2 for a description of the designs). When running the simulation on the version of the design that resulted in twisted origami (left half of Figure 5.4c), the helices crash into each other. This kind of behavior qualifies as a failure according to the quick stability test. Note that our engine is not able to predict the actual twisted shape of the design, however it can say that its representation as a flat rectangle is not realistic. When running the simulation on the corrected design that assembled into a flat origami (right half of Figure 5.4c), the helices stay still, meaning that this design passes ENSnano's quick stability test.

**System of rigid components attached to grid.** In order to run our simulation engine on objects made of all the helices attached to a grid, we need to compute the inertia matrices of these objects. The first step is to compute the center of mass of the object. The center of mass of a component  $C$  made of the helices  $H_1, H_2, \dots, H_n$  is given by the weighted sum

$$\mathcal{O}_C = \sum_{k=1}^n \frac{h_k}{S} \mathcal{O}_k \quad (5.17)$$

where  $h_k$  is the height of helix  $k$ ,  $\mathcal{O}_k$  is its center of mass, and  $S = \sum_{i=1}^k h_k$  is a normalization factor. The contribution of each helix to the total inertia of the component equal to the inertia of the helix when taking into account its position relative to the center of mass of the component:

$$I_k = I_{\text{cylinder}} + h_k \begin{pmatrix} r_{k,y}^2 + r_{k,z}^2 & -r_{k,x}r_{k,y} & -r_{k,x}r_{k,z} \\ -r_{k,x}r_{k,y} & r_{k,x}^2 + r_{k,z}^2 & -r_{k,y}r_{k,z} \\ -r_{k,x}r_{k,z} & -r_{k,y}r_{k,z} & r_{k,x}^2 + r_{k,y}^2 \end{pmatrix}$$

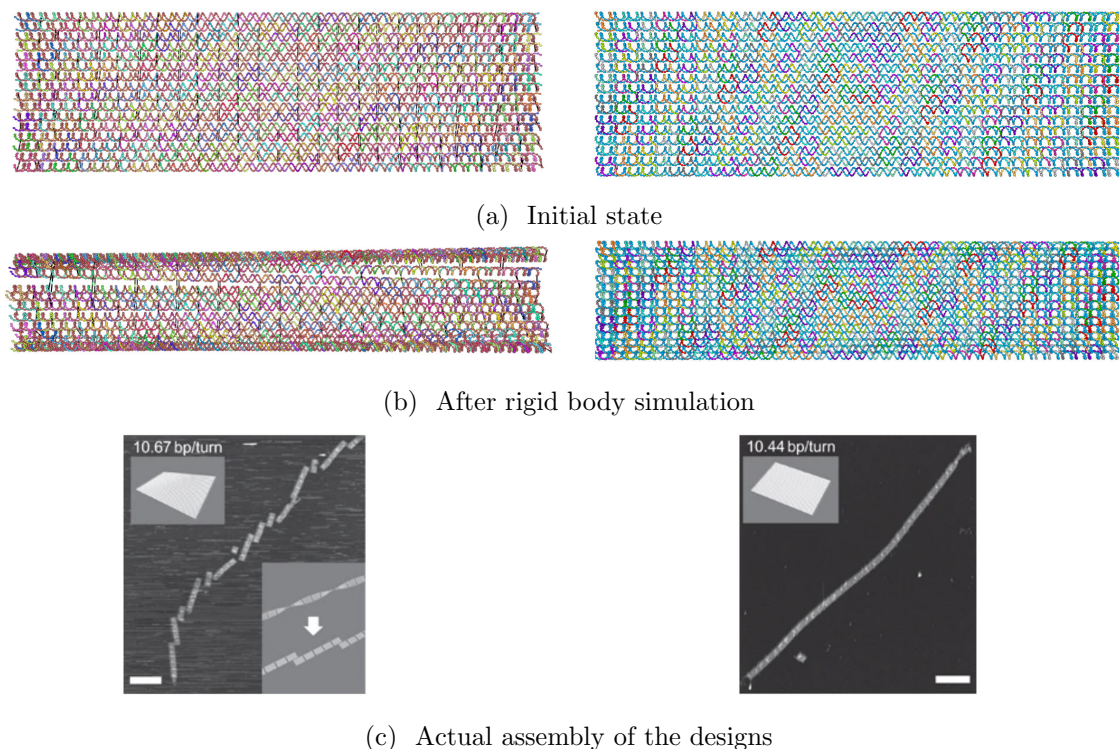
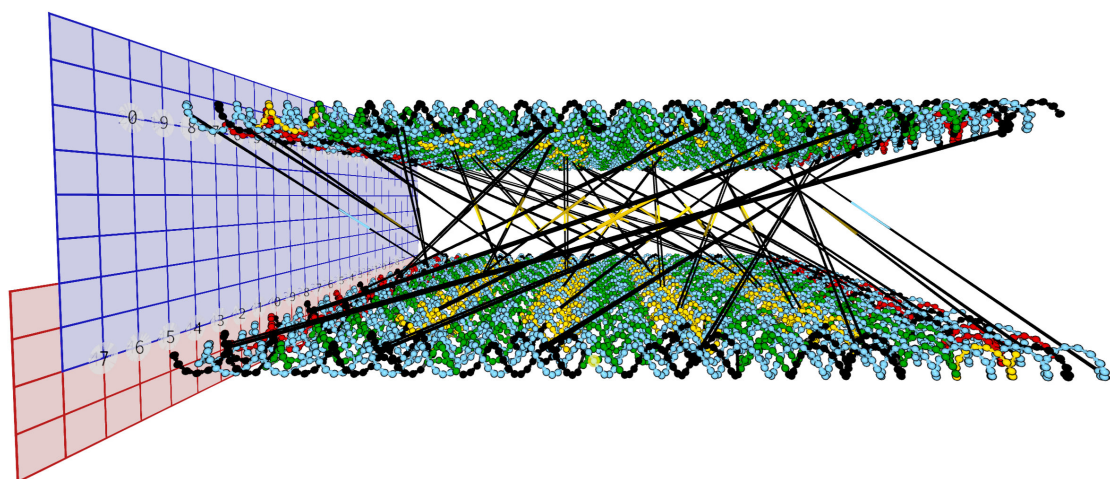


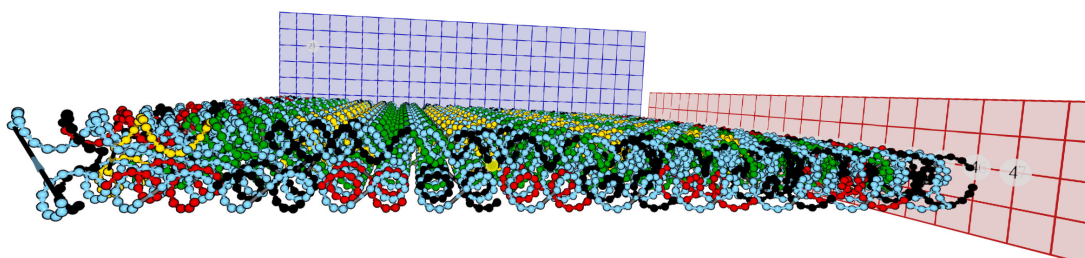
Figure 5.4: **Rigid body simulation on the helices of the rectangle origami designs from [WR11]**. a: Designs after import from cadnano and automatic roll optimization. b: Designs after a few steps of rigid body simulations. c: Actual assembly of the origami, as presented in [WR11]. See also Figure 2.4 in Chapter 2 for a description of the designs.

where  $r_k = \begin{pmatrix} r_{k,x} \\ r_{k,y} \\ r_{k,z} \end{pmatrix} = \mathcal{O}_k - \mathcal{O}$  is the vector from the center of mass of the component to that of the helix.

In this system we consider only the crossovers between helices that are attached to different grids. The forces exerted on by the crossovers on the grids is the same as in the other systems and is given by Equation (5.1). The torque of these forces however is computed relatively to the center of mass of the whole component. We tested this simulation engine on the double layer origami design from [TLJ<sup>+</sup>17]. This design consists of two orthogonal layers of helices and was originally designed in cadnano. Our simulation engine is capable of retrieving the fact that the two layers are supposed to be orthogonal. However since it does not enforce volume exclusion, the two layers end up slightly superposed and manual adjustment of their position is still required after the simulation. In addition to enforcing volume exclusion this system would benefit from performing roll optimizations during the simulation, which it currently does not.



(a) As imported from cadnano



(b) After rigid body simulation on the grid components

Figure 5.5: **Using rigid body simulator to position grids in ENSnano.** a: Double layer origami from [TLJ<sup>+</sup>17] as imported from cadnano. Cadnano files do not have information about geometry so the grids are imported as parallel. b: Position of the grids after rigid body simulation. Some helices are intersecting each other because ENSnano's rigid body engine does not enforce volume exclusion.



## Chapter 6

# An efficient user interface for the conception of DNA nanostructures

In this chapter, we present ENSnano’s interface for designing DNA nanostructures. We also describe some of ENSnano’s original features and explain how to perform the most frequent edit operations with our software. This chapter is mostly adapted from our paper introducing ENSnano at the 27th international conference on DNA computing and molecular programming [LS21].

### 6.1 Overview of ENSnano’s interface

The interface of ENSnano is designed so that the most common actions are only one click away: editing either grids, helices, strands crossovers, 5’ or 3’ ends of domains, . . . can all be performed from the normal action mode. The interface is divided in four area (Figure 6.1):

- The top bar (1) contains buttons to open/save files, to organize the 2D and 3D view, to undo/redo operations and to change the selection/action mode.
- The Left panel (2-5) contains GUI elements. It is itself divided into four regions (see next paragraph).
- The main view area (6). By default, both the 3D and 2D views are displayed in that area, but it is also possible to focus on only one of them to gain space.
- The bottom bar (7) is used to display information about the current action during edition.

**The Left panel** is divided into four regions.

The *camera* panel (3) gathers buttons that set the camera in standard positions, which is useful to align the design with the axis.

The *contextual* (4) panel displays either a text summarizing how to perform basic actions in ENSnano, or information about the current selection.

The *organizer* (5) allows elements of the design to be gathered into named groups. These groups can be used to quickly select one part of the design, or to adjust altogether the properties of the elements in the set. Groups can for instance be used to quickly enable

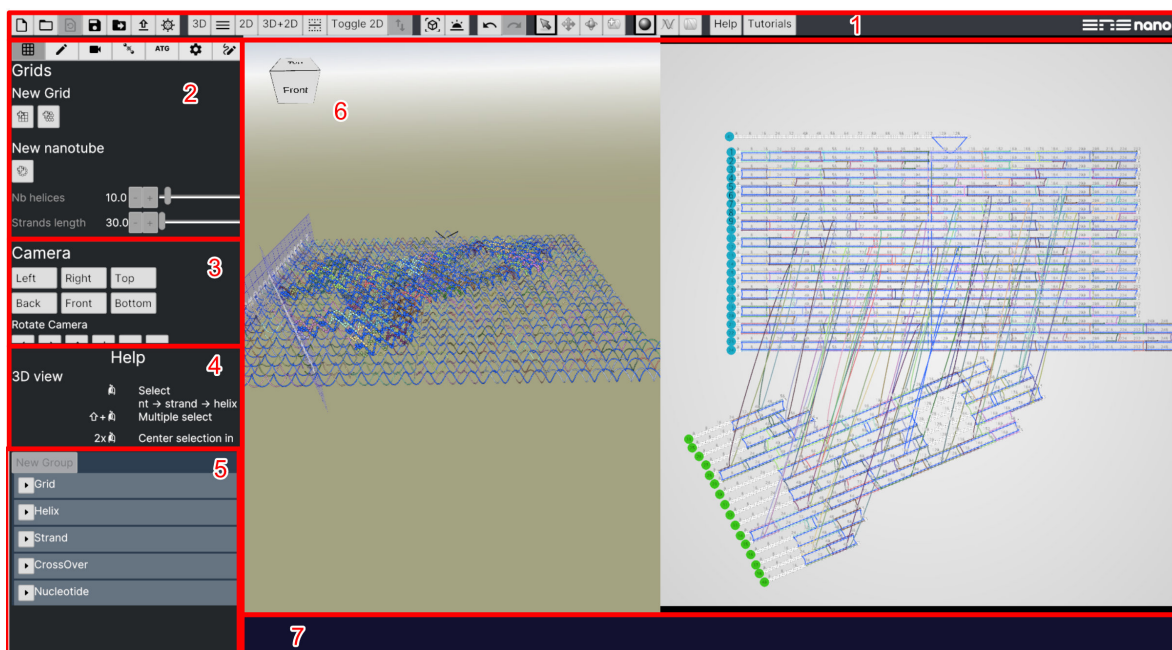







Figure 6.1: **Overview of ENSnano’s interface.** 1: Top bar. 2: Tool panel. 3: Camera shortcut panel. 4: Contextual panel. 5: Organizer. 6: Main 2D/3D views. 7: Status bar.

crossover suggestions between the helices of two groups. They can also be used to temporarily hide or show parts of the design. Finally, when staple strands are placed in named groups, these will appear in the export format. This can be useful when making modular designs, as it allows to quickly identify the components to which each strand belongs.

Note that the group structure in ENSnano does not require the groups to be disjoint. This is typically useful in the case where one wants to visualize the interface between two components linked together by a set of crossovers. To do so, one can create two groups, each of them containing one of the components and the set of crossover. Using these groups, one can choose to hide everything in a design but one of the components and the crossovers at the interface (Figure 6.2).

The tool panel is composed of eight tabs:

- The grid tab  gathers the tools to create grids and add helices to them.
- The edit tab  gathers the tools to edit nucleotides, strands and helices.
- The camera tab  presents the visualization parameters.
- The rigid body engine tab  presents the physics simulation tools presented in Chapter 5.
- The sequence tab **ATG** is used to set the sequence of the scaffold and export the sequences of the staples of an origami.
- The parameter tab  allows to change some parameters like font size and scrolling sensitivity.



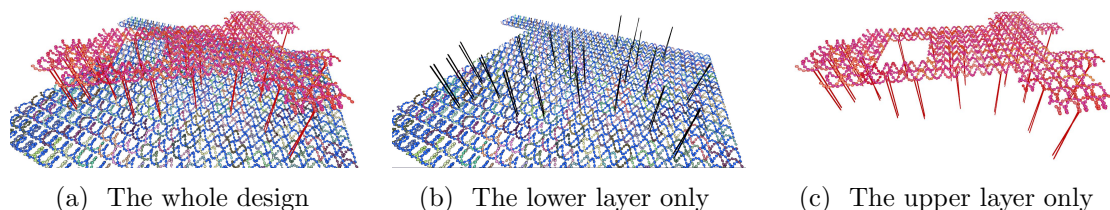














Figure 6.2: **Using ENSnano’s groups to hide elements of the design.** a: The two layers of a rocket design which are spread apart to ease the reading of the crossovers in the figures. Two overlapping heterogeneous groups, gathering helices and crossovers, have been created in the organizer, and can be shown or hidden on demand in the 3D view. b: A first group gathering the helices in the rectangular base, and the crossovers between the two layers. c: A second group gathering the helices of the rocket and the crossovers between the two layers.

- The Bézier path tab  allows creating and editing Bézier paths (see Section 9.2 in Chapter 9).
- The revolution tab  allows creating twisted revolution surfaces (see Chapter 11).

**Action and selection modes.** Objects are selected by clicking on them in one of the view. What is selected determined by the *selection mode*. The selection modes are: nucleotides , strands  and helices .

The action mode determine how the user interacts with the design. The most common editing operations are done in the *normal mode* . The *translation mode*  allows to translate the selected object and the *rotation mode*  allows to rotate it. The *helix creation mode*  is used to create helices on grid positions.

Grids are added to the design using buttons from the Grid tab. These buttons are:  for square grid,  for hexagonal grid, and  for nanotubes. Creating a new grid automatically set the action mode to *helix creation*. In this action mode, helices are added to a grid by clicking on the desired position on the grid. One can choose to equip a helix with a double strand at its creation: just set the starting position and length of the strands in the text fields in the contextual panel. By default, a phantom helix is displayed when a helix is created, this can be switched off later on in contextual panel after selecting the grid.

**Building a first design.** Figure 6.3 presents step-by-step how to build a very simple design in ENSnano.

## 6.2 The main 2D and 3D views

Figure 6.4 shows how an example design is represented in both the 2D and 3D views.

ENSnano’s interface is designed so that these two views work together. They are fully synchronized: modifications of the design made in one view are immediately visible in the other one. Moreover, hovering a design element (strand, helix or nucleotide) with the mouse cursor in one view will highlight it (in green) in both views. This makes the correspondence between the two representations easier to grasp. Finally, double-clicking on a nucleotide or crossover in one the two views will have the other one focus on it.



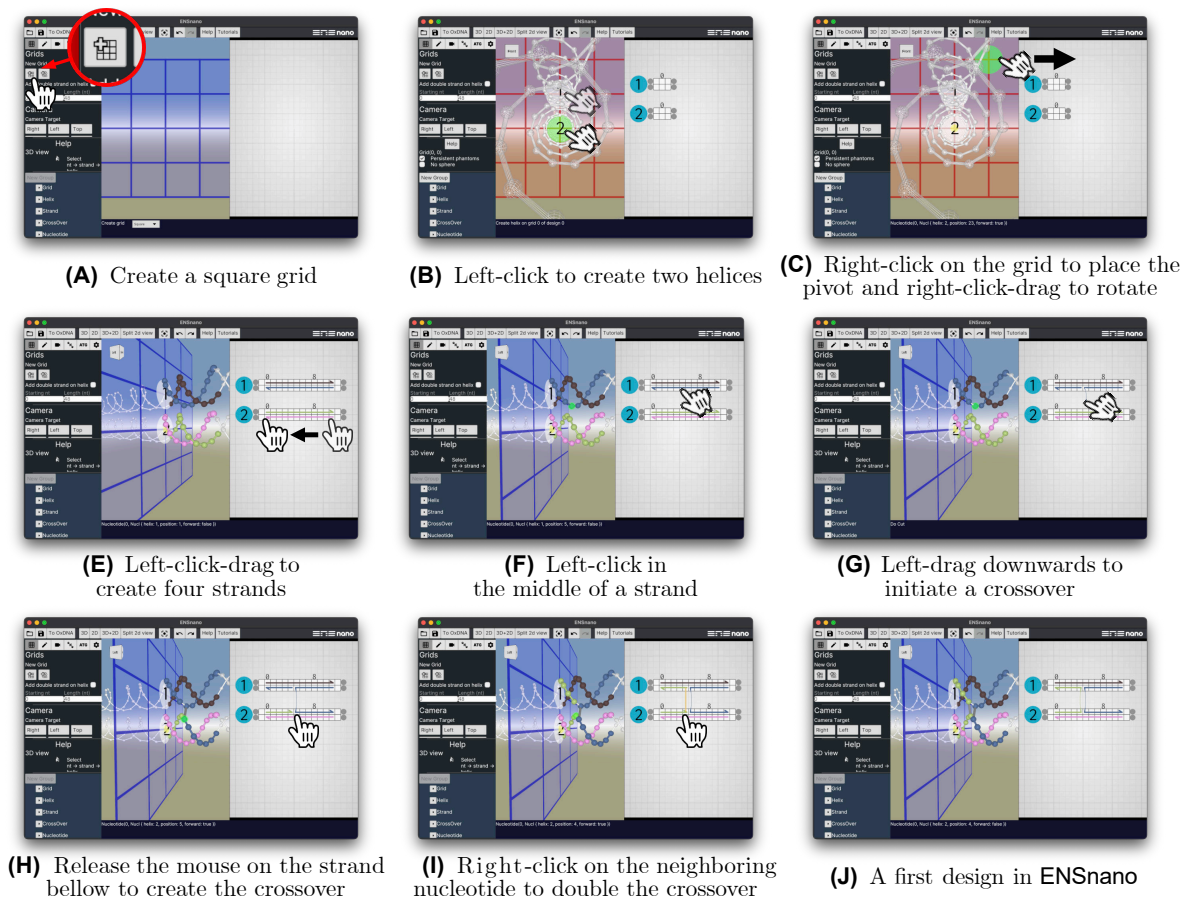


Figure 6.3: **Step-by-step construction of a first design in ENSnano.**

**3D-view.** The main purpose of the 3D view is to visualize and organize the components in space.

**3D organization of the design.** The grids and helices can be rearranged by selecting them and choosing the  $\oplus$  or  $\odot$  action mode. Red, blue and green handles appear to be pulled in the three possible directions (Figure 6.5). These handles can be chosen to be either aligned with the selected object's frame, or with the canonical world's frame.

**Fog.** Sometimes, the 3D view can become jammed in many helices are in the field of view. The *fog* feature from the camera tab  $\blacksquare$  allows displaying only the part of the design within a given radius around either the pivot or the position of the camera, fading the rest progressively to invisible (Figure 6.6).

**Editing strands and crossovers in the 3D view.** There are some cases where the 3D view is better suited for editing strands and crossovers. Left-clicking and dragging the extremity of a strand or crossover allows to translate it along the helix. To create crossovers, click on a nucleotide and hold until it becomes highlighted in blue (250 ms), then drag to the other end of the crossover and release.

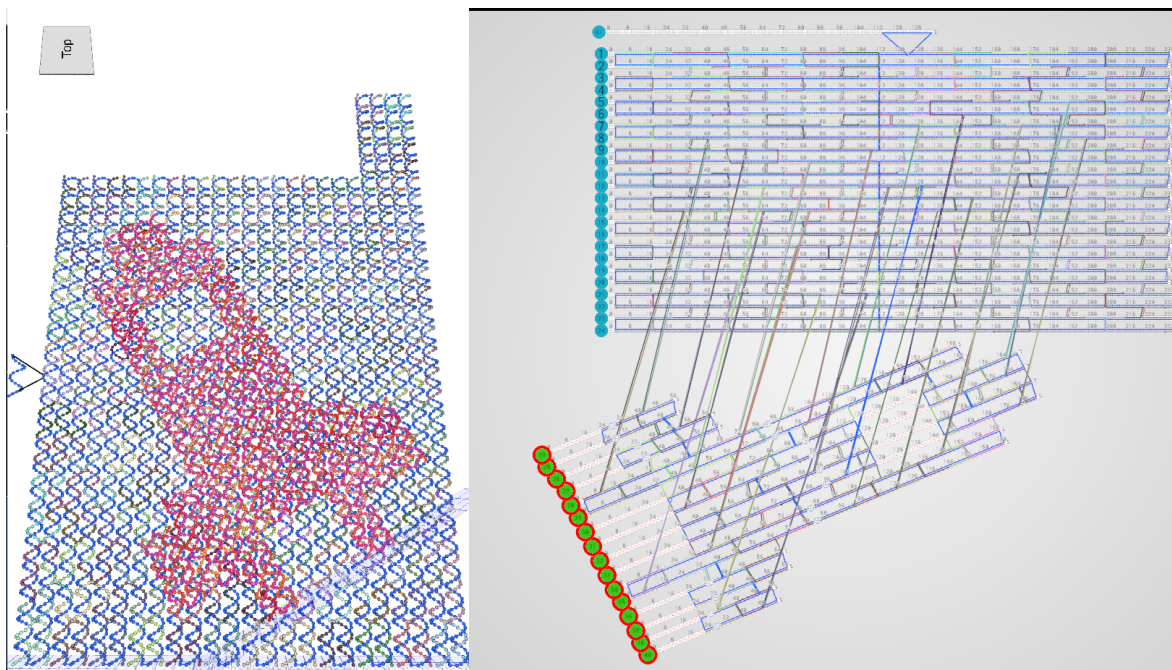

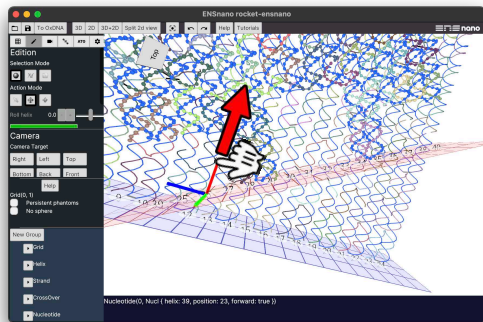


Figure 6.4: **Comparison between the 2D and the 3D views.** The 3D interface shows the shape, that we wish that our design will adopt. Since this design is made of two layers making an odd angle. It cannot be faithfully represented in 2D. By separating the helices of the two layers, and adjusting the orientation of the helices composing the rocket layer, one can build a 2D representation of the design that is as close as possible to its intended 3D structure:

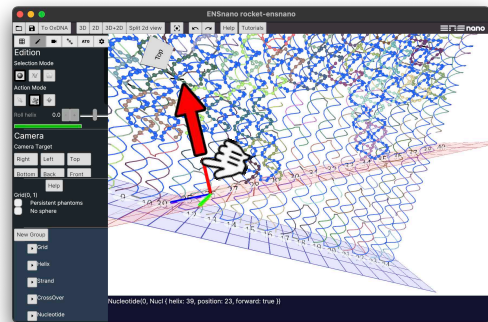
**Crossover length color shading.** ENSnano offers visual clues for assessing the length of crossovers. In the 3D view, the crossovers are displayed as cylinder whose color depends on the distance between their extremities. Short crossovers have the same color as the strand they belong to, while crossovers of excessive length are shaded from light grey to black, darker nuances indicating longer crossovers (Figure 6.7).

**The 2D view.** The 2D view is the blueprint of the design. It follows and extends the streamlined interface of cadnano and scadnano by adding a list of features that improves its ergonomics:

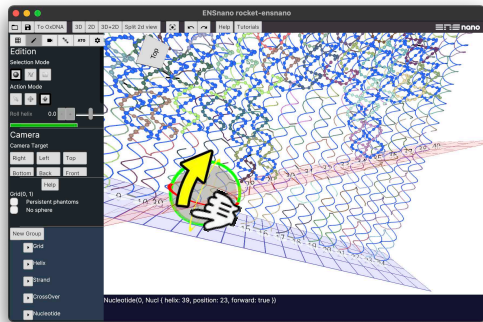
- Creation and edition of strands, creation and translation of crossovers, cutting and ligating strands are all done in the same edition mode, using only the mouse, as shown in Figure 6.8.
- The 2D and 3D views work together: a translucent green ball indicates in the 3D view which nucleotide is hovered by the mouse in the 2D view (e.g., see Fig. 6.8F and 6.8H). Also, double-left-clicking a nucleotide in the 2D view centers it in the 3D view.
- The helix representations automatically extend when needed, e.g. when elongating a strand. The helix representations can be tighten back using the buttons “All/Selected” under “Tighten 2D helices” in the edition tab , or simply by clicking on their handles.



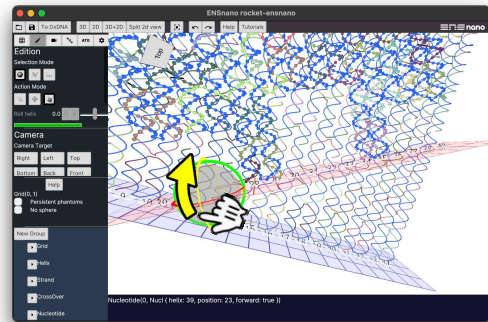
(A) Translation handles aligned with the canonical axes



(B) Translation handles aligned with the object axes



(C) Rotation handles aligned with the canonical axes



(D) Rotation handles aligned with the object axes

Figure 6.5: **Translating/rotating objects of the design.**

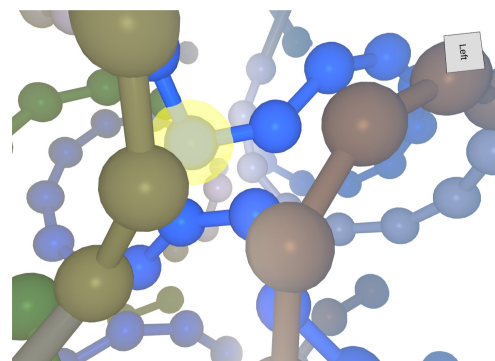
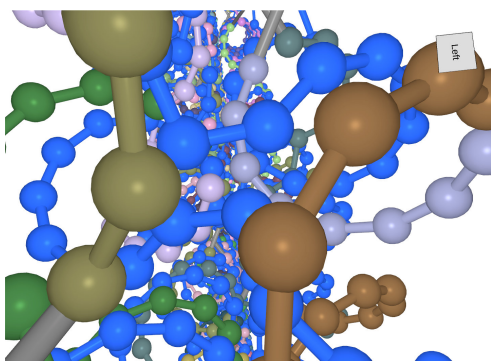


Figure 6.6: **The fog feature.** On the left, the fog feature is disabled. The background is jammed with strands, and it is hard to focus on the crossovers between the two layers; On the right, thanks to the fog feature, the background is cleared and the crossover is now clearly visible, ready to be adjusted if needed.



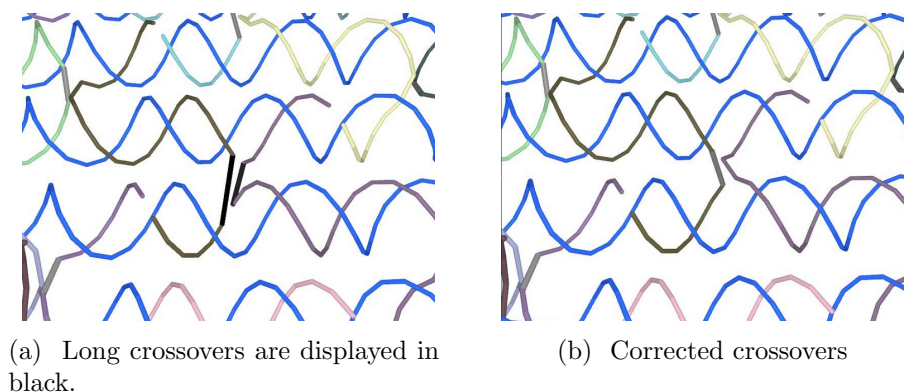


Figure 6.7: **Length color shading of the crossovers in the 3D view.** a: In this example, several crossovers of various lengths are visible. Crossovers that are short and don't require the designer's attention are displayed in the same color as their strand. Some crossovers are displayed in light-grey indicating that their moderately excessive length may only represent a minor problem in the design. However, one pair of crossovers is displayed in black. This should catch the designer's eyes and indicates that this pair of crossovers should be relocated. b: After correcting the two faulty crossovers, they are now shorter and appear in a lighter shade. This indicates that the design on the right panel is more likely to be feasible.

- Any helix representation can be translated and rotated arbitrarily in the 2D view, to match as closely as possible the 3D arrangement of design, or serve any other purposes (e.g. Figure 6.4). Left-clicking on their number and drag will translate the selected helices, while right-clicking and drag will rotate them.
- Moving in both views is done by middle-click and drag. Zooming in and out is done by scrolling the mouse wheel.

**Splitting the 2D view.** As mentioned earlier, some 3D complex structures cannot be faithfully mapped into 2D, and some parts that are next to each other in space, will inevitably be mapped far apart in any 2D view. This usually makes 2D representation of 3D DNA nanostructures complex to read and even more to edit. For instance, very long crossovers cross each other in every possible direction in the cadnano representation of the double-layer origami in [TLJ<sup>+</sup>17], and make it almost impossible to edit. ENSnano's 2D split view solves this issue elegantly by allowing to zoom and to *act seamlessly on two distant parts* of the 2D view, as if they were next to each other. Indeed, one can build a crossover from one split view to the other just as if it was one single 2D view, see Figure 6.9. Moreover, crossovers whose ends are on opposite sides of the split view, *are drawn across the views*, making them easy to read and edit.

Note that left double-clicking on a crossover in the 3D view, splits the 2D view as soon as both of its ends do not fit in the 2D view. We thus recommend to 1) create the desired crossover approximately in the 3D view and then 2) to double-click on it, so that it gets focused and drawn across the split 2D view, where the user can then edit it comfortably.

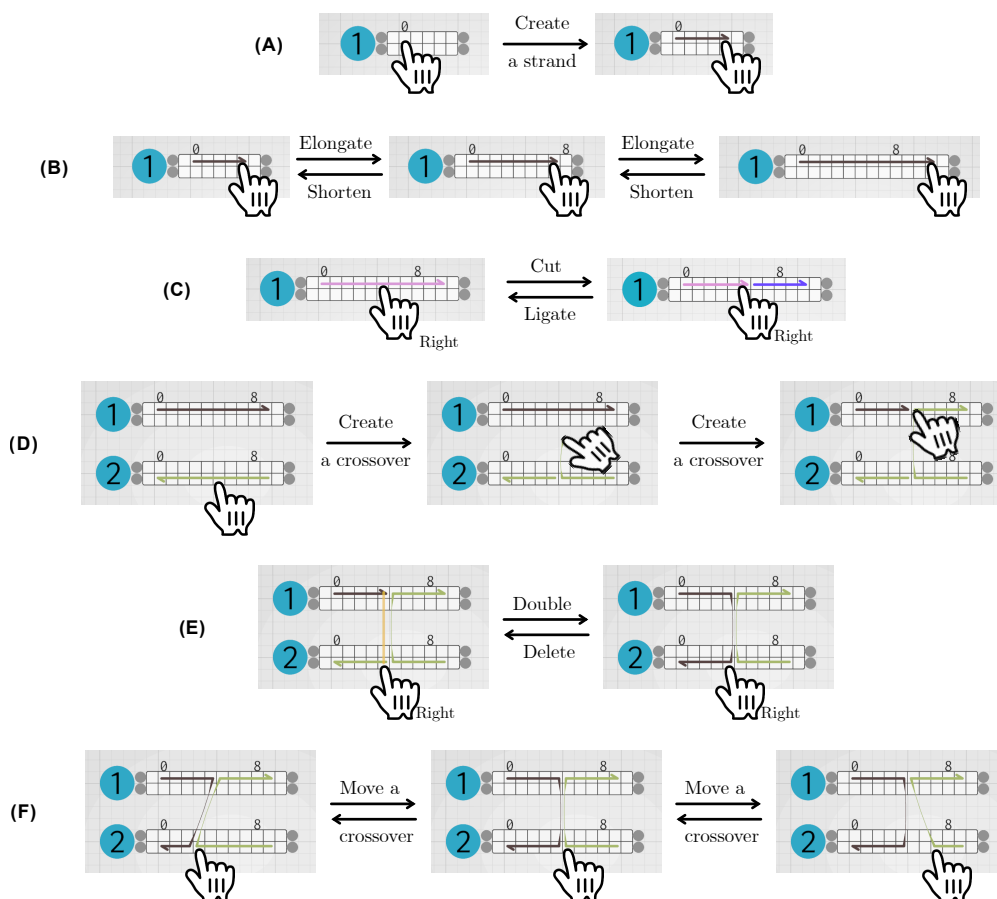


Figure 6.8: **Swift strand editing in the 2D view.** A) **Create a strand:** Left click on an empty position and drag. B) **Extend a domain:** Left click on an end of a domain and drag. Note that the helix automatically extends if needed. C) **Cut & ligate a strand:** A right click in the middle of strand cuts the strands. A right click on the end of a strand next to another ligates them. D) **Create a crossover:** A left click and drag up-/down-wards on a strand initiates the creation of a crossover that will bind the initial click position to the position where the mouse is dragged to. E) **Double/Delete a crossover:** A right click on an unconnected end of a strand next to a crossover will double the crossover. A right click on one end of a crossover will break it. F) **Move a crossover:** A left click-and-drag at one end of a crossover will move this crossover. Note that the possibly neighboring crossover will be pushed and pulled back during the dragging until the final position is set.

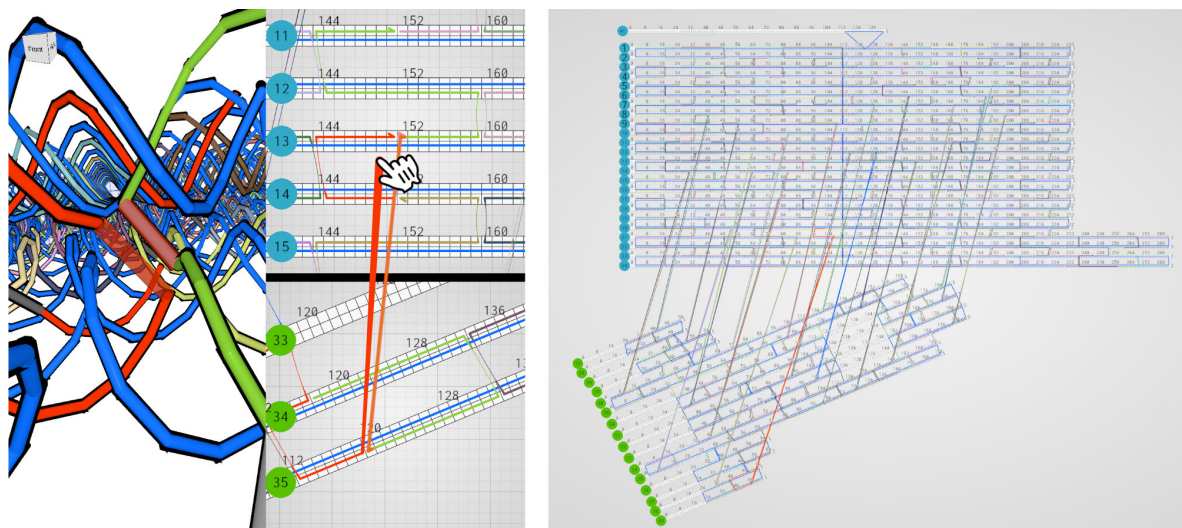


Figure 6.9: **Editing with the 2D split view.** Left: Building a crossover between “2D-distant” helices by dragging the mouse from one split view to the other. Right: In a single 2D view, the zoom factor required to see both ends of the crossover (highlighted in red) would be so small that precise editing would be impractical.

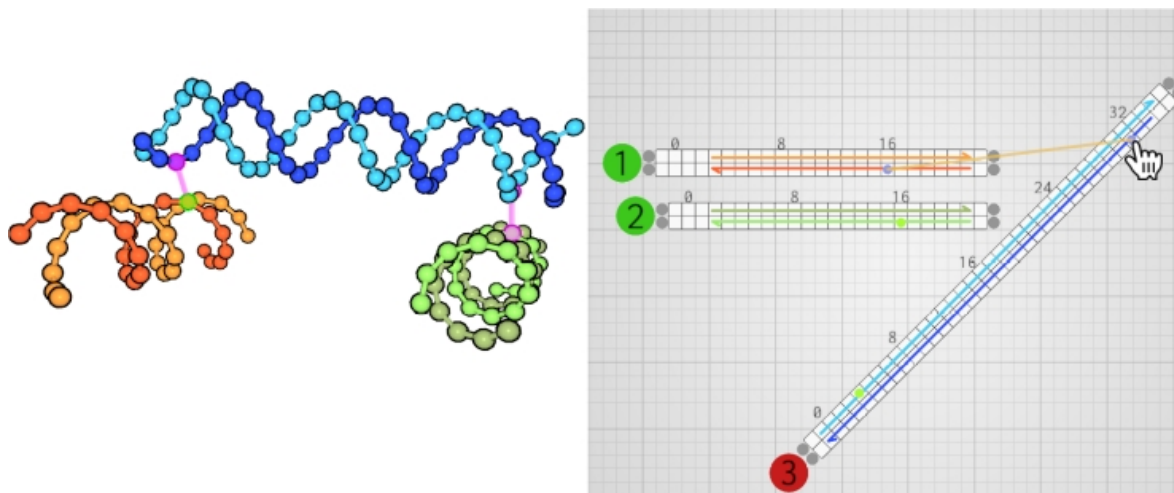


Figure 6.10: **Crossover suggestions.** As none of the three helices are parallel, finding crossover positions may be difficult. We thus assign the helices 1 and 2 (the orange and green stranded) to the green family and helix 3 (the blue stranded) to the red family in the organizer (note that the color family is displayed as the background color of their identifier disc in the 2D view). In the 3D view, the suggested crossovers are indicated by a translucent purple connection between two nucleotides. In the 2D view, the nucleotides that could be bound by a crossover are indicated by pairs of dots of matching color.

### 6.3 Geometry-based features

Some features of our software exploit the fact that ENSnano’s data structure directly embeds the geometry of the design.

**Crossover suggestions.** Cadnano offers crossover suggestions by indicating periodic positions at which neighbouring DNA helices can be connected. These suggestions are simply constructed by repeating a 32-periodic pattern, assuming that all helices are straight and parallel, and that DNA double helices have a fractional number of base pairs per turn ( $\frac{32}{3} = 10.67$  bp/turn). While these suggestions allow the user to gain time, they cannot be used to connect non-parallel helices. Since they are based on an approximation of the geometry of DNA, the resulting design also need to be adjusted, for example by adding a deletion every 48 nucleotides [WR11].

In ENSnano the 3D positions of all nucleotides is known at all time. This can be exploited to offer geometry-based crossover suggestions: the suggested crossovers are simply those that would connect nucleotides with a short-enough distance between them (Figure 6.10).

To avoid cluttering the interface, crossover suggestions are, by default, only shown between user-specified helices. Helices can be assigned a *crossover suggestion family*: none, red or green. Crossover suggestions are made between helices from the red family and the green family. Crossover suggestions can be used to facilitate the connection of non-parallel components (Figure 6.11). To connect two components, assign their helices to a crossover family. All the possible crossovers will then appear in both the 3D view (as a purple line connecting the nucleotides), and 2D view (as a pair of dots of matching colors).

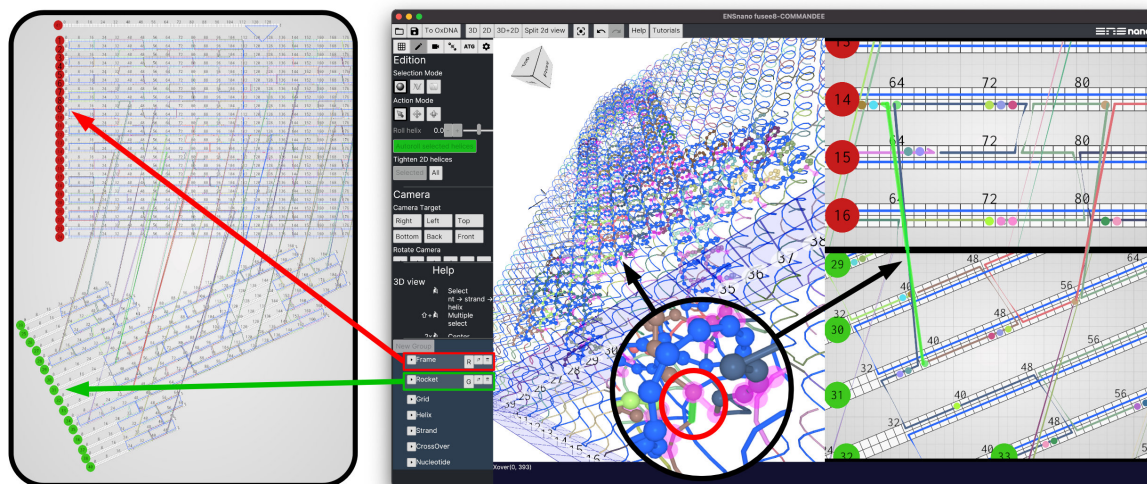


Figure 6.11: **Crossover suggestions between the red and green color families.** The helices are partitioned into two groups in the organizer: “Frame” and “Rocket” assigned resp. to the red and green families. One can see the pair of matching dots in the split view marking recommended positions for crossovers.

**Grid-geometry aware copy and paste.** Many DNA nanostructure designs contain a pattern that is repeated multiple times. This is for example the case of SST nanotubes [YHS<sup>+</sup>08, WDM<sup>+</sup>19], rectangular DNA origami [WR11], or 3D SST designs [KOSY12]. For these kinds of design, the possibility to duplicate patterns can save a lot of time. Scadnano for example offers the possibility to copy and paste strands.

ENSnano allows strands and crossovers to be duplicated by using the grid structure to compute the 3D path followed by the copied pattern, to paste the same path at a different location, regardless of the indices of the helices and of their relative positions in the 2D view. Complex strands can thus be copied and pasted across the design (Figure 6.12a). For example, a strand binding two consecutive helices in a nanotube can be copied all around the nanotube: ENSnano will automatically loop the strand around the tube from the last to the first helix.

**Paste and repeat.** In addition to the classic copy and paste feature, ENSnano offers a geometry-aware *paste-and-repeat* which remembers as well the translation (in the grid and along the helices) between the original and the first pasted pattern, to keep pasting iteratively the pattern with the same translation. This is particularly useful for large repetitive designs such as SST nanotubes or rectangular parts of an origami.

After copying the strands or crossover pattern with  $\text{Ctrl}/\text{⌘}+\text{C}$ , the first duplication is made by pressing  $\text{Ctrl}/\text{⌘}+\text{J}$ . Once the first copy is positioned, the translation from the original to the copy is memorized. Pressing  $\text{Ctrl}/\text{⌘}+\text{J}$  repetitively, will copy over and over the pattern with the same offset, as long as there are helices to support them (Figure 6.13)



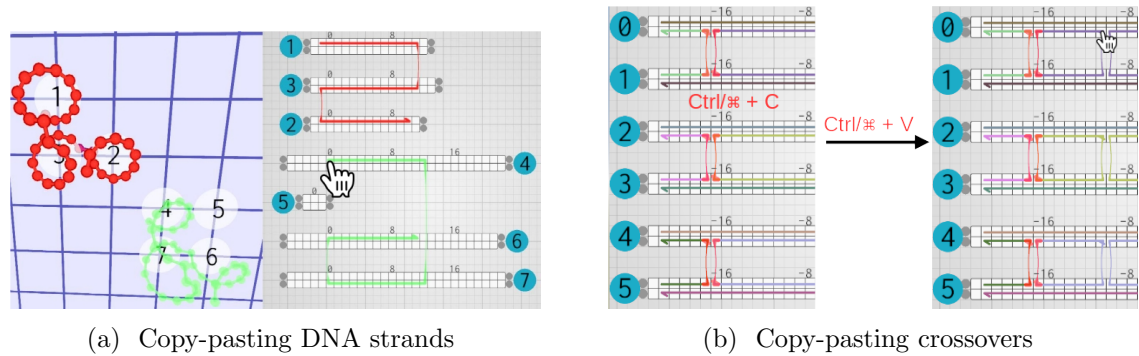


Figure 6.12: **Grid geometry-aware copy and paste of strands and crossovers.** a: Duplication of a strand. The red strand gets duplicated on other helices of the grid. One can check in the 3D interface that the path of the strand is correctly being copied, even if the 2D view could be reorganized to present a clearer representation of the strand. b: Duplication of crossovers. Four crossovers are being copied at once on existing strands.

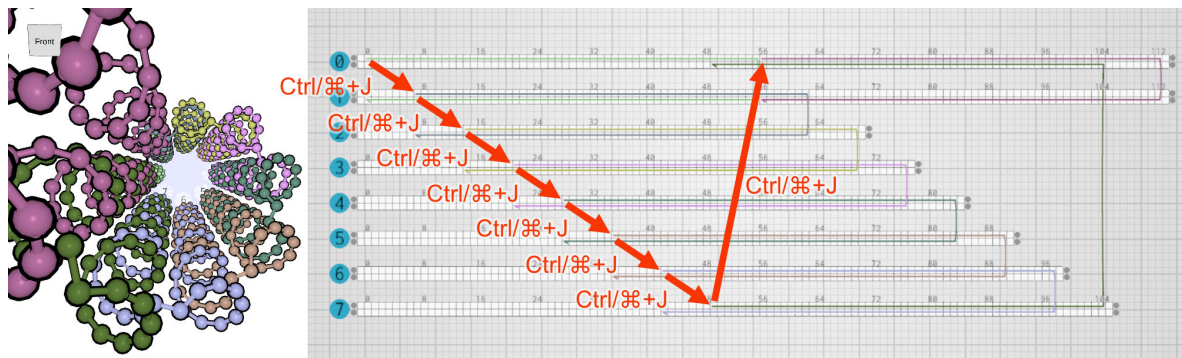


Figure 6.13: **Paste and repeat.** Starting from the green strand made of two domains of 56 nucleotides each on helices 0 and 1 of an 8-helices nanotube, this strand is copied with  $\text{Ctrl}/\text{⌘} + \text{C}$  and pasted with  $\text{Ctrl}/\text{⌘} + \text{J}$ , one helix below and 7 nucleotides forward. Repeating  $\text{Ctrl}/\text{⌘} + \text{J}$  compulsively 7 more times creates automatically the other strands, applying repetitively the same translation and thus filling the nanotube with strands in no time. Note that ENSnano is aware of the nanotube-grid geometry and places the 7th pasted olive strand appropriately, binding helices 7 and 0.

## 6.4 Experimental validation: rocket origami with two non-parallel layers

In order to test ENSnano’s geometric model, we design and assemble an origami with two non-parallel layers. The origami has the shape of a rocket lying on a rectangular frame. Both layers are made of parallel helices, but there is a  $30^\circ$  angle between the helices of the rocket and those of the frame.

The shape of the rocket was designed with students of the CR11 class on Molecular Computing at ENS de Lyon, under the supervision of Nicolas Schabanel (NS). After positioning the helices forming each layer on their respective grid, the scaffold routing and staple design was straightforward thanks to ENSnano’s 3D view and crossover recommendation’s. The split 2D view was of great help as well, especially for adjusting the crossovers between the two layers. The final design can be seen on several figures from the previous section, for example on Figures 6.1 and 6.4. *We would like to stress the fact that no external simulation software was used at any time of the design process.* The staples strands were ordered from IDT when the origami looked satisfying in ENSnano’s 3D view.

The origami was designed to use a p7249 M13 scaffold. The sequences of the staple strands were computed directly by ENSnano. NS annealed the staple strands at 10mM with 1mM scaffold in  $1\times$  TAE buffer with 12.5mM  $\text{MgCl}_2$ , with an annealing ramp starting at  $95^\circ\text{C}$  and decreasing to  $55^\circ\text{C}$  at  $-1^\circ\text{C}/\text{min}$ , then from  $55^\circ\text{C}$  to  $45^\circ\text{C}$  at  $-1^\circ\text{C}/15\text{min}$  then hold at  $25^\circ\text{C}$ . AFM images of the assembly can be seen in Figure 6.14.

The AFM images reveal that the origami assembled into the desired shape, which constitutes a first experimental validation of ENSnano’s geometry-based design method.

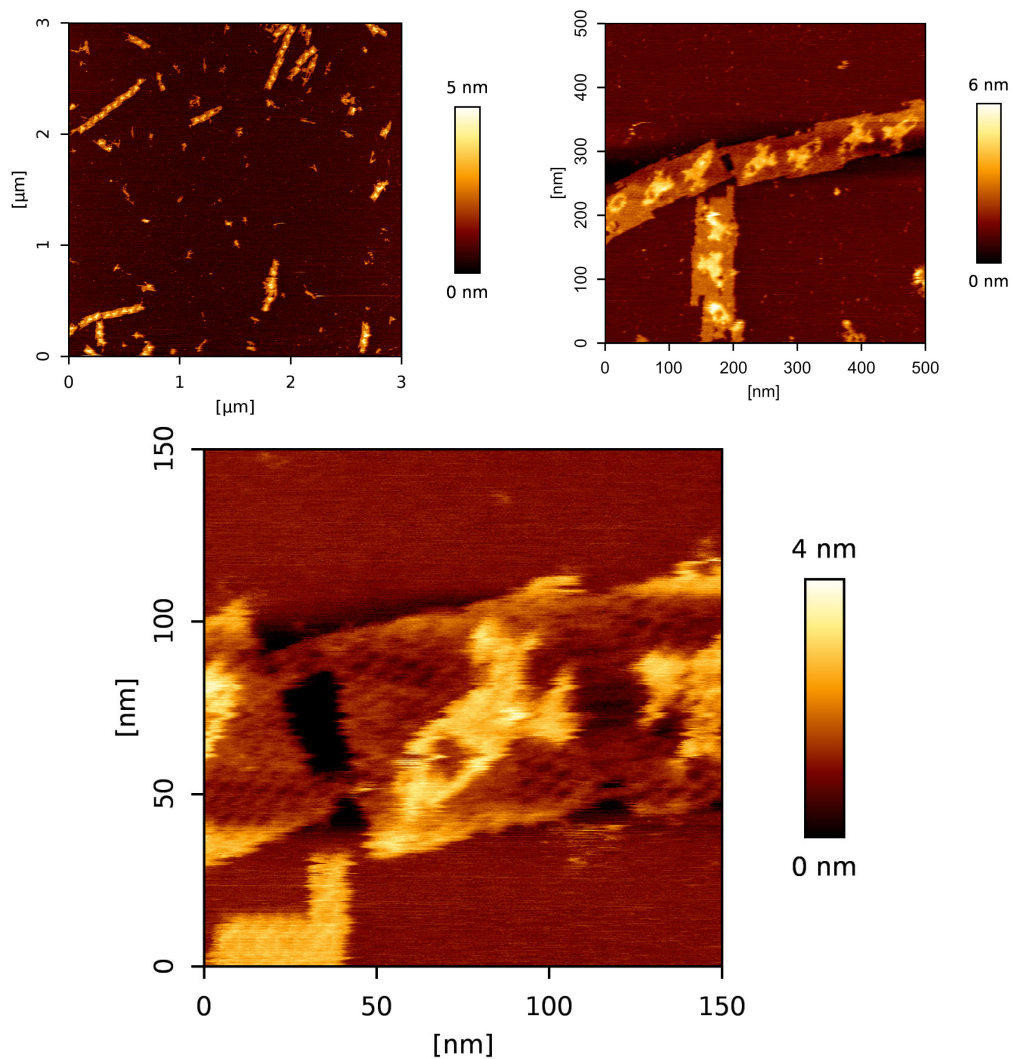


Figure 6.14: **AFM images of our rocket design.** These were obtained on a JPK Fastscan Nanoworld 4 equipped with a Nanoworld USC-F0.3-k0.3 tip in tapping mode — 20 $\mu$ L sample of: m13mp18 scaffold at 1nM with staples at 10nM in 1 $\times$  TAE buffer with 12.5mM magnesium. Assembly and imaging by Nicolas Schabanel.

## Part III

# Design of curved DNA nanostructures



# Chapter 7

## Chebyshev Polynomials

### 7.1 Mathematical properties

Chebyshev polynomials constitute a powerful tool in numerical analysis [MH02, Tre19], and we will use them extensively in this part. In Chapter 8, we will use them to approximate integrals. In Chapters 9 and 10, we will use them to create continuous interpolations of discrete sets of point/values.

The goal of this section is to introduce Chebyshev polynomials of the first kind. We will recall their properties that are useful to us, and see how to compute them efficiently.

**Definition 1.** The  $n$ -th Chebyshev polynomial of the first kind is the polynomial  $T_n$  defined by the relation

$$T_n(\cos(\theta)) = \cos(n\theta). \quad (7.1)$$

**Property 1.** Chebyshev polynomials satisfy

$$\begin{aligned} T_0 &= 1, \\ T_1 &= X, \end{aligned}$$

and the recurrence relation

$$\forall n \geq 2, T_n = 2XT_{n-1} - T_{n-2} \quad (7.2)$$

**Proof** (of relation (7.2)). Recall the trigonometric identity

$$\begin{aligned} \cos(n\theta) &= \cos((n-1)\theta - \theta) \\ \cos(n\theta) &= \cos((n-1)\theta)\cos(\theta) - \sin((n-1)\theta)\sin(\theta) \end{aligned} \quad (7.3)$$

and

$$\begin{aligned} \cos((n-2)\theta) &= \cos((n-1)\theta - \theta) \\ \cos((n-2)\theta) &= \cos((n-1)\theta)\cos(\theta) + \sin((n-1)\theta)\sin(\theta). \end{aligned} \quad (7.4)$$

Summing (7.3) and (7.4) gives

$$\cos(n\theta) + \cos((n-2)\theta) = 2\cos(\theta)\cos((n-1)\theta),$$

which by, Definition 1, gives the recurrence relation (7.2).

Since Definition 1 implies that  $T_0 = 1$  and  $T_1 = X$ , the relation (7.2) implies that  $T_n$  is of degree  $n$  for all  $n$ .

One of the reason for the popularity of Chebyshev polynomials in numerical analysis, is that they can be used to construct polynomial interpolations of smooth functions. These interpolations are “good” in a sense that will be formalized by Theorem 2.

These interpolations are constructed by interpolating the objective function on a special set of points.

**Definition 2.** The *Chebyshev points of the first kind* associated to the integer  $n$  are the  $n + 1$  roots of  $T_{n+1}$

$$C_n = \{\tilde{x}_k\} = \left\{ \cos \left( \frac{(k - \frac{1}{2})\pi}{n + 1} \right), 1 \leq k \leq n + 1 \right\} \quad (7.5)$$

**Definition 3.** The  $n$ -th *Chebyshev interpolant* of a function  $f$  is the unique polynomial  $p_n$  of degree at most  $n$  so that for every Chebyshev point  $\tilde{x}_k \in C_n$ ,  $f(\tilde{x}_k) = p_n(\tilde{x}_k)$ .

**Remark 1.** In the other chapters, Chebyshev interpolants of a function  $f$  are designated by the notation  $\tilde{f}$ . Here, we use the notation  $p_n$  to stress the fact that they are polynomials.

When  $f$  is continuous,  $p_n$  is a good interpolation of  $f$  in the following sense:

**Theorem 2** ([Riv06]). Let  $f$  be a continuous real-valued function on  $[-1, 1]$ , let  $p_n$  be the  $n$ -th Chebyshev interpolant of  $f$  and let  $p^*$  be a polynomial satisfying

$$\|p^* - f\|_\infty = \min_{p \in \mathbb{R}_n[X]} \|p - f\|_\infty,$$

where  $\|\cdot\|_\infty$  is the infinite norm

$$\|g\|_\infty = \max_{x \in [-1, 1]} |g(x)|.$$

Then there exists a constant  $\Lambda_n$  so that

$$\|p_n - f\|_\infty \leq (1 + \Lambda_n) \|p^* - f\|_\infty$$

and

$$\Lambda_n \leq \frac{2}{\pi} \log(n + 1).$$

The constant  $\Lambda_n$  is called the *Lebesgue constant* associated to the polynomial  $p_n$ . For comparison’s sake, in the case of the polynomial  $q_n$  obtained by interpolating  $f$  on a set of  $(n + 1)$  equispaced points, the associated Lebesgue constant is much larger and verifies ([Tre19], Theorem 15.2)  $\Lambda_n > \frac{2^{n-2}}{n^2}$ .

It remains to show how to build Chebyshev interpolants with Chebyshev polynomials. Since each  $T_i$  is of degree  $i$ ,  $(T_i)_{i \leq n}$  is a basis of  $\mathbb{R}_n[X]$ . The value of the coefficients associated to the decomposition of  $p_n$  in this basis are given by the following theorem:

**Theorem 3** ([MH02], Theorem 6.7). The  $n$ -th Chebyshev interpolant  $p_n$  can be expressed as a sum of Chebyshev polynomials:

$$p_n = \sum_{i=0}^n c_i T_i,$$

where, for  $i \geq 1$ ,

$$c_i = \frac{2}{n+1} \sum_{k=1}^{n+1} f(x_k) T_i(x_k),$$

and

$$c_0 = \frac{1}{n+1} \sum_{k=1}^{n+1} f(x_k) T_i(x_k).$$

**Proof** (of Theorem 3, adapted from [MH02] Sections 4.6 and 6.3). Consider the sum

$$s_n^{(1)}(\theta) = \sum_{x_k \in C_n} \cos\left(\left(k - \frac{1}{2}\right)\theta\right) = \cos\left(\frac{1}{2}\theta\right) + \cos\left(\frac{3}{2}\theta\right) + \cdots + \cos\left(\frac{2n+1}{2}\theta\right). \quad (7.6)$$

If we take  $z = e^{i\theta}$ , one can see that

$$\begin{aligned} s_n^{(1)}(\theta) &= \Re\left(z^{\frac{1}{2}}(1 + z + \cdots + z^n)\right) \\ s_n^{(1)}(\theta) &= \Re\left(z^{\frac{1}{2}} \frac{1 - z^{n+1}}{1 - z}\right) \\ s_n^{(1)}(\theta) &= \Re\left(\frac{(z^{-\frac{n+1}{2}} - z^{\frac{n+1}{2}}) z^{\frac{n+1}{2}}}{z^{-\frac{1}{2}} - z^{\frac{1}{2}}}\right) \\ s_n^{(1)}(\theta) &= \Re\left(\frac{-2i \sin\left(\frac{n+1}{2}\theta\right) (\cos\left(\frac{n+1}{2}\theta\right) + i \sin\left(\frac{n+1}{2}\theta\right))}{2i \sin\left(\frac{\theta}{2}\right)}\right) \\ s_n^{(1)}(\theta) &= \frac{2 \sin\left(\frac{n+1}{2}\theta\right) \cos\left(\frac{n+1}{2}\theta\right)}{2 \sin\left(\frac{\theta}{2}\right)} \\ s_n^{(1)}(\theta) &= \frac{\sin((n+1)\theta)}{2 \sin\left(\frac{\theta}{2}\right)}. \end{aligned} \quad (7.7)$$

Therefore, we have (from (7.6))  $s_n^{(1)}(0) = n+1$ ,  $s_n^{(1)}(2\pi) = -(n+1)$ , and (from (7.7)) for integers  $0 < r < 2(n+1)$ ,  $s_n^{(1)}\left(\frac{r\pi}{n+1}\right) = 0$ .

This gives, for an integer  $r \leq 2(n+1)$ ,

$$s_n^{(1)}\left(\frac{r\pi}{n+1}\right) = \begin{cases} n+1 & \text{if } r = 0 \text{ (from (7.6))} \\ 0 & \text{if } 0 < r < 2(n+1) \text{ (from (7.7))} \\ -(n+1) & \text{if } r = 2(n+1) \text{ (from (7.6))} \end{cases} \quad (7.8)$$

Now, we consider

$$a_{ij} = \sum_{x_k \in C_n} T_i(x_k) T_j(x_k) \quad (7.9)$$

where  $C_n$  is the set of roots of  $T_{n+1}$  (see Definition 2). For  $1 \leq k \leq n+1$ , we write  $\theta_k =$



$\arccos(x_k)$ , we have (from Definition 1)

$$\begin{aligned}
a_{ij} &= \sum_{k=1}^{n+1} \cos(i\theta_k) \cos(j\theta_k) \\
a_{ij} &= \frac{1}{2} \sum_{k=1}^{n+1} \cos((i+j)\theta_k) \cos((i-j)\theta_k) \\
a_{ij} &= \frac{1}{2} \left( s_n^{(1)} \left( \frac{(i+j)\pi}{n+1} \right) + s_n^{(1)} \left( \frac{(i-j)\pi}{n+1} \right) \right) \\
a_{ij} &= \begin{cases} 0 & \text{if } i \neq j \text{ and } i, j \leq n \\ n+1 & \text{if } i = j = 0 \\ \frac{1}{2}(n+1) & \text{if } 0 < i = j \leq n \end{cases}. \tag{7.10}
\end{aligned}$$

Now, back to the decomposition of  $p_n$  in the basis  $(T_i)_{i \leq n}$ . By definition of  $p_n$  we have for  $x_k \in C_n$

$$f(x_k) = p_n(x_k) = \sum_{i=0}^n c_i T_i(x_k).$$

By multiplying by  $\frac{2}{n+1} T_j(x_k)$  and summing, we get

$$\begin{aligned}
\frac{2}{n+1} \sum_{k=1}^{n+1} f(x_k) T_j(x_k) &= \sum_{i=0}^n c_i \left( \frac{2}{n+1} \sum_{k=1}^{n+1} T_i(x_k) T_j(x_k) \right) \\
&= \sum_{i=0}^n c_i \frac{2}{n+1} a_{ij} \\
&= \begin{cases} c_j & \text{if } j > 0 \\ 2c_0 & \text{otherwise} \end{cases} \tag{from 7.10}
\end{aligned}$$

**Evaluating Chebyshev polynomials** Now that we have decomposed  $p_n$  in the basis of Chebyshev polynomials, it remains to see how to evaluate  $p_n(x)$  for a given  $x \in [-1, 1]$ .

In general, a polynomial of the form  $P = \sum_{i=0}^n a_i X^i$  can be efficiently evaluated by Horner's method [VZGG13, Section 5.2]. However, in the case of a sum of Chebyshev polynomials, this method is not the best. First, we have not computed the coefficients  $a_i$  associated to the decomposition of  $p_n$  in the canonical basis of  $\mathbb{R}_n[X]$ . Second, while it is possible to compute these coefficients using the recurrence formula (7.2), there is a better suited method to evaluate  $p_n$ .

The method, presented in [MH02, Section 2.4.1], is as follows: In order to evaluate

$$p_n(x) = \sum_{i=0}^n c_i T_i(x),$$

we can rewrite it as

$$p_n(x) = (c_0, c_1, \dots, c_n) \begin{pmatrix} T_0(x) \\ T_1(x) \\ \vdots \\ T_n(x) \end{pmatrix} = \mathbf{c}^T \mathbf{p}.$$



**Interpolation of a set of point-values.** This case can be reduced to the interpolation of a function. To do so, the set  $\{(x_i, y_i)\}$  is sorted so that  $x_0 < x_1 < \dots < x_{N-1}$  and the Chebyshev interpolant is constructed by interpolating the piecewise linear function  $\hat{f}$  defined on  $[x_0, x_{N-1}]$  by

$$\hat{f}(x) = \begin{cases} y_i & \text{when } x \text{ is equal to some } x_i \\ y_i + \frac{x-x_i}{x_{i+1}-x_i}(y_{i+1} - y_i) & \text{when } x_i < x < x_{i+1}. \end{cases}$$

The function  $\hat{f}$  is then interpolated using the method presented in the next paragraph. The interpolation error for  $\hat{f}$  will be evaluated on the set  $\{x_i\}$ .

**Interpolation of a function.** In order to define an interpolant for the function  $f : [a, b] \rightarrow \mathbb{R}$ , we shift it on the interval  $[-1, 1]$  by defining

$$f^{\leftrightarrow}(x) = f\left(2\frac{x-a}{b-a} - 1\right).$$

Using Theorem 3, we obtain an interpolant  $p_n^{\leftrightarrow}$  of  $f^{\leftrightarrow}$  which we can use to interpolate  $f$  using the relation

$$f(x) = f^{\leftrightarrow}\left(a + (x+1)\frac{b-a}{2}\right) \simeq p_n^{\leftrightarrow}\left(a + (x+1)\frac{b-a}{2}\right)$$

Our interpolant of degree  $n$  is therefore defined as

$$p_n(x) = p_n^{\leftrightarrow}\left(a + (x+1)\frac{b-a}{2}\right). \quad (7.13)$$

The interpolation algorithm (Algorithm 1) consists in constructing the  $n$ -th interpolant of  $f$  for successive values of  $n$  until the error

$$\delta_n = \max_{0 \leq i < N} |f(x_i) - p_n(x_i)|$$

is smaller than  $\varepsilon$ . If we have not found a good enough interpolant when  $n$  reaches a certain value `MAX_DEGREE`, we return the interpolant that minimizes  $\delta_n$  among those that have been computed.

In our implementation, we take `MAX_DEGREE = 100` and this bound is rarely reached. Note that the most time-consuming step in this algorithm is the computation of  $\delta$  on Line 6 of Algorithm 1. In order to provide a reactive user interface, we want the execution time of Algorithm 1 to be bounded by a few tens of milliseconds. To do so, we restrict ourselves to cases where  $N \leq 10^5$ . We also benefit from the fact that the computation of  $\delta$  can be easily parallelized by dividing the set  $\{x_i\}$  in  $k$  subsets of approximately equal size. In Rust this is easily done thanks to the `rayon` library [SM] that provides parallel iterators. Listing 7.1 shows how the interpolation error is evaluated.

This is an example of how the Rust compiler guarantees the safety of concurrent code: In order to be allowed to use the `par_iter` method, the compiler needs to be able to check that the interpolated function `F` verifies the property `Send + Sync` which implies that it can be safely shared between threads. If we try to use this method to interpolate a function that does not have this property (for example because it has side effects), then the compiler will issue an error, stating that the thread safety requirements for `F` are not met.

---

**Algorithm 1** Construction of an interpolation of  $f : [a, b] \rightarrow \mathbb{R}$  with error threshold  $\varepsilon$  evaluated on a set of points  $\{x_i\} \subset [a, b]$

---

```

1:  $n \leftarrow 0$ 
2:  $\delta^* \leftarrow \infty$  ▷ Error of the currently best interpolant
3:  $p^* \leftarrow \text{None}$  ▷ Currently best interpolant
4: while  $n \leq \text{MAX\_DEGREE}$  do ▷ As defined by (7.13)
5:    $p \leftarrow p_n$ 
6:    $\delta \leftarrow \max_{0 \leq i < N} |f(x_i) - p_n(x_i)|$ 
7:   if  $\delta < \varepsilon$  then
8:     return  $p_n$ 
9:   else if  $\delta < \delta^*$  then
10:     $p^* \leftarrow p_n$ 
11:     $\delta^* \leftarrow \delta$ 
12:   end if
13: end while
14: return  $p^*$ 

```

---

Listing 7.1: Parallelized evaluation of the maximum interpolation error on a specified set of points

---

```

struct FunctionInterpolator<F>
where
  F: Fn(f64) -> f64 + Send + Sync // F is a function f64->f64 that can be shared between threads
{
  f: Box<F>, // The size of an 'F' cannot be known at compile time so we access it through a pointer
  polynomial: ChebyshevPolynomial,
  /*
  ...
  */
}
impl<F> FunctionInterpolator<F>
where
  F: Fn(f64) -> f64 + Send + Sync
{
  /// Evaluate of the error of the current interpolation on the given set of points
  fn error_max(&self, points: &[f64]) -> f64 {
    points
      .par_iter() // Run the following computation on a parallel iterator
      .map(|x| (self.polynomial.evaluate(*x) - (self.f)(*x)).abs())
      .max_by(|a, b| {
        if a < b { // The type f64 is not totally ordered
          std::cmp::Ordering::Less // (because NaN cannot be compared to the other)
        } else { // so we must provide a custom total
          std::cmp::Ordering::Greater // ordering function
        }
      })
      .unwrap_or(std::f64::INFINITY)
  }
}

```

---



## Chapter 8

# A model for curved DNA double helices

### 8.1 P-stick model for curved DNA helices

In order to generate the nucleotides position for curved DNA helices, we will extend The P-stick model for straight DNA helices presented in 4.3. Recall that in the P-stick model for straight helices, each helix with origin  $\mathcal{O}_H$  is equipped with a frame  $\mathcal{F}_H \in \text{SO}_3(\mathbb{R})$ , and that

nucleotides position were of the form  $\mathcal{O}_H + \mathcal{F}_H \begin{pmatrix} R \cos(\theta) \\ R \sin(\theta) \\ \text{total rise} \end{pmatrix}$

We now assume that the axis of the helix  $H$  follows a curved path  $\mathcal{C}_H : [0, T_{\max}] \rightarrow \mathbb{R}^3$ . By analogy with the P-stick model for straight helices, the nucleotides of this curved helix lie on circles centered on the axis of the helix. However, the total rise of each nucleotide is now measured along a curved path. Therefore, the projection of the nucleotide on the axis of the helix is no longer of the form  $\mathcal{O}_H + \text{total rise} \cdot Z_H$ , and is instead the point  $\mathcal{C}_H(t)$  where  $t = \sigma(\text{total rise})$  and

$$s : T \mapsto \int_0^T \|\dot{\mathcal{C}}_H(t)\|_2 dt \text{ and } \sigma : x \mapsto s^{-1}(x) \quad (8.1)$$

are is the curvilinear abscissa and inverse curvilinear abscissa of  $\mathcal{C}_H$  (with  $\dot{\mathcal{C}}_H(t) = \frac{d\mathcal{C}_H(t)}{dt}$ ).

Moreover, the circles on which the nucleotides lie are not necessarily all parallel, instead the circle centered on  $\mathcal{C}_H(t)$  will be contained in the plane  $X_H(t)Y_H(t)$  of a local frame  $\mathcal{F}_H(t) = (X_H(t), Y_H(t), Z_H(t))$ .

This means that in order to extend the P-stick model to curved helices, we need to be able to do two things:

1. Construct the set  $T = \{t_i^f\} \cup \{t_i^b\}$  of parameters corresponding to projections of the nucleotides on the axis of the helix.
2. Construct a moving frame  $\mathcal{F}_H : I \rightarrow \text{SO}_3(\mathbb{R})$ .

Provided that these two objects have been constructed, the positions  $N_i^f$  (resp.  $N_i^b$ ) of the  $i$ -th nucleotide of the forward (resp. backward) axis are given by the following formulas:

$$N_i^f = \mathcal{C}_H(t_i^f) + RX_H(t_i^f) \cos(\rho + \beta \cdot i) - RY_H(t_i^f) \sin(\rho + \beta \cdot i) \quad (8.2)$$

$$N_i^b = \mathcal{C}_H(t_i^b) + RX_H(t_i^b) \cos(\rho + \beta \cdot i + \alpha) - RY_H(t_i^b) \sin(\rho + \beta \cdot i + \alpha), \quad (8.3)$$

where

$$t_i^f = \sigma(i \cdot \Delta) \quad (8.4)$$

and

$$t_i^b = \sigma(i \cdot \Delta + I). \quad (8.5)$$

## 8.2 Discretization of the curve

In order to construct the set  $T = \{t_i^f\} \cup \{t_i^b\}$ , we will compute a Chebyshev interpolation of the inverse curvilinear abscissa  $\sigma$ . To do so, we will construct a set of points-value  $\{(t_i, \bar{s}(t_i))\}_i$  where  $\bar{s}$  is an approximation of the curvilinear abscissa. This approximation depends on a *discretization parameter*  $\delta$  and is given by the Riemann sum

$$\bar{s}(k\delta) = \sum_{i=1}^k \|\mathcal{C}(i\delta) - \mathcal{C}((i-1)\delta)\|_2 \simeq \int_0^{k\delta} \|\dot{\mathcal{C}}(t)\|_2 dt = s(k\delta), \quad 0 \leq k \leq \left\lfloor \frac{T_{\max}}{\delta} \right\rfloor. \quad (8.6)$$

The quality of our interpolation will be determined by the value of  $\delta$ . We chose to pick a value so that, on average,  $\|\mathcal{C}(t + \delta) - \mathcal{C}(t)\| \simeq \frac{\Delta}{10}$  where  $\Delta$  is the rise of the helix. This ensures that, on average, the length between the positions of two consecutive nucleotides is obtained by summing 10 chords, which should provide a fine enough approximation of our purpose.

Therefore, to pick  $\delta$ , we first compute an approximation of the total length of the curve

$$\bar{L} = \sum_{i=1}^{N_0} \left\| \mathcal{C}\left(\frac{(i+1)T_{\max}}{N_0}\right) - \mathcal{C}\left(\frac{iT_{\max}}{N_0}\right) \right\|_2 \simeq \int_0^{T_{\max}} \|\mathcal{C}'(t)\|_2 dt.$$

Then  $\left\lfloor \frac{\bar{L}}{\Delta} \right\rfloor$  is an approximation of the number of nucleotides along the curve and set

$$\delta = T_{\max} \frac{1}{10 \left\lfloor \frac{\bar{L}}{\Delta} \right\rfloor}.$$

We then construct the set of points-value  $\{(t_i, \bar{s}_i)\}$  as defined by (8.6):

For  $0 \leq i \leq N = \frac{T_{\max}}{\delta} = 10 \left\lfloor \frac{\bar{L}}{\Delta} \right\rfloor$ ,

$$t_i = i\delta, \quad \bar{s}_i = \begin{cases} 0 & i = 0 \\ \bar{s}_{i-1} + \|\mathcal{C}(t_i) - \mathcal{C}(t_{i-1})\|_2 & 1 \leq i \leq N. \end{cases}$$

We then construct an approximation  $\tilde{\sigma}$  of  $\sigma$  by Chebyshev interpolation on the set  $\{(\bar{s}_i, t_i)\}$  using Algorithm 1 with an error threshold  $\varepsilon = 10^{-4}$ nm.

Likewise, we construct an approximation  $\tilde{s}$  of  $s$  by Chebyshev interpolation on the set  $\{(t_i, \bar{s}_i)\}$  using Algorithm 1 with an error threshold  $\varepsilon = 10^{-4}$ nm, and we write

$$\tilde{L} = \tilde{s}(T_{\max}).$$

**Adjustment of the geometric parameters for closed curve.** If  $\mathcal{C}$  is closed, (that is to say  $\mathcal{C}(T_{\max}) = \mathcal{C}(0)$ ), then we want the associated helix to loop on itself. This means that we wish that the position of its last nucleotide coincides with the position of its first nucleotide. This implies that

- The length of the helix must be an integer multiple of  $\Delta$ , *i.e.*

$$\frac{\tilde{L}}{\Delta} \in \mathbb{N} \quad (8.7)$$

- The helix must make an integer number of full turns along  $\mathcal{C}$ .

There is no reason for any of these requirements to be met *a priori*. Fortunately, when  $\tilde{L} \gg \Delta$ , it is possible to slightly tweak the geometric parameters  $\Delta$  and  $\beta$  so that both conditions hold. In order to satisfy the Constraint (8.7), we can substitute  $\Delta$  by a value  $\Delta' \simeq \Delta$  in our model.

To compute  $\Delta'$  we set  $N = \left[ \frac{\tilde{L}}{\Delta} \right]$ , where  $[\cdot]$  is the rounding function, and set

$$\Delta' = \frac{\tilde{L}}{N}.$$

When  $\tilde{L} > \Delta$ , we now have

$$\Delta' - \Delta = \frac{\tilde{L}}{\left[ \frac{\tilde{L}}{\Delta} \right]} - \Delta \leq \frac{\tilde{L}}{\frac{\tilde{L}}{\Delta} - \frac{1}{2}} - \Delta \leq \frac{\frac{\Delta}{2}}{\frac{\tilde{L}}{\Delta} - \frac{1}{2}} \quad (8.8)$$

and

$$\Delta - \Delta' = \Delta - \frac{\tilde{L}}{\left[ \frac{\tilde{L}}{\Delta} \right]} \leq \Delta - \frac{\tilde{L}}{\frac{\tilde{L}}{\Delta} + \frac{1}{2}} \leq \frac{\frac{\Delta}{2}}{\frac{\tilde{L}}{\Delta} + \frac{1}{2}} \quad (8.9)$$

Inequalities (8.8) and (8.9) give

$$|\Delta - \Delta'| \leq \frac{\frac{\Delta}{2}}{\frac{\tilde{L}}{\Delta} - \frac{1}{2}}.$$

In practice, this method will be applied to helices with at least 1000 nucleotides which gives  $\tilde{L} \geq 10^3 \Delta = 332\text{nm}$ , and in that case,

$$\Delta' = \Delta \pm 1.7 \times 10^{-4} \text{nm}.$$

The difference between  $\Delta$  and  $\Delta'$  can therefore be considered to be insignificant as it is smaller than the known precision for  $\Delta$ .

As we will see in the next Section, the adjustment of the twist parameter depends on the moving frame  $\mathcal{F} : [0, T_{\max}] \rightarrow \text{SO}_3(\mathbb{R})$  with which we equip the curve.



### 8.3 Construction of a moving right-handed orthonormal frame along a curved path

There are several ways to define a frame  $\mathcal{F}$  along a curve. One of them is the Frenet frame [AT12, Section 1.3] that is constructed using derivative of  $\mathcal{C}$  with respect to time:

**Definitions 4.** Let  $I$  be a closed interval of  $\mathbb{R}$ . Let  $\mathcal{C} : I \rightarrow \mathbb{R}^3$  be a 3D curve of class  $C^k$  with  $k \geq 2$ .

i The unit *tangent* of  $\mathcal{C}$  at point  $\mathcal{C}(t)$  is the vector

$$\mathbf{t}(t) = \frac{\dot{\mathcal{C}}(t)}{\|\dot{\mathcal{C}}(t)\|}.$$

ii The *curvature* of  $\mathcal{C}$  at point  $\mathcal{C}(t)$  is the value

$$\kappa(t) = \|\dot{\mathbf{t}}(t)\|.$$

If  $\kappa$  is everywhere nonzero on  $I$ , we say that  $\mathcal{C}$  is *biregular*.

iii If  $\mathcal{C}$  is biregular, the unit *normal* of  $\mathcal{C}$  at point  $\mathcal{C}(t)$  is the vector

$$\mathbf{n}(t) = \frac{\dot{\mathbf{t}}(t)}{\|\dot{\mathbf{t}}(t)\|}.$$

iv If  $\mathcal{C}$  is biregular, the *unit binormal vector* of  $\mathcal{C}$  at point  $\mathcal{C}(t)$  is the vector

$$\mathbf{b}(t) = \mathbf{t}(t) \wedge \mathbf{n}(t).$$

Notice that, by definition of  $\mathbf{t}$ , for all  $t \in I$ , we have

$$\langle \mathbf{t}(t) | \mathbf{t}(t) \rangle = \|\mathbf{t}(t)\|^2 = 1. \quad (8.10)$$

And therefore, by taking the derivative of (8.10) with respect to time, we get

$$\langle \dot{\mathbf{t}}(t) | \mathbf{n}(t) \rangle = 0.$$

Therefore, the triplet  $(\mathbf{n}(t), \mathbf{b}(t), \mathbf{t}(t))$  is an orthonormal basis of  $\mathbb{R}^3$ . It is this triplet that is called the *Frenet frame*.

While this construction appears to be natural, it is not well suited for our purpose. The reason is that, in a DNA double helix, stacking interactions tend to enforce a constant twist [CTP<sup>+</sup>08]. Conversely, the Frenet frame tends to rotate suddenly at inflexion points of the curve (Figure 8.1).

This is why, we will instead construct a “lazily adaptive” frame that tries to rotate as little as possible along the curve  $\mathcal{C}$  while keeping its  $Z$  axis equal to the curve’s tangent. To do so, the axes  $X$  and  $Y$  are constructed incrementally along the curve at the required locations  $t_i^f, t_i^b$ .

- Let  $T = \{t_i^f\} \cup \{t_i^b\}$  be the set of the parameters where we need to compute the frame and sort it as  $T = \{0 = t_1 < \dots < t_{2n}\}$ .

- $X(0)$  and  $Y(0)$  are set arbitrarily so that  $(X(0), Y(0), Z(0))$  is a properly oriented frame, for instance: take  $u \in \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$  such that  $u$  is not collinear to  $Z(0)$  and set  $Y(0) = Z(0) \wedge u$  and  $X(0) = Y(0) \wedge Z(0)$ .
- For  $k = 2, 3, \dots, 2n$ , we set  $Y(t_k) = Z(t_k) \wedge X(t_{k-1})$  and  $X(t_k) = Y(t_k) \wedge Z(t_k)$ .

This algorithm ensures that  $Y(t_k) = Z(t_k) \wedge X(t_{k-1})$  (by definition of  $Y(t_k)$ ) tries to stay as close as possible to  $Y(t_{k-1}) = Z(t_{k-1}) \wedge X(t_{k-1})$  (by definition of  $X(t_{k-1})$ ), hopefully adapting minimally to the new tangent, and minimizing the resulting twist in the frame from one step to the next.

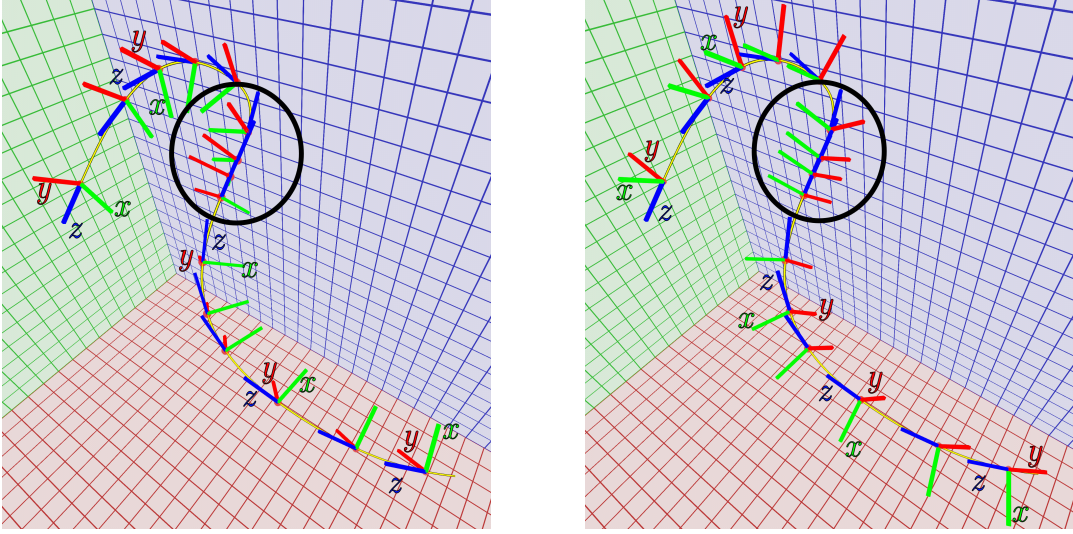


Figure 8.1: **Comparison between the Frenet frame (left) and the lazily adaptive frame (right) for a given curve.** Notice the quick rotation performed by the Frenet frame at the circled location.

**Adjustment of the twist parameter for closed curves.** As we discussed in the previous Section, when  $\mathcal{C}$  is closed, we want the position of the last nucleotide to coincide with the position of the first nucleotide. This implies in particular that the helix makes an integer number of turns along  $\mathcal{C}$ , which can be written

$$X(T_{\max}) \cos(\beta N_{\text{nt}}) + Y(T_{\max}) \sin(\beta N_{\text{nt}}) = X(0), \quad (8.11)$$

where  $N_{\text{nt}}$  is the last nucleotide position on the helix.

In order to meet this condition, we can tweak the parameter  $\beta$  by replacing it by a value  $\beta' \simeq \beta$  so that Condition (8.11) holds if we substitute  $\beta$  by  $\beta'$ . In order to pick the value  $\beta'$ , we first need to compute the number of turns  $N_{\text{turn}}$  that the helix makes along  $\mathcal{C}$ , according to our model. To do so, notice that the fact that  $\mathcal{C}$  is closed implies that  $Z(0) = Z(T_{\max})$ . Therefore, there exists an angle  $\theta$  associated to the rotation  $R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$  so that

$$\begin{pmatrix} X(T_{\max}) & Y(T_{\max}) \end{pmatrix} = R_\theta \begin{pmatrix} X(0) & Y(0) \end{pmatrix}.$$

This angle is given by the formula

$$\theta = \text{atan2}[\langle X(T_{\max})|Y(0)\rangle, \langle X(T_{\max})|X(0)\rangle],$$

where  $\text{atan2} : (y, x) \mapsto \arg(x + iy)$  is the two-arguments arctangent function (Figure 8.2).

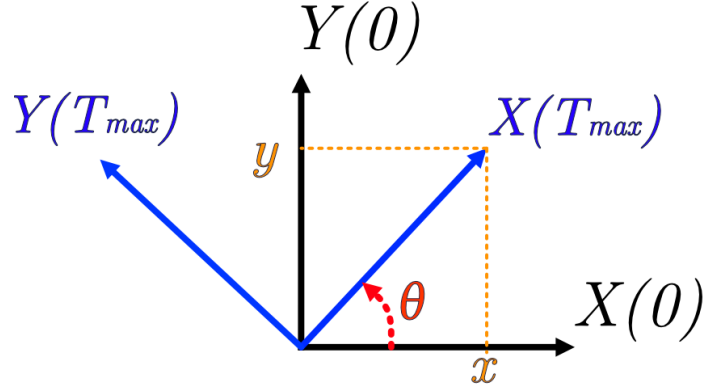


Figure 8.2: **Computation of the angle between two frames.** The values  $x$ ,  $y$  and  $\theta$  are given by:  $x = X(T_{\max}) \cdot X(0)$ ;  $y = X(T_{\max}) \cdot Y(0)$ ; and  $\theta = \text{atan2}(y, x)$

The number of turns that the helix makes along  $\mathcal{C}$  is then given by

$$N_{\text{turn}} = \frac{N_{\text{nt}}\beta + \theta}{2\pi}.$$

We can now pick the value  $\beta'$  by setting  $M = [N_{\text{turn}}]$ , where  $[\cdot]$  is the rounding operator, and

$$\beta' = \frac{2\pi M}{N_{\text{nt}}}.$$

This parameter tweaking will be applied to long helices where  $N_{\text{nt}} \geq 1000$ , which gives

$$\beta' = \frac{2\pi \left[ \frac{N_{\text{nt}}\beta + \theta}{2\pi} \right]}{N_{\text{nt}}} = \beta \pm \frac{\pi}{N_{\text{nt}}} = \beta \pm 0.18^\circ.$$

This corresponds to a helicity between 10.38 base pairs (bp) per turn and 10.50 bp per turn. For straight and long bundle of helices, it has been shown that using a value of 10.50 bp per turn for the helicity lead to an undesired twist in the resulting structure [WR11]. It is however important to note that this effect was observed on long rectilinear structures. We believe that when a helix is constrained to follow closed curved path, the constraints should also have an impact on its helicity, and that this impact is rightfully modeled by the adjustment that we make to the value of  $\beta$ .

## 8.4 Implementation of the curved P-stick model and *in silico* validation

We will now see how the curved P-stick model is implemented in ENSnano. While a straight helix  $H$  only needs to store its position and its frame, a curved helix needs to store additional values that have been constructed in the previous sections.

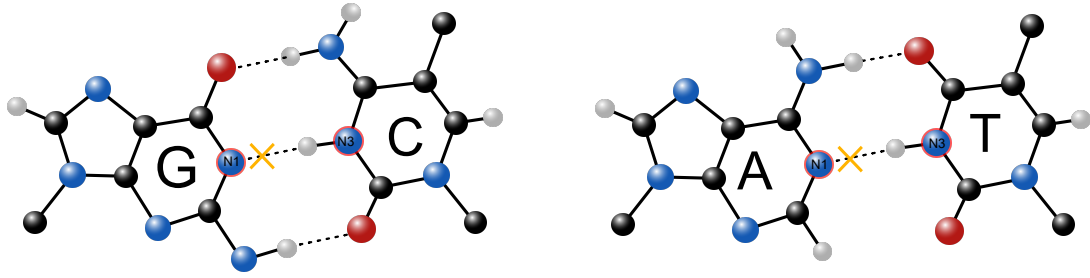


Figure 8.3: **Estimation of the axis position (yellow cross) for each base pair.** For each base pair, the axis position is estimated to be located at the position  $0.2N_3 + 0.8N_1$ , where  $N_3$  is the location of the N3 atom of the pyrimidine and  $N_1$  is the location of the  $N_1$  atom of the purine.

In ENSnano, the objects representing curved helices store

- The positions  $\{\mathcal{C}(t_i^f)\}_i$  and  $\{\mathcal{C}(t_i^b)\}_i$  constructed in Section 8.2
- The frames  $\{\mathcal{F}(t_i^f)\}_i$  and  $\{\mathcal{F}(t_i^b)\}_i$  constructed in Section 8.3
- The time parameters  $\{t_i^f\}_i$  constructed in Section 8.2
- The modified value  $\beta'$  constructed in Section 8.2

and are built using Algorithm 2. The set of positions  $\{\mathcal{C}(t_i^f)\}_i$  and  $\{\mathcal{C}(t_i^b)\}_i$ , the set of frames  $\{\mathcal{F}(t_i^f)\}_i$  and  $\{\mathcal{F}(t_i^b)\}_i$ , and the value of  $\beta'$  are used to compute the 3D positions of the nucleotides around the helix, and therefore to build a 3D representation of the helix. The set of time parameters  $\{t_i^f\}_i$  is used to build a 2D representation of the helix.

***In silico* validation of the curved P-stick model.** The curved P-stick model that we developed in this section can be used to derive the 3D positions of the nucleotides of a helix whose axis follows a curved path. In order to evaluate the quality of our model, we can compare its output with crystallography data of a naturally occurring curved DNA nanostructure.

The *nucleosome* is the smallest unit of DNA packaging in eukaryotic cells [Kor77]. It consists in a segment of DNA wrapped around an assembly of eight histone proteins. For our evaluation, we use a high-resolution crystal structure of the nucleosome complex published in [LMR<sup>+</sup>97]. We constructed an approximation  $\tilde{P}$  of the path of the axis of the DNA segment by Chebyshev interpolation of the estimated axis positions of each base pair, estimated to be located at 20% of the way the N1 atom of the purine and the N3 atom of the pyrimidine (Figure 8.3). We then superposed an STL representation of the reference crystal structure of the DNA segment [LMR<sup>+</sup>97] and the 3D position of the nucleotides of the helix with axis  $\tilde{P}$  as predicted by our model (Figure 8.4), revealing a good correspondence between our P-stick model and the crystallography data.

**Summary and conclusion.** In this section, we extended the P-stick model to curved helices. The process, summarized in Figure 8.5 rely on the discretization of the curve (Section 8.2) and on the construction of a lazily adaptive frame along the curve (Section 8.3).

---

**Algorithm 2 Construction of an object representing a curved helix**

---

**Input:** A real number  $T_{\max}$  and a curve  $\mathcal{C} : [0, T_{\max}] \rightarrow \mathbb{R}^3$ .

▷ Adjustment of the geometric parameters  
 $\tilde{s} \leftarrow \text{interpolate\_curvilinear\_abscissa}(\mathcal{C})$   
 $\tilde{\sigma} \leftarrow \text{interpolate\_inverse\_curvilinear\_abscissa}(\mathcal{C})$   
 $\tilde{L} \leftarrow s(T_{\max})$   
**if**  $\mathcal{C}$  is closed **then**  
     $N \leftarrow \left\lceil \frac{\tilde{L}}{\Delta} \right\rceil$   
     $\Delta \leftarrow \frac{\tilde{L}}{N}$   
     $M \leftarrow \left\lceil \frac{N}{2\pi} \beta \right\rceil$   
     $\beta' \leftarrow \frac{2\pi M}{N}$   
**else**  
     $\beta' \leftarrow \beta$  ▷  $\Delta$  keeps its usual value  
**end if**  
▷ Discretization of the curve  
 $\text{current\_time} \leftarrow 0$   
 $\text{time\_points} \leftarrow [0]$  ▷ The set  $\{t_i\}$   
 $\text{positions\_forward} \leftarrow [\mathcal{C}(0)]$  ▷ The set  $\{\mathcal{C}(t_i^f)\}_i$   
 $\text{positions\_backward} \leftarrow []$  ▷ The set  $\{\mathcal{C}(t_i^b)\}_i$   
 $\text{current\_frame} \leftarrow \text{init\_frame}(\mathcal{C})$   
 $\text{frames\_forward} \leftarrow [\text{current\_frame}]$  ▷ The set  $\{\mathcal{C}(t_i^f)\}_i$   
 $\text{frames\_backward} \leftarrow []$  ▷ The set  $\{\mathcal{C}(t_i^b)\}_i$   
 $\text{current\_abscissa} \leftarrow 0$   
 $\text{next\_abscissa\_forward} \leftarrow \Delta$   
 $\text{next\_abscissa\_backward} \leftarrow I$   
**while**  $\text{current\_time} \leq T_{\max}$  **do**  
     $\text{next\_point\_is\_on\_forward} \leftarrow \text{next\_abscissa\_forward} \leq \text{next\_abscissa\_backward}$   
     $\text{next\_abscissa} \leftarrow \min(\text{next\_abscissa\_forward}, \text{next\_abscissa\_backward})$   
     $\text{current\_time} \leftarrow \tilde{\sigma}(\text{next\_abscissa})$   
     $\text{current\_frame} \leftarrow \text{itterative\_frame}(\text{current\_frame}, \text{current\_time}, \mathcal{C})$   
    **if**  $\text{next\_point\_is\_on\_forward}$  **then**  
         $\text{time\_points.append}(\text{current\_time})$   
         $\text{positions\_forward.append}(\mathcal{C}(\text{current\_time}))$   
         $\text{frames\_forward.append}(\text{current\_frame})$   
         $\text{next\_abscissa\_forward} += \Delta$   
    **else**  
         $\text{positions\_backward.append}(\mathcal{C}(\text{current\_time}))$   
         $\text{frames\_backward.append}(\text{current\_frame})$   
         $\text{next\_abscissa\_backward} += \Delta$   
    **end if**  
**end while**

---

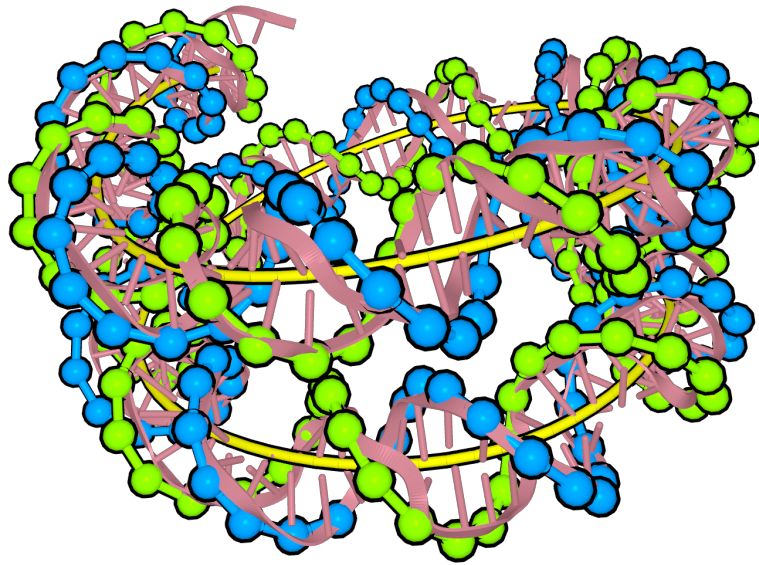


Figure 8.4: **Superposition of crystallography data (pink) with the 3D positions of the nucleotides as predicted by the curved P-stick model.** The axis of the helix (in Yellow) was constructed by Chebyshev interpolation of the axis location of each base pair (Figure 8.3). The positions of the nucleotide on the green a blue strand are those predicted by the curved P-stick model.

The model was validated *in silico* by comparing its prediction with crystallography data of the nucleosome. In the next sections, we will validate the model experimentally by designing various classes of curved DNA origamis.

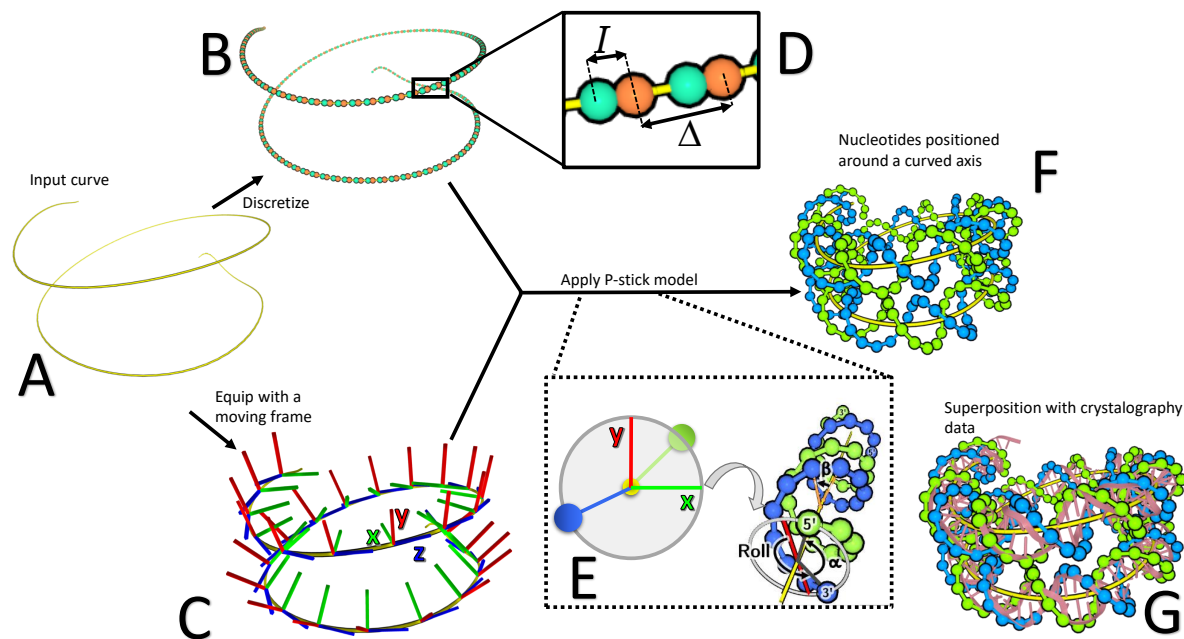


Figure 8.5: **Computation of the 3D position of the nucleotides around the curve axis.** (A): The input of the discretization algorithm is a curve  $C : \mathbb{R} \rightarrow \mathbb{R}^3$  (B) The first step to position nucleotides on a curved axis is to split the curve into segment of constant curvilinear length. (C) The curve is the equipped with an iterative frame whose  $z$  axis is always collinear to the curve's tangent. (D) Illustration of the inclination ( $I$ ) and rise ( $\Delta$ ) parameters. The inclination is the curvilinear distance along the axis between the forward and the backward nucleotides in a pair. The rise is the curvilinear distance along the axis between two consecutive nucleotides on the same strand. (E) Once the curve is discretized and equipped with a moving frame, the P-stick model can be applied. The nucleotides are positioned in the  $xy$  plane of the frame. (F): Positions of the nucleotides around a curved path in ENSnano's curved DNA model (G): Superposition of the curved helix as computed by ENSnano and crystallography data extracted from [LMR<sup>+</sup>97]

## Chapter 9

# Design of curved bundle of helices

### 9.1 Design principles

In this Chapter, we design and assemble DNA origami made of curved bundles of helices. The design of these bundles rely on the notion of grid defined in Chapter 4. Bundles are created by positioning helices on a grid, and having the grid follow a curved path (Figure 9.1). As we have seen in the previous chapter, the position of a straight helix located at position  $(i, j)$  on a grid  $\mathcal{G}$  is given by the formula

$$\mathcal{O}_H = \mathcal{O}_G + \mathcal{F}_G \cdot \Lambda_G(i, j),$$

where  $\mathcal{O}_G$  is the origin of  $\mathcal{G}$ ,  $\mathcal{F}_G$  is the frame associated to  $\mathcal{G}$ , and  $\Lambda_G : \mathbb{Z}^2 \rightarrow \mathbb{R}^3$  is the lattice associated to  $\mathcal{G}$  (see Section 4.3). If we want the grid to follow a curved path  $\mathcal{P}_G : [a, b] \rightarrow \mathbb{R}^3$ , we obtain a curved path  $\mathcal{P}$  for the axis of the helix:

$$\mathcal{P}(t) = \mathcal{P}_G(t) + \mathcal{F}_G(t) \cdot \Lambda_G(i, j), \quad (9.1)$$

where  $\mathcal{F}_G : [a, b] \rightarrow \text{SO}_3(\mathbb{R})$  is now the moving frame associated to  $\mathcal{P}_G$  constructed with the lazily adaptive process described in Section 8.3. We say the Equation (9.1) defines a *translated curve*.

**Discretization of a translated curve** Looking at Equation (9.1), we notice that, in order to construct the path  $\mathcal{P}$ , one needs to construct a moving frame  $\mathcal{F}_G$  for the path  $\mathcal{P}_G$ . This creates a problem because in Section 8 we did not construct a *continuous* moving frame  $\mathcal{F} : [0, T_{\max}] \rightarrow \text{SO}_3(\mathbb{R})$ , but instead only constructed values  $\mathcal{F}(t)$  for time the parameters  $t \in \{t_i^f\} \cup \{t_i^b\}$  where it was needed.

However, the “curved translation” operation that is defined in Equation (9.1) does not preserve the curvilinear abscissa, and therefore, the time parameters corresponding to nucleotide positions on the translated curve cannot be known when constructing the frame  $\mathcal{F}_G$ .

To address this issue, we first construct a continuous frame  $\tilde{\mathcal{F}}_G : [0, T_{\max}] \rightarrow \text{SO}_3(\mathbb{R})$  by first creating a discrete set  $\{\mathcal{F}_G(t_i)\}_i$  and interpolating it.

More precisely, for  $1 \leq i, j \leq 3$ , we create a Chebyshev interpolation  $\tilde{a}_{i,j}$  of the set of point-values  $\{(t_k, (\mathcal{F}_G(t_k))_{ij})_k$ . This gives us a parameterized matrix

$$\tilde{A} : t \mapsto \tilde{A}(t) = (\tilde{a}_{ij}(t))_{i,j} \in \mathcal{M}_3(\mathbb{R}). \quad (9.2)$$



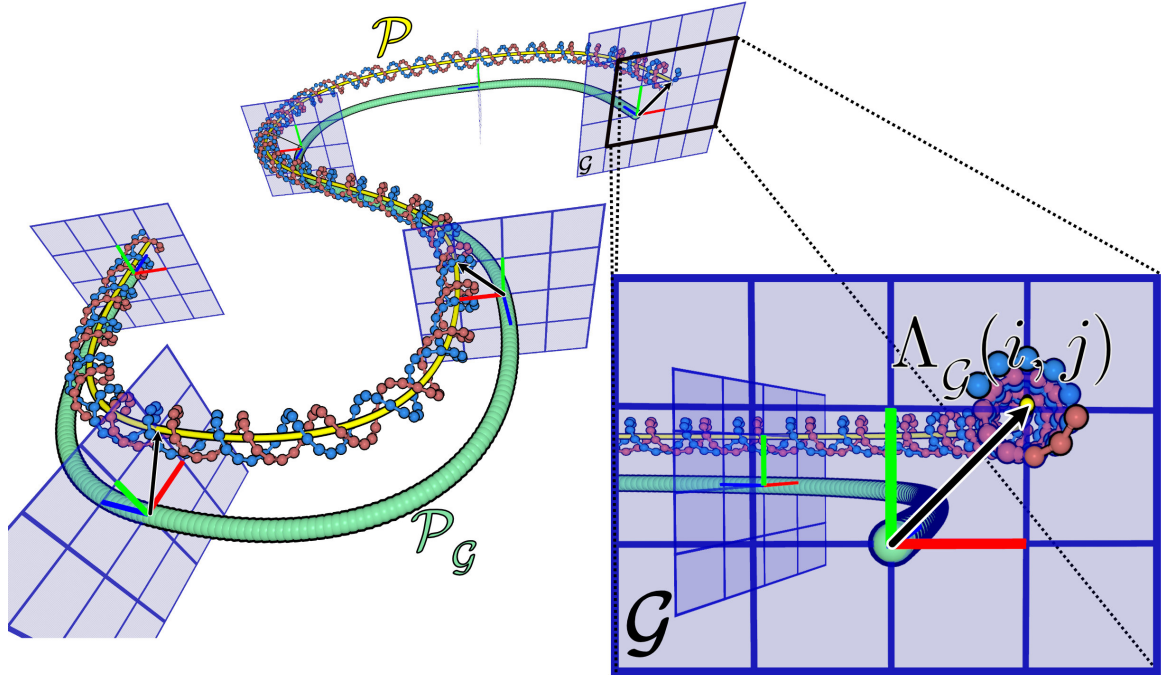


Figure 9.1: **A helix whose axis is a translated curve.** The helix is located at position  $(i, j)$  of the grid  $\mathcal{G}$  following the path  $\mathcal{P}_{\mathcal{G}}$  equipped with the moving frame  $\mathcal{F}_{\mathcal{G}}$ . The path followed by the axis of the helix is given by equation  $\mathcal{P}(t) = \mathcal{P}_{\mathcal{G}}(t) + \mathcal{F}_{\mathcal{G}}(t) \cdot \Lambda_{\mathcal{G}}(i, j)$ .

For a given  $t \in [0, T_{\max}]$ ,  $\tilde{A}(t)$  is, in general, not a right-handed orthonormal matrix. We therefore need to “orthonormalize” it. To do so, we apply a similar strategy as in Section 8.3: We call  $\mathbf{x}(t), \mathbf{y}(t)$  and  $\mathbf{z}(t)$  the column vectors of  $\tilde{A}(t)$ . The column vectors  $\tilde{X}_{\mathcal{G}}(t), \tilde{Y}_{\mathcal{G}}(t)$  and  $\tilde{Z}_{\mathcal{G}}(t)$  of  $\tilde{\mathcal{F}}_{\mathcal{G}}(t)$  are then iteratively constructed from  $\mathbf{x}, \mathbf{y}$  and  $\mathbf{z}$  by the formulas

$$\begin{aligned}\tilde{Z}_{\mathcal{G}}(t) &= \frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|_2} \\ \tilde{Y}_{\mathcal{G}}(t) &= \frac{\tilde{Z}_{\mathcal{G}}(t) \wedge \mathbf{x}(t)}{\|\tilde{Z}_{\mathcal{G}}(t) \wedge \mathbf{x}(t)\|_2} \\ \tilde{X}_{\mathcal{G}}(t) &= \tilde{Y}_{\mathcal{G}}(t) \wedge \tilde{Z}_{\mathcal{G}}(t).\end{aligned}$$

**Computer optimization.** The computational cost of constructing  $\tilde{\mathcal{F}}_{\mathcal{G}}$  is affordable but not negligible ( $\simeq 10\text{ms}$  on an Intel Core i9-10885H). In the current implementation, this is compensated by the fact that translated curves are discretized using a modified version of Algorithm 2 where instead of constructing an iterative frame for  $\mathcal{P}$ , we equip it with  $\tilde{\mathcal{F}}_{\mathcal{G}}$ . This allows the discretization procedure of the curve to be run in parallel as the computation for each point can be multithreaded.

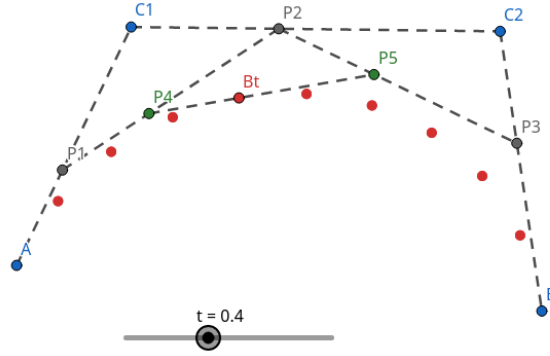


Figure 9.2: **Construction of a Bézier curve by successive interpolations.** The red dots show the positions of  $\mathcal{B}(t)$  for  $t = 0.1, 0.2, \dots, 0.9$ .

## 9.2 Design of curved bundles in ENSnano

Currently, ENSnano offers an interface to design bundle of helices following a path that belongs to a specific class of curves that we call *piecewise cubic Bézier curve*.

Bézier curves form a special class of polynomials that is widely used in computer graphics [Mor99, Chapter 15].

**Definition 5** (Cubic Bézier curve). Let  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}_1$  and  $\mathbf{c}_2$  be four points of  $\mathbb{R}^n$ . The *cubic Bézier curve* defined by these four points is the parametric curve

$$\mathcal{B} : t \mapsto \mathcal{B}(t),$$

where  $\mathcal{B}(t)$  is constructed by the following repeated interpolations process as illustrated by Figure 9.2:

1. Let  $P_1, P_2$  and  $P_3$  be  $(1-t, t)$ -linear interpolations of  $\mathbf{a}$  and  $\mathbf{c}_1$ , of  $\mathbf{c}_1$  and  $\mathbf{c}_2$ , and of  $\mathbf{c}_2$  and  $\mathbf{b}$ :

$$P_1 = (1-t)\mathbf{a} + t\mathbf{c}_1$$

$$P_2 = (1-t)\mathbf{c}_1 + t\mathbf{c}_2$$

$$P_3 = (1-t)\mathbf{c}_2 + t\mathbf{b}$$

2. Let  $P_4$  and  $P_5$  be  $(1-t, t)$ -linear interpolations of  $P_1$  and  $P_2$ , and of  $P_2$  and  $P_3$ :

$$P_4 = (1-t)P_1 + tP_2$$

$$P_5 = (1-t)P_2 + tP_3$$

3. Define  $\mathcal{B}(t)$  as a  $(1-t, t)$ -linear interpolation of  $P_4$  and  $P_5$ :

$$\mathcal{B}(t) = (1-t)P_4 + tP_5$$

The points  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}_1$  and  $\mathbf{c}_2$  are called the *control points* of  $\mathcal{B}$ .

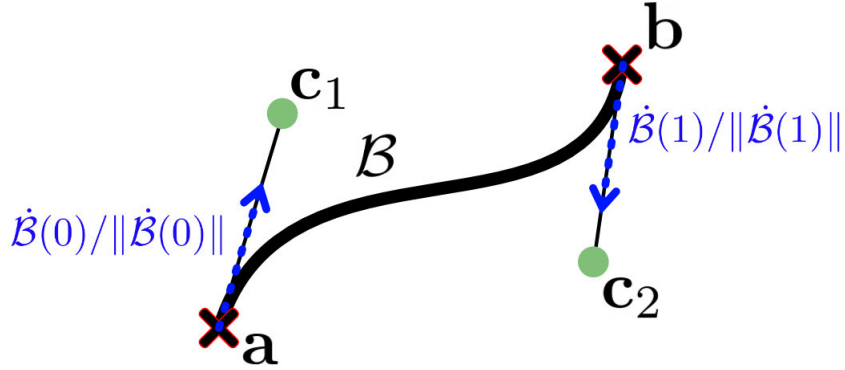


Figure 9.3: **A cubic Bézier curve defined by the control points  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}_1$  and  $\mathbf{c}_2$ .** Notice that  $\mathbf{a}$  and  $\mathbf{b}$  define the extremities of the curve, while  $\mathbf{c}_1$  and  $\mathbf{c}_2$  determine the direction of the curve's tangent at its extremities.

**Property 2.** It can be verified that

$$\mathcal{B}(t) = (1-t)^3\mathbf{a} + 3(1-t)^2t\mathbf{c}_1 + 3(1-t)t^2\mathbf{c}_2 + t^3\mathbf{b}.$$

As a consequence, the Bézier curve defined by  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}_1$  and  $\mathbf{c}_2$  verifies the following properties illustrated by Figure 9.3.

- $\mathcal{B}(0) = \mathbf{a}$
- $\mathcal{B}(1) = \mathbf{b}$
- $\dot{\mathcal{B}}(0) = 3 \cdot (\mathbf{c}_1 - \mathbf{a})$
- $\dot{\mathcal{B}}(1) = 3 \cdot (\mathbf{b} - \mathbf{c}_2)$

The manipulation of curved path seen as a concatenation of Bézier curves can thus be made quite intuitive. By offering the possibility to move the control points, the user can easily set the origin/end of the curve (by moving  $\mathbf{a}$  and  $\mathbf{b}$ ) and the tangent at the origin/end of the curve (by moving  $\mathbf{c}_1$  and  $\mathbf{c}_2$ ).

**Definition 6.** Let  $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_k$  and  $\mathbf{c}_0^+, \mathbf{c}_1^-, \mathbf{c}_1^+, \dots, \mathbf{c}_{k-1}^-, \mathbf{c}_{k-1}^+, \mathbf{c}_k^-$  be  $3k+1$  points of  $\mathbb{R}^n$  for some integer  $n$ .

These points define  $k$  cubic Bézier curves  $\mathcal{B}_1, \dots, \mathcal{B}_k$  where for  $1 \leq i \leq k$ ,  $\mathcal{B}_i$  is the cubic Bézier curve defined by the control points  $\mathbf{a}_{i-1}, \mathbf{a}_i, \mathbf{c}_{i-1}^+$  and  $\mathbf{c}_i^-$ .

If, for all integer  $1 \leq i \leq k-1$ , the vectors  $\mathbf{c}_{i-1}^+ - \mathbf{a}_i$  and  $\mathbf{a}_i - \mathbf{c}_i^-$  are collinear<sup>1</sup>, then these curve can be concatenated (see Figure 9.4) into a single *piecewise cubic Bézier curve*  $\mathcal{B} : [0, k-1] \rightarrow \mathbb{R}^n$  defined by

$$\mathcal{B}(t) = \begin{cases} \mathcal{B}_{[t]}(\{t\}) & \text{if } t < k-1 \\ \mathbf{a}_{k-1} & \text{if } t = k-1, \end{cases}$$

<sup>1</sup>This requirement is not essential but ensures that there is no sudden change of direction in the concatenated curve.

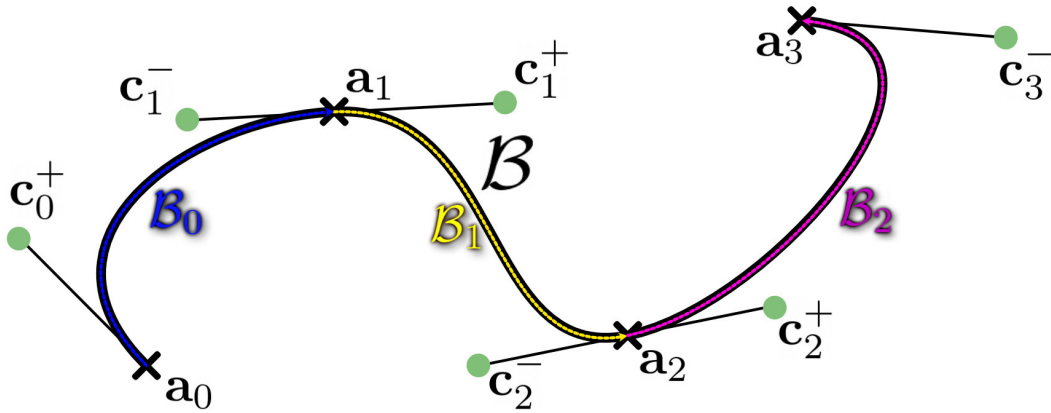


Figure 9.4: **A piecewise Bézier curve.** The curve  $\mathcal{B}$  is the concatenation of the curves  $\mathcal{B}_0, \mathcal{B}_1$  and  $\mathcal{B}_2$

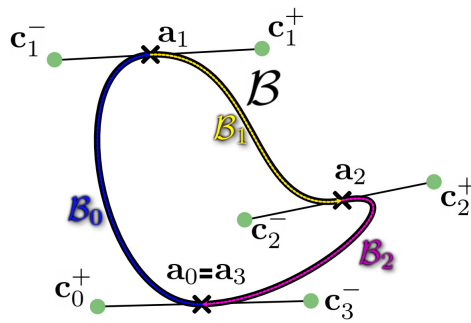


Figure 9.5: **A closed piecewise Bézier curve**

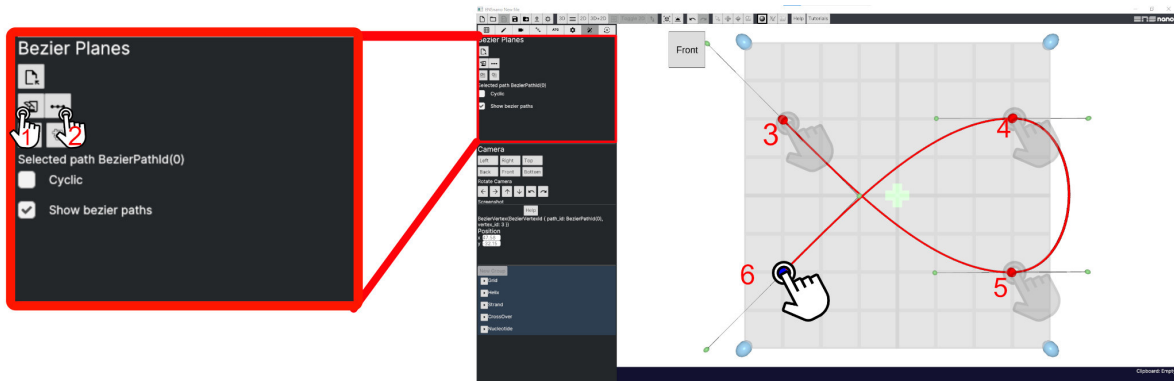
where  $[t]$  is the integer part of  $t$ ,  $\{t\} = t - [t]$  is the fractional part of  $t$ . The curve  $\mathcal{B}$  is everywhere left- and right-differentiable, and is everywhere continuously differentiable if for all  $1 \leq i \leq k - 1$ ,  $\mathbf{c}_{i-1}^+ - \mathbf{a}_i = \mathbf{a}_i - \mathbf{c}_i^-$ .

Notice that, in the case where  $\mathbf{a}_0 = \mathbf{a}_{k+1}$ ,  $\mathcal{B}$  is closed (Figure 9.5).

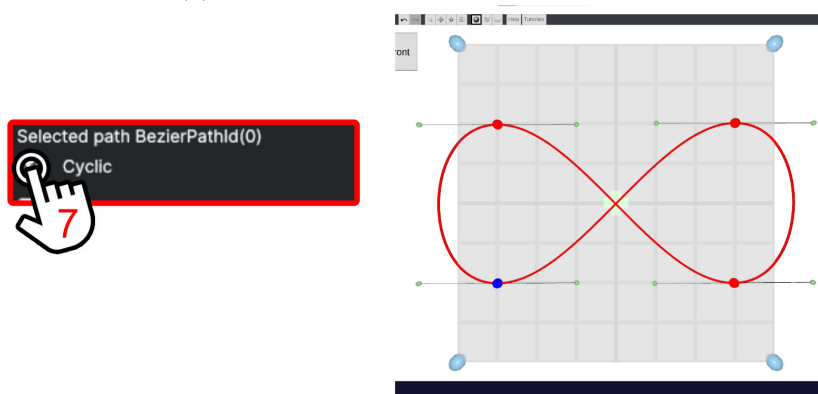
**Drawing piecewise Bézier curve in ENSnano.** ENSnano offers a tool to draw piecewise Bézier curves. To do so, user create a *Bézier sheet* that is a plane on which vertices of a *Bézier path* can be positioned (Figure 9.6). Bézier paths define piecewise Bézier curves: the vertices of the paths correspond to the extremities of the pieces of the curve, *i.e.* the points  $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_k$  of Definition 6. The other control points are automatically given a default position that depends on the position of the vertices of the path. The position of these other control points can also be modified at will for precise specification of the curve.

Finally, it is possible to close the curve by ticking a checkbox in the GUI.

**Creation of curved bundle of helices in ENSnano.** Once the desired Bézier path has been drawn, it can be equipped with a grid, either squared or hexagonal by clicking on the



(a) Create the plane and draw the path.



(b) Make the path cyclic.

Figure 9.6: **Drawing Bézier paths in ENSnano.** This figure presents the sequence of actions to perform in order to draw a Bézier path in ENSnano. a: Create the plane (1) and select the Bézier path action mode (2), then position the control points by clicking on the plane (3-6). b: To close/open the path, tick/untick the “cyclic” checkbox (7).

corresponding button in the GUI interface. The created grid is then displayed in its local frame on every vertex of the Bézier path. Helices can then be created on this grid, by using the Helix Creation action mode, like for straight bundle of helices (Figure 9.7).

Each instance of the grid on a vertex of the path can be translated and rotated to modify the 3D position and tangent 3D direction of the curve on the corresponding vertex (Figure 9.8). This can for example be useful when designing self-overlapping designs.

**2D embedding of curved DNA helices.** As we saw in Chapters 3 and 4, many DNA nanostructure design software, including ENSnano, use an abstract 2D representation of DNA helices as arrays with two rows, where each row represents one strand of the helix. As we discussed, this representation, first introduced in caDNA<sub>no</sub> [DMT<sup>+</sup>09], is particularly ergonomic for editing designs made of parallel helices.

In order to create a usable abstract 2D representation of curved DNA helices, we need to take into account the fact that, in the curved region of a DNA nanostructure, the inner helices are typically shorter than the outer ones [DDS09]. This makes it more challenging to

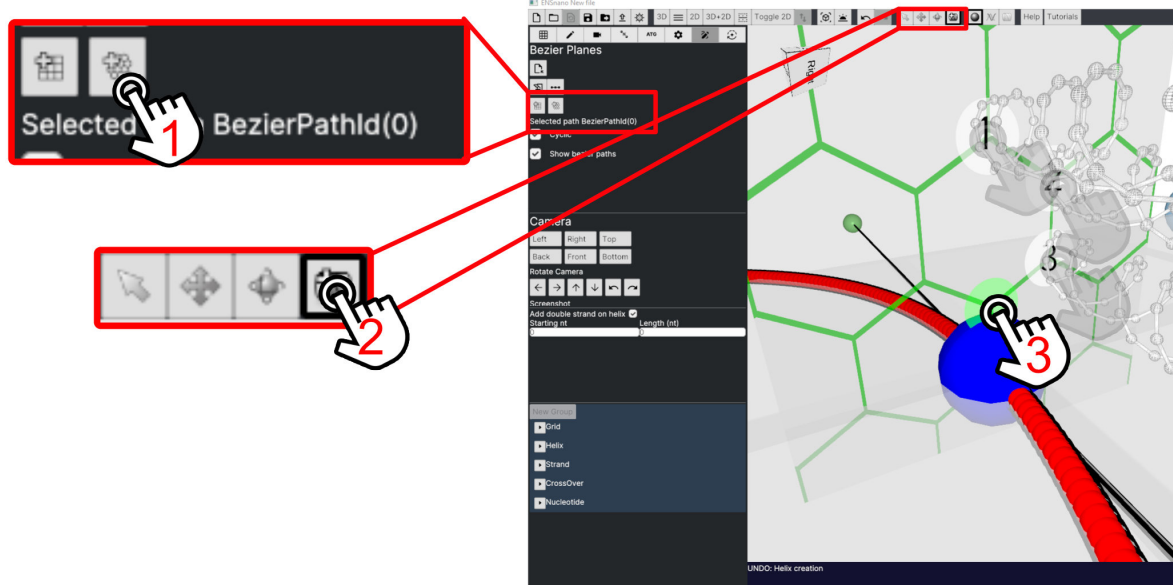


Figure 9.7: **Creating helices that follow a curved path.** Once the path has been created (Figure 9.6), it can be equipped with a grid (1). Then helices can be created on the grid by choosing the “add helix” action mode (2) and clicking on the desired grid positions (3).

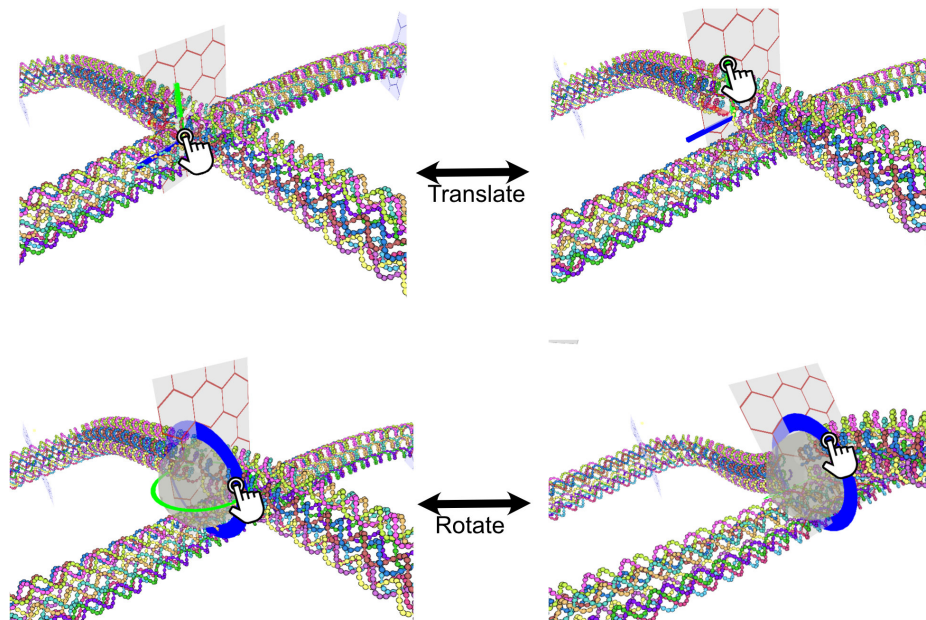


Figure 9.8: **Translating/Rotating the grids on the paths.** The grids on a Bézier path can be translated or rotated like regular grids, to modify the positions and tangents of the helices at a specific control point.

represent these helices in 2D because the nucleotide positions continuously drift with respect to one another. The classical solution consists in adding insertion and deletion marks in **caDNA**<sub>no</sub>: insertions to extend the outer part of the double helices, deletions to contract their inner parts. If this approach allows to keep the 2D representation of the double helices synchronized, it puts however the burden on the designer to accurately keep track of the correspondence between the desired curvature and the shift between the helices. Moreover, it requires a manual input from the user who needs to place those deletions manually, hopping for the best without feedback.

We developed a new approach that relies on our P-stick model for curved DNA. Each helix follows a path  $\mathcal{C} : [0, T_{\max}] \rightarrow \mathbb{R}^3$ . This parametrization can be chosen so that for adjacent helices share the same parameterization. In that case, each  $t \in [0, T_{\max}]$  corresponds to a given frontal cut  $S(t)$  of the design (see Figure 9.9). The goal is then to ensure that each front cut is mapped to vertically aligned cells in the 2D view, (as is the case in **caDNA**<sub>no</sub> for flat designs). To do so, the width of the cell representing each nucleotide at position  $n$  depends linearly on the value  $w_n = t_{n+1}^f - t_n^f$ , where  $t_i^f$  is defined in Equation (8.4) in Section 8.1.

### 9.3 Experimental validation: design and assembly of a curved bundle with the shape of a looped square

In order to validate our design method for curved bundle of helices, we designed and assemble a six-helices bundle with the shape of a looped square (Figure 9.10).

#### 9.3.1 Description of the design and its variants

The looped square shape was chosen to demonstrate the capabilities of **ENS**<sub>nano</sub> because it is a closed, self-overlapping shape that should easily be identifiable on AFM or TEM images. This shape also exhibits a non-uniform curvature (Figure 9.11) that would have been difficult to design with methods that require tracking the curvature to adjust the crossover pattern.

In order to test whether the curvature induced locally by the crossovers was enough to constrain the origami to the desired shape, several variants of the design were tested. These variants differ by the presence or absence of two kinds of linking staples.

**Junctions staples.** The desired squared-loop shape passes over itself at four locations (Figure 9.12). In the most constrained variant of the design, for each of these four intersections, there is a pair of staples linking the two overlapping parts of the bundle. These linking staples are called *junction staples*.

**Closing staples.** The desired squared-loop shape is closed. In the most constrained variants of the design, *linking staples* connect both ends of the bundle to close it (Figure 9.13). Variants of the design that include those staples are called *closed*. In *open* variants of the design, these staples are truncated so that the two ends of the bundle may be disconnected, and a quadruplet of T is added on both ends to prevent stacking.

#### 9.3.2 Annealing and imaging of the origami

Nicolas Schabanel (NS) annealed the four variants of the looped square origami and produced atomic force microscopy (AFM) images of non purified samples (Figure 9.14).

NS annealed the origami in 1x TAE with 12.5 mM MgCl<sub>2</sub> using 5nM of scaffold and 50nM of staples. The annealing ramp was  $-1^{\circ}\text{C}/\text{min}$  from  $65^{\circ}\text{C}$  to  $45^{\circ}\text{C}$ .

Allan Mills (AM) annealed the four variants of the looped square origami, and produced transmission electron microscopy (TEM) images of samples purified by electrophoresis on a 1% agarose gel (Figure 9.15).

AM annealed the origami in a Folding buffer (5mM Tris, 1mM EDTA, pH=8) with 12mM Mg using 100nM of scaffold and 500nM of staples. The folding ramp was: 15min at  $65^{\circ}\text{C}$  followed by a descent from  $60^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  at  $-1^{\circ}\text{C}/\text{hour}$ .

## Results

**Closed variant with junction staples (CJ).** The AFM images of the closed variant of the looped square origami with junction (Figure 9.14a) reveal a variety of shapes that includes the desired shape. The desired shape can be seen much more consistently in the TEM images of a purified sample of the same origami (Figure 9.15a). The TEM images of the origami allow us to claim that they have the desired shape. The electrophoresis gel also show that the origami is produced with a good yield.

**Open variant with junction staples (OJ).** The AFM images of the open variant of the design, with junctions (Figure 9.14b) reveal several instances of an open shape with four loops. Similar results can be observed on TEM images of the same variant of the design (Figure 9.15b). This suggests that closing staples are required to enforce the closing of the shape.

**Closed variant without junction staples (CnoJ).** In both AFM (Figure 9.14c) and TEM (Figure 9.15c) several shapes can be seen consistently. These shapes can be obtained with plumber pipes linked with 3/4-turns corners (Figure 9.16). This is a sign that even though the crossovers are enough to locally enforce bending in the 6-helices bundle, they cannot constrain the orientation of those turns. Note that in some of the unwanted shapes, the torsion is quite high and this seems to have been compensated in the origami by the absence of some of the staples in parts of the origami as can be observed in Figure 9.16. This suggests that the crossover scheme designed with ENSnano enforces a curvature in the corners of the design that is locally correct, but not constrained enough to control the overall shape of the structure. Junction staples, like closing staples seem to also be required to enforce the folding of the structure in the intended shape.

**Open variant without junction staples (OnoJ).** TEM images of the open variant of the design, without junction staples (Figure 9.15d) reveal a variety of open, curved bundle. In regard to the above results this variety of shape is surprising. Still, we do observe a curvature induced by the crossover pattern.



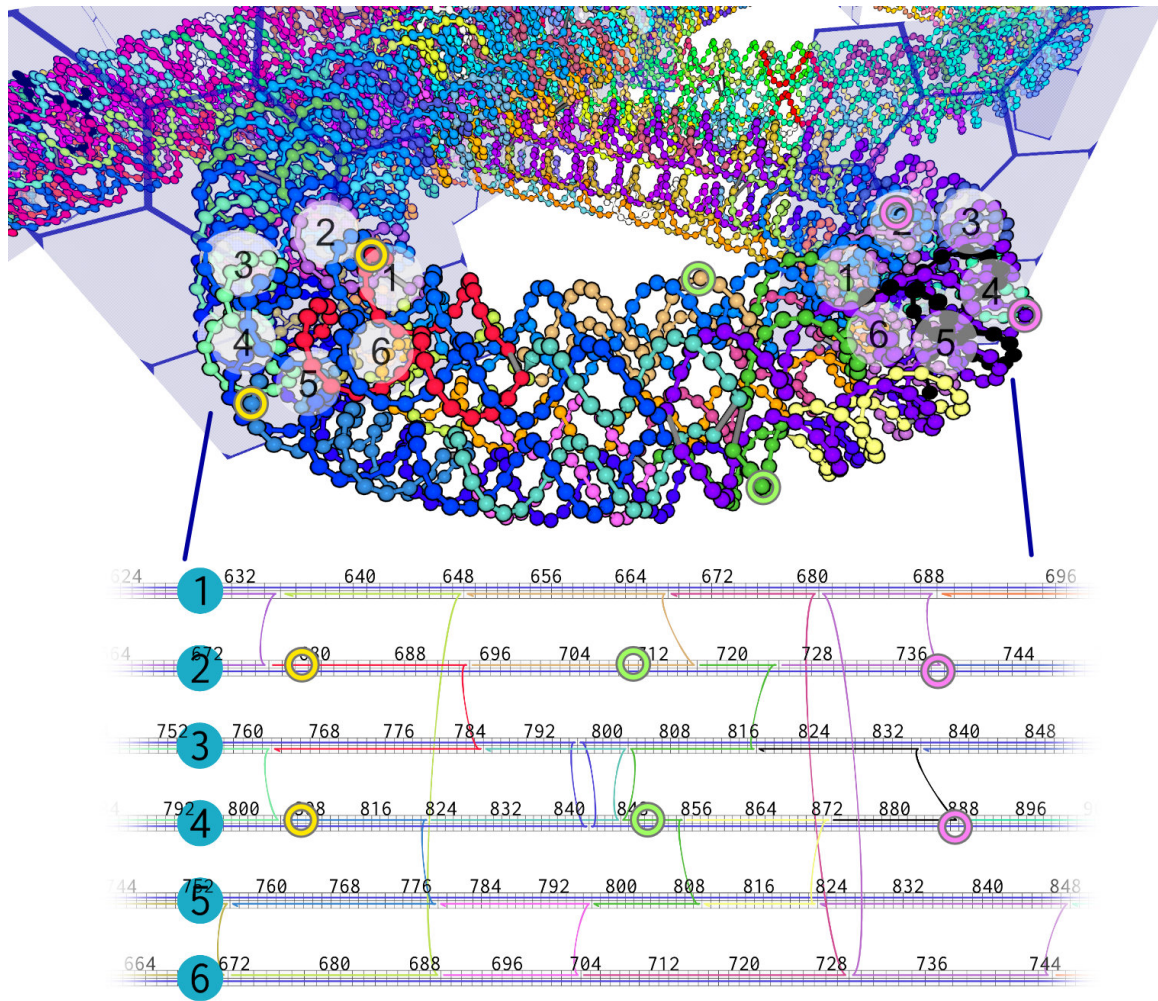


Figure 9.9: **ENSnano's 2D deletion/insertion-free adaptive representation of curvy parallel double-helices** as arrays whose cell widths are automatically adjusted to match the progress of each double helix. Notice that between these two front cuts, the outermost helix (helix 4) progresses by about 100 nucleotides, while the innermost helix (helix 1) progresses by about 70 nucleotides only.

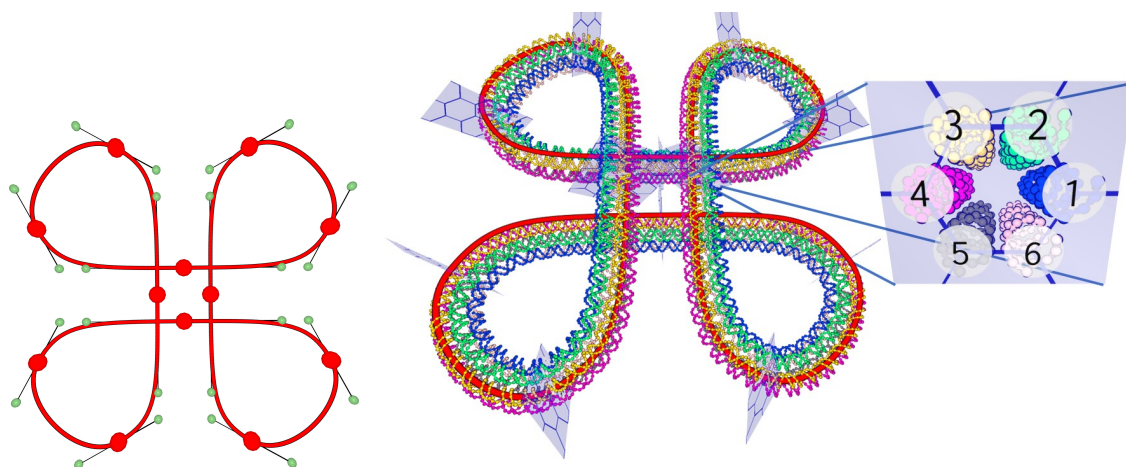


Figure 9.10: **Design of the six-helices bundle with the shape of a looped square.** Left: The Bézier path followed by the bundle as drawn in ENSnano. Right: The helices of the bundle. The helices are numbered from 1 to 6. These numbers are used in the 2D representation of the design and in Figure 9.11

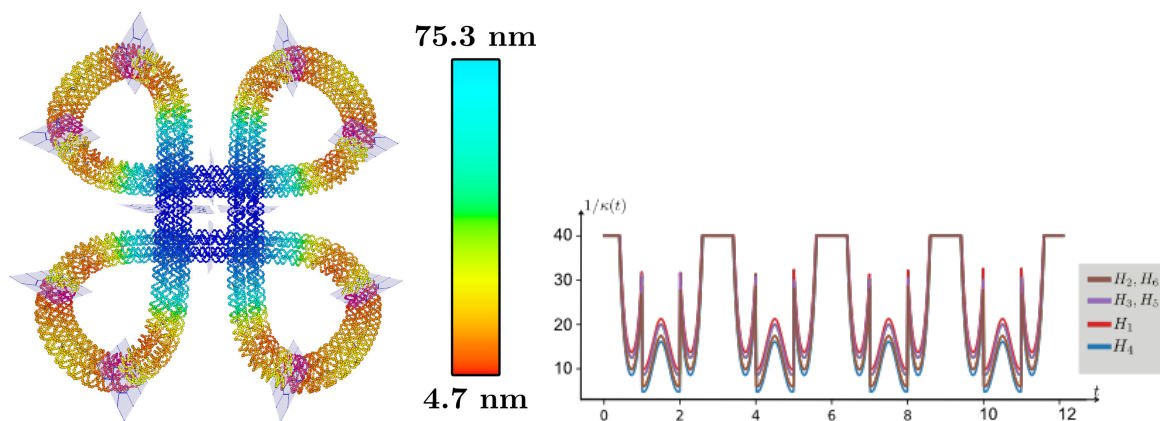


Figure 9.11: **The curvature of the looped square bundle.** Left: Heat map of the bundle's radius of curvature. The dark blue color corresponds to region where the bundle is locally almost straight, and where the radius of curvature is therefore nearly infinite. Right: Plot of the radius of curvature of each helix of the bundle. Values above 40 nm are truncated. Note that there are two pairs of helices that share the same curvature, therefore only four distinct graphs are plotted.

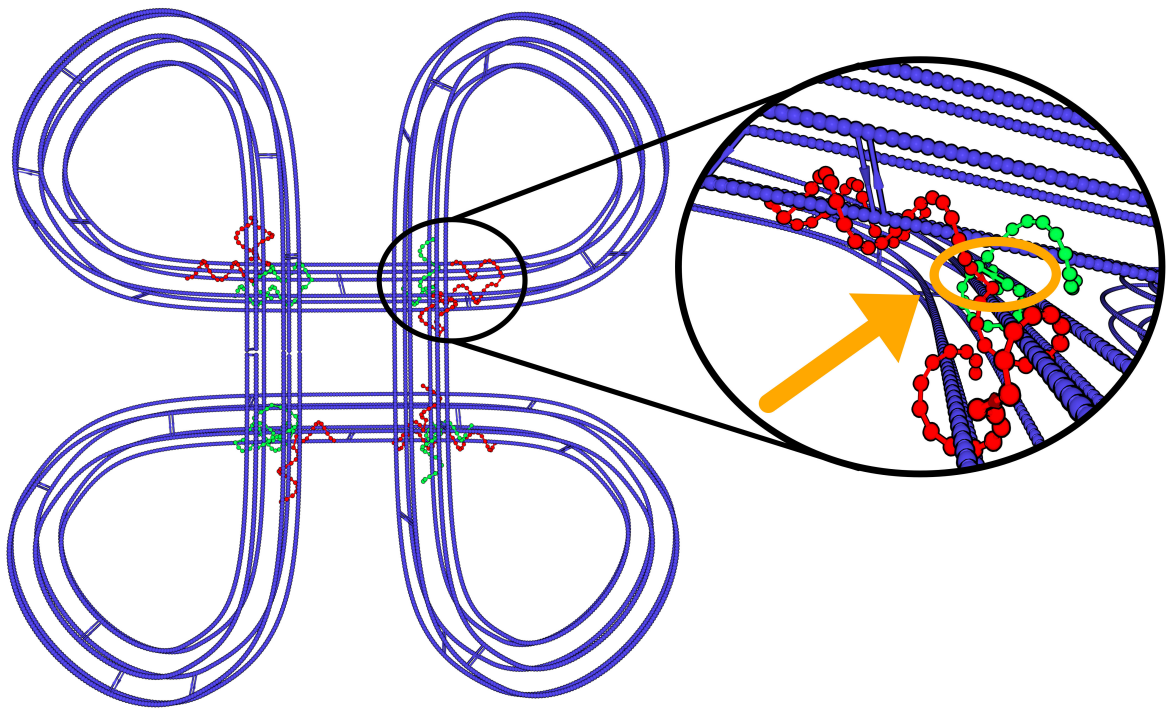


Figure 9.12: **Positions of the junction staples.** At each of the four positions where the 6 helices bundle passes above itself, two staples (colored in green and red) connect the overlapping parts. The junctionless variants are obtained by cutting the crossovers connecting the two parts, such as the ones circled in orange. For clarity, only the (flattened) scaffold (in blue) and the junction staples are shown.

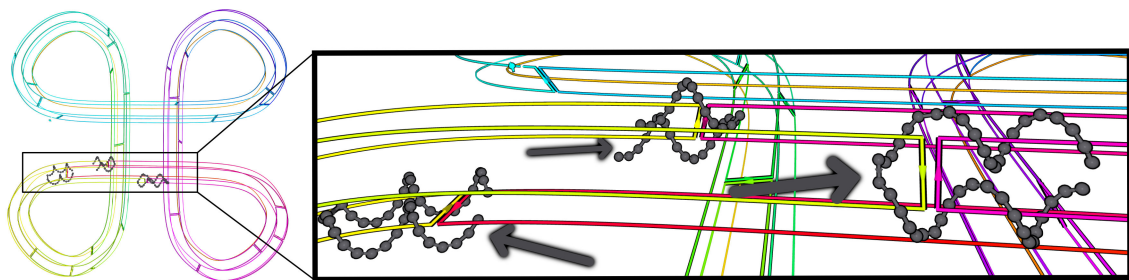
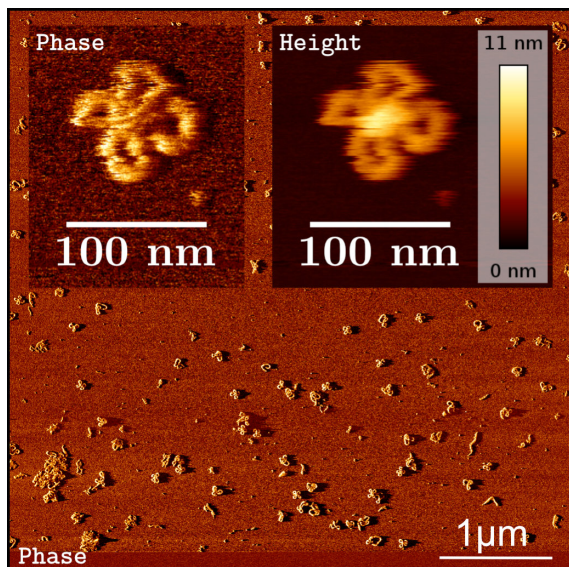
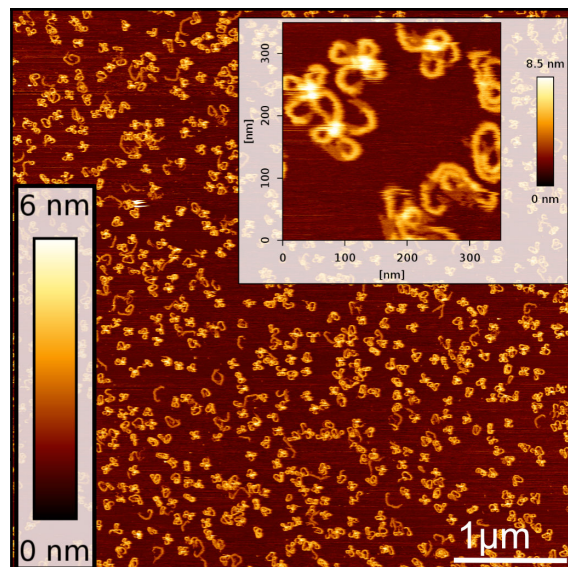


Figure 9.13: **Positions of the three closing staples (in gray) binding both ends of the 6 double helices together.** For clarity, only the scaffold and the closing staples are shown. The scaffold is displayed using a “rainbow” color gradient to show the disconnected parts: The closing staples links the red and pink regions to the yellow and green ones.

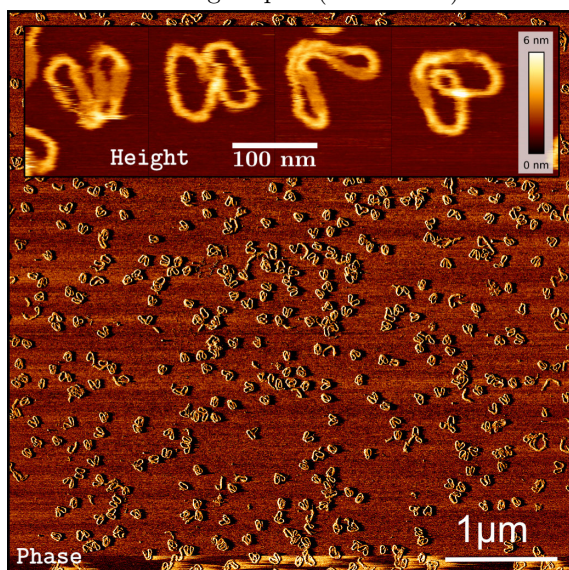




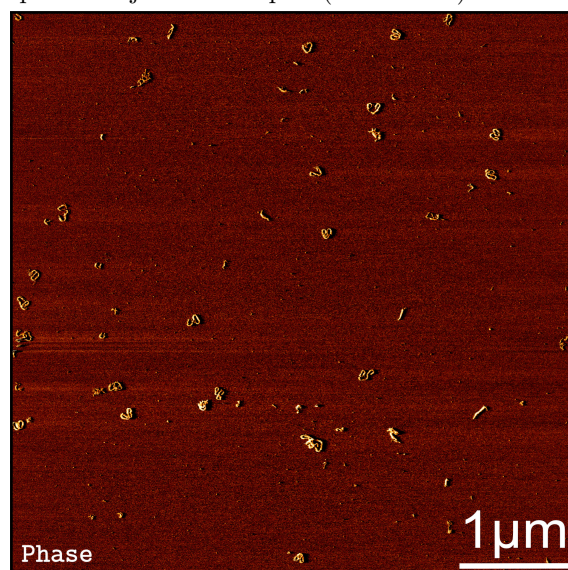
(a) AFM images of the looped square origami closed with linking staples (CJ variant).



(b) AFM images of the looped square origami open with junction staples (OJ variant).



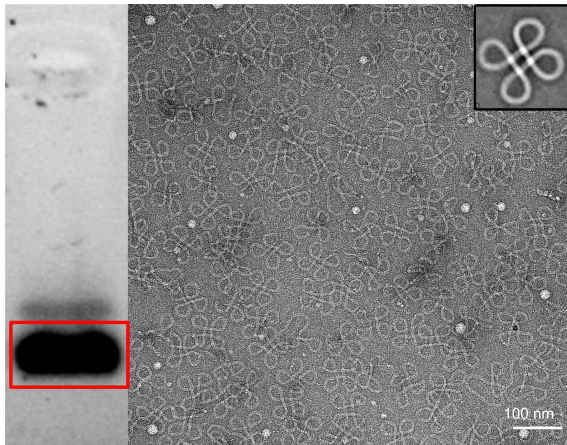
(c) AFM images of the closed and junctionless variant of the looped square origami (CnoJ variant).



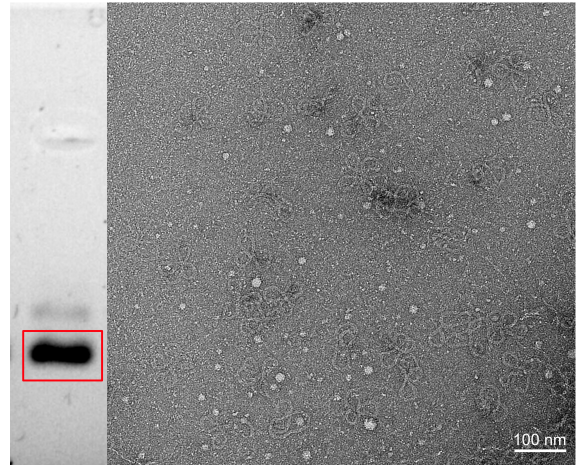
(d) AFM images of the open and junctionless variant of the looped square origami (OnoJ variant).

Figure 9.14: **AFM images of unpurified samples of the looped square origami** ( $2.5 \mu\text{L}$  sample with  $2.5 \mu\text{L}$  of  $22\text{mM}$  Ni) annealed in TAE 1x with  $12.5\text{mM}$  Mg at  $5\text{nM}$  p7249 with  $50\text{nM}$  staples at  $-1^\circ\text{C}/5\text{min}$  from  $65^\circ\text{C}$  to  $45^\circ\text{C}$ . Annealing and imaging of the origami were done by Nicolas Schabanel.

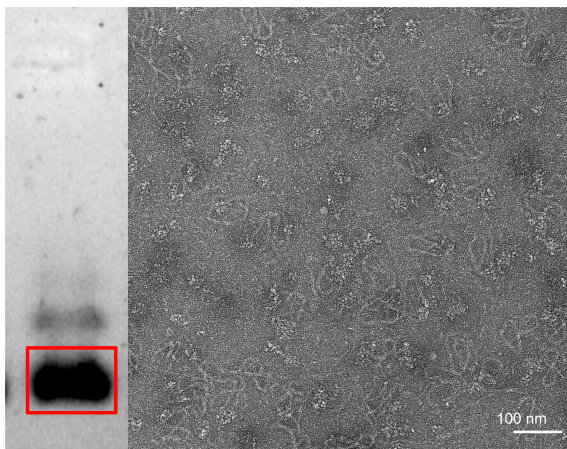




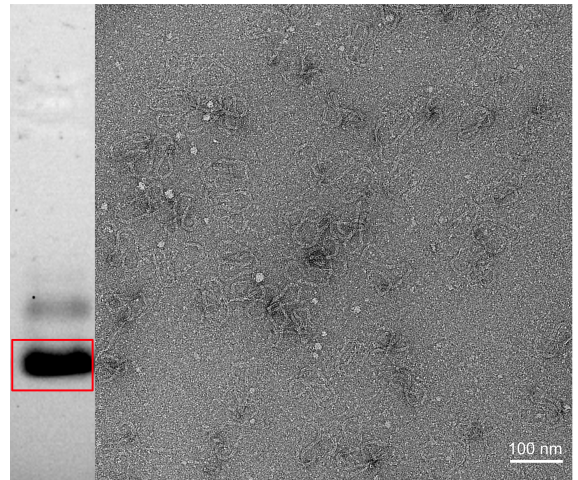
(a) Electrophoresis gel and TEM images of the looped square origami closed with linking staples (CJ variant). Top right corner: 2D class averaging.



(b) Electrophoresis gel and TEM images of the looped square origami open with junction staples (OJ variant).



(c) Electrophoresis gel and TEM images of the closed and junctionless variant of the looped square origami (CnoJ variant).



(d) Electrophoresis gel and TEM images of the open and junctionless variant of the looped square origami (OnoJ variant).

**Figure 9.15: TEM images of purified samples of the looped square origami.** The origami was annealed in a Folding buffer (5mM Tris, 1mM EDTA, pH=8) with 12mM Mg using 100nM of scaffold and 500nM of staples. The folding ramp was: 15min at 65° C followed by a descent from 60° C to 40° C at -1° C/hour.

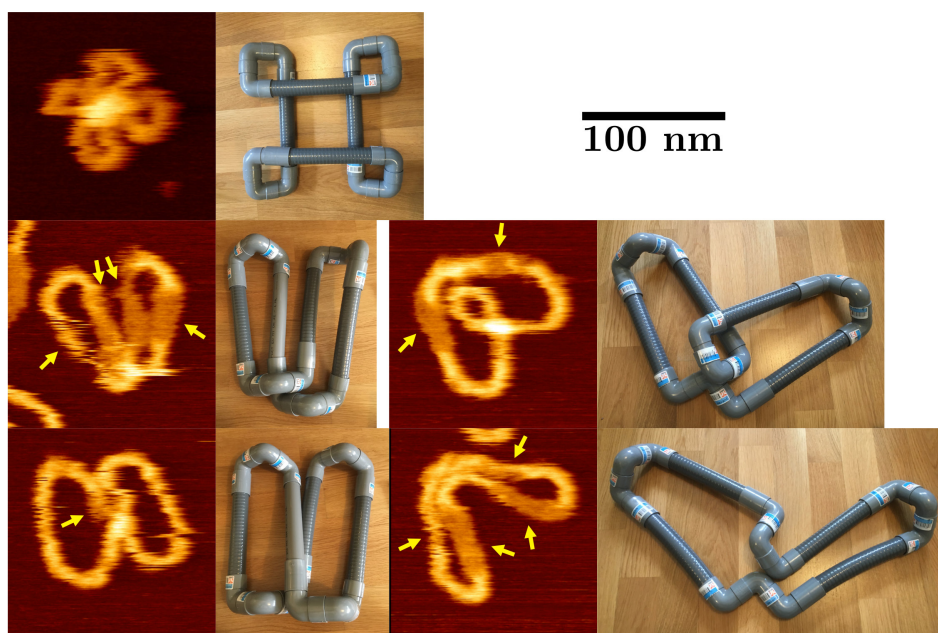


Figure 9.16: **Realization of the various shapes observed in AFM images using plumber pipes linked with 3/4-turns corners.** We could not obtain other closed shapes with the pipes. Note that the most twisted shapes could only be obtained in origami where staples in the most twisted segments seem to be missing as pointed out by the yellow arrows. (Note that the scale bar only apply to the AFM images.)



## Chapter 10

# Routing spiraling helices around shapes with 3D curvature

In this chapter, we develop a strategy for covering the surface of curved 3D shapes with DNA helices. Previous work [HPN<sup>+</sup>11, FPNP<sup>+</sup>22], used concentric rings of DNA. While this method allowed the successful assembly of 3D curved shapes, it suffers from several limitations.

- The shapes that can be obtained by stacking concentric rings of DNA, are restricted to revolution surfaces that feature an axial symmetry.
- The geometry of the DNA double helix limits the set of radii available for each layer. Indeed, the radius must correspond to an integer number of full turns of the helix. While this constraint can be relaxed for larger rings (see Section 8.2), it strongly limits the set of possible radii in the smaller range.
- The DNA helix advances by discrete steps of about 30°. Since the successive layers stacked in are parallel planes, the angle between them is restricted to the discrete set of crossover angles [HPN<sup>+</sup>11].
- Since the radii of the rings are restricted to a discrete range, it is difficult to produce a routing that uses exactly all the nucleotides of the scaffold.

The strategy that we develop in this chapter addresses these limitations by using long spiraling DNA helices whose axis are not contained in a plane. Roughly speaking, the idea is to “peel” the surface into an even number of closed ribbons (Figure 10.1).

The spiraling technique also *gives us access to a continuous range of crossover angles* and the resulting helices are typically much longer: the origami we made use only between 2 and 6 helices that are all at least 1000 bp long. Long helices give us access to a continuous range of revolution radii, because a slight tweak of their geometric parameters is needed to ensure that they close properly (see Section 8.3 in Chapter 8).

This extends the class of constructible shapes by allowing the design of origami with shapes that feature 3D curvatures such as the one shown on Figure 10.1.

Our spiraling technique is based on a parameterization of the surface by a *revolution angle*  $\theta \in [0, 2\pi]$  and a *section parameter*  $s \in [0, 1]$  (Figure 10.2).



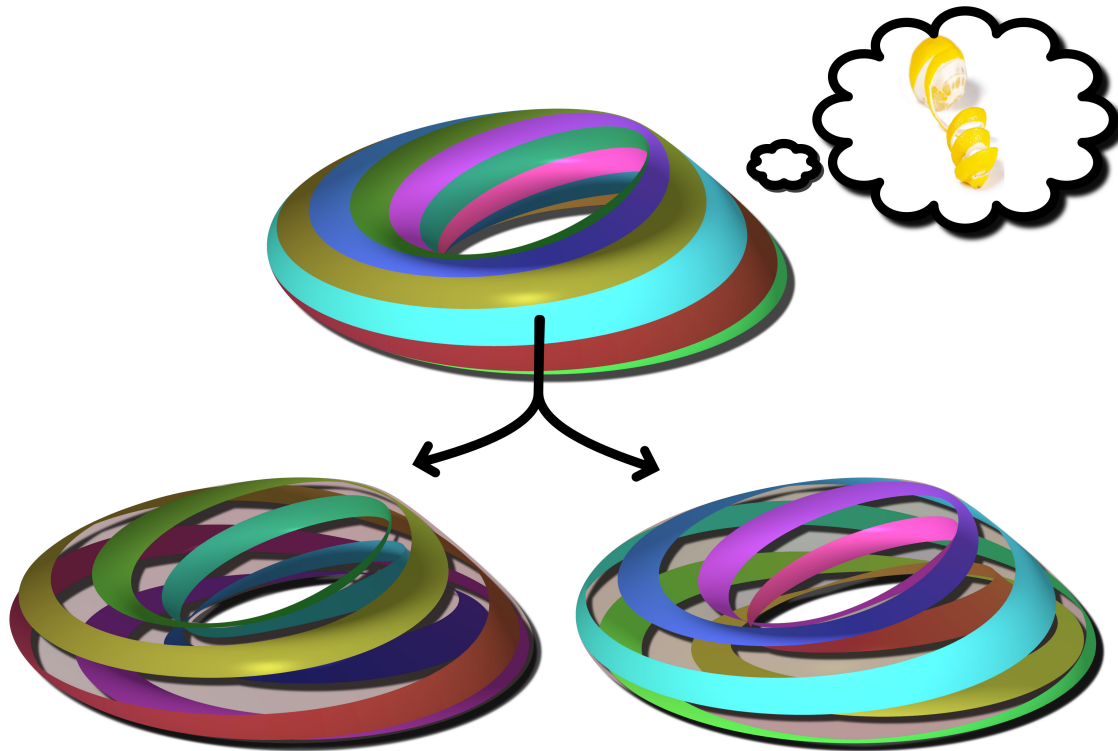


Figure 10.1: **Peeling the shape of a surface into closed spiral.** Notice that both sides of the bottom panel each show one unique continuous spiral. Together these two spirals cover the whole shape of a twisted torus.

## 10.1 Motivations and overview of the pipeline

**Motivations.** We designed our first origami with a 3D curvature before developing the methods presented in this chapter. The origami had a shape of a twisted torus with an elliptic section that was making half a turn in a revolution (see Figure 11.3). We called this design a *Möbius torus* (see Section 11.2 in Chapter 11 for more details on this design and its experimental assembly).

To design this origami, we manually derived the equation of the path of the helices (Figure 10.3), and optimized the geometric parameters of the shape so that the total length of the helices matches the length of the p7560 scaffold. This manual optimization was a particularly tedious process.

We then attempted to apply a similar method to design toruses with more twists. However, using the paths that we derived by hand for the spiraling helices led to a routing where the inter-helices distances was not uniform enough (Figure 10.4). While this was not detrimental to the design of the Möbius, the variation in inter-helices distances was significant in the torus with 5 half-twists (Figure 10.4). We therefore needed to find a way to balance the inter-helices distances.

We also critically needed an automated pipeline to design a double-sphere origami. Our first attempt was to design this shape with several modules, two spheres and two cylinders,

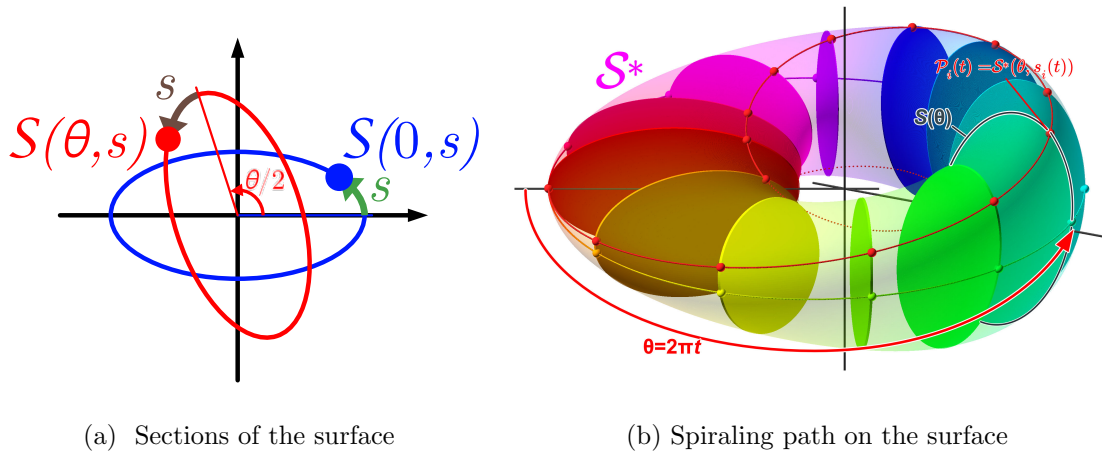


Figure 10.2: **Describing a path on a curved surface.** We consider a function  $\mathcal{S} : [0, 2\pi] \times [0, 1] \rightarrow [0, 1]$  so that for every *revolution angle*  $\theta$ ,  $\mathcal{S}(\theta, \cdot)$  is a closed curve (Panel a). Under some conditions (see Section 10.2), these sections describe a closed surface when revolving around an axis. In this example,  $\mathcal{S}(\theta, \cdot)$  is an ellipse rotated by an angle  $\frac{\theta}{2}$  so that the resulting surface is a twisted torus. The surface  $\mathcal{S}^*$  can then be parameterized by a function  $\mathcal{S}^* : [0, 2\pi] \times [0, 1] \rightarrow \mathbb{R}^3$  given by  $\mathcal{S}^*(\theta, s) = \mathbf{Q}_y(\theta)(Rz + \mathcal{S}(\theta, s))$  where  $\mathbf{Q}_y(\theta)$  is the rotation of angle  $\theta$  around axis  $y$  and  $R$  is a constant called the *revolution radius*. We consider spiraling paths along  $\mathcal{S}^*$  that are of the form  $\mathcal{P}_i(t) = \mathcal{S}^*(2\pi t, s_i(t))$  (Panel b). The path is therefore determined by the function  $s_i : [0, 1] \rightarrow [0, 1]$ . *The goal of this chapter is to construct the functions  $s_i$  so that the associated spiraling paths are equally spaced along the sections.*

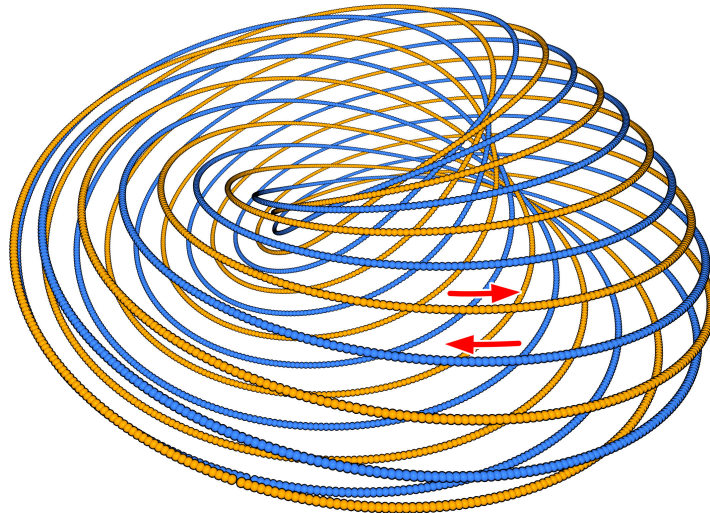


Figure 10.3: **Cover of the surface of a Möbius torus with two spirals.** This covering is obtained by having  $\mathcal{P}_i$  progress at constant curvilinear speed along the sections (see Section 10.2). We constructed this routing manually before implementing the automated methods presented in this chapter.

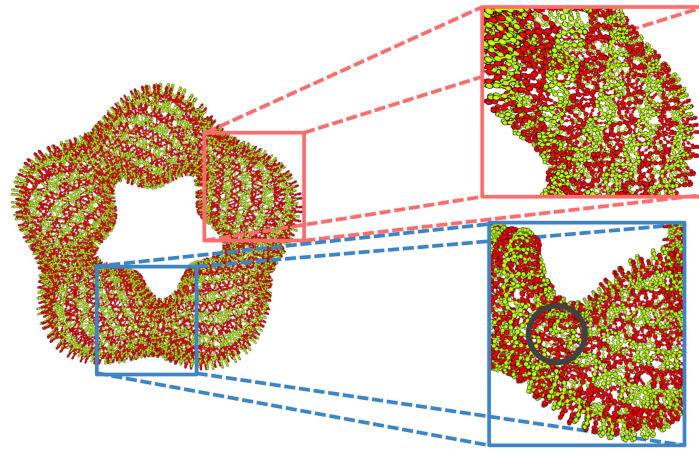
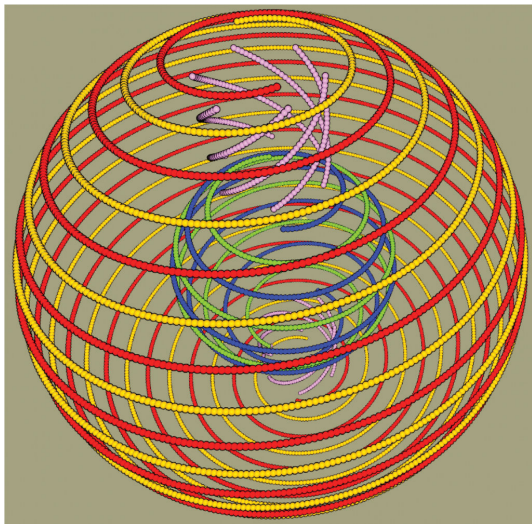
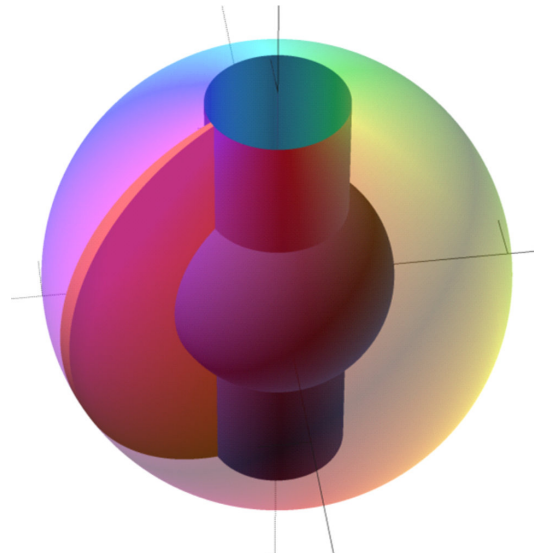


Figure 10.4: **Applying the method described in Figure 10.3 to a more complex surface** lead to a routing with significant imbalances in the inter-helices distances. Here in the more twisted region, the helices are almost intersecting each other while being further apart in the flatter region.



(a) Our attempt to make a double-sphere with several modules.



(b) The double-sphere shape seen as a revolution surface.

Figure 10.5: **Using revolution surface to describe complex shapes.** When dividing the double sphere shapes into modules, we did not manage to connect them smoothly. Seeing the double-sphere as a revolution surface allows using the pipeline that we describe in this chapter instead.

and to connect these modules by hand (Figure 10.5a). However, connecting the modules appeared to be a challenging task. Moreover, the optimization of the geometric parameters of this shape to match the scaffold length would certainly also have been difficult if not impossible.

Our automatic pipeline allowed us to solve this issue by expressing the shape of the double-sphere origami as a revolution surface with a ‘C’ shaped section (Figure 10.5b).

**Overview of the chapter.** We present a new technique that routes helices around a curved surface with uniform inter helix distance.

In Section 10.2, we define twisted revolution surfaces and see how to route helices around these shapes by having them follow a spiraling path that progresses at constant curvilinear speed on the consecutive sections.

The spiraling paths that we construct are characterized by functions  $s_i$  that describe the successive positions of the helices on the section of the surface (Figure 10.2). We then need to construct these functions  $s_i$  so that the inter-helices distances are balanced.

We will see in Section 10.3 that this is a challenging task. Indeed, the inter-helices distance depends on the angles at which the helices intersect the section plane passing through the revolution axis, and that these angles varies continuously. We will address this problem by introducing a spring system connecting the helices (Figure 10.6 B).

Finally, in Section 10.4, we explain see how we optimize the geometric parameters of the shape so that the total length of the routed helices matches the desired scaffold length (Figure 10.6 D).

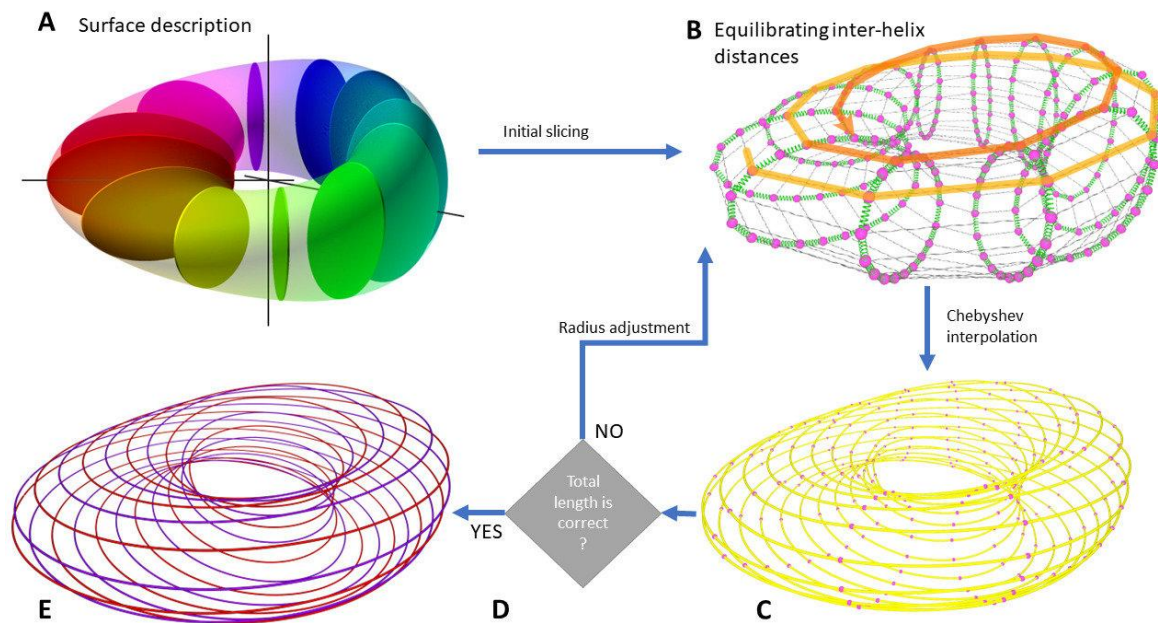


Figure 10.6: **Slicing of a twisted torus into spirals.** (A) Initial routing (Section 10.2): The surface is split into  $\xi$  sections orthogonal to the revolution plane. Spirals are initialized to go through points on those sections whose curvilinear abscissa grows linearly with the revolution angle.

(B) Inter-helix spacing (Section 10.3): The spirals are seen as successions of mass on the sections of the surface. A physical system containing springs connecting neighbouring points along each section is simulated. The equilibrium length of each spring takes into account the current angle at which the spiral intersects the corresponding section. During the relaxation of the system, the points are constrained to their respective section.

(C) The path of each spiral is constructed by a Chebyshev interpolation of the position of the points.

(D) The curves are discretized and their total length (in number of nucleotides) is computed. If the total length does not match the objective, the revolution radius is adjusted (Section 10.4).

(E) Final routing of the helices with target total length



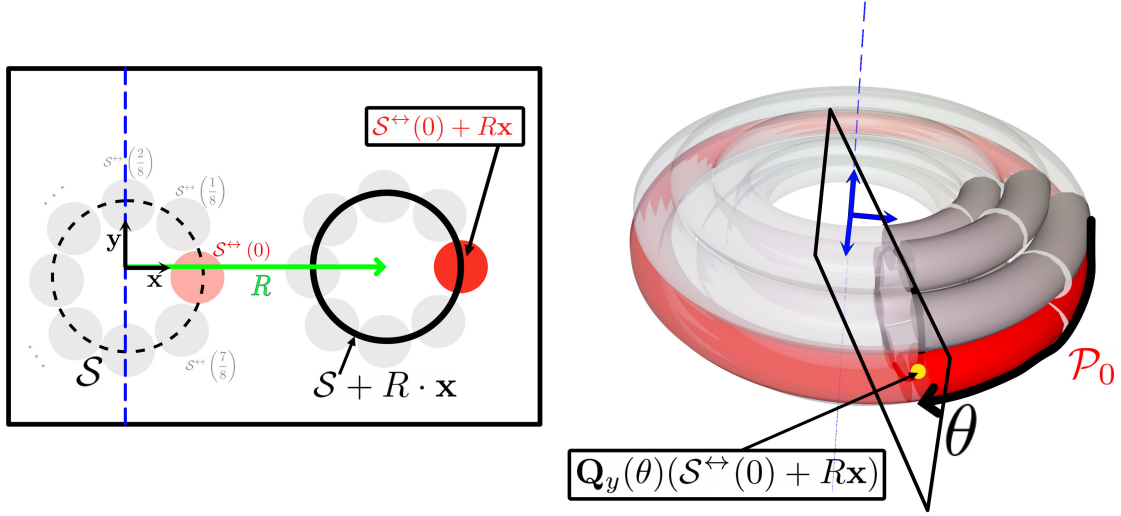


Figure 10.7: **Routing helices around untwisted revolution surface.** When  $\mathcal{S}$  is constant, it is possible to cover the revolution surface with helices that stay at a constant height. This corresponds to using *constant functions*  $s_i$  and leads to a helix routing made concentric circles as introduced in [HPN<sup>+</sup>11].

## 10.2 Initial Routing

Our method can be applied to shapes that are obtained by revolving a closed section  $\mathcal{S}$  around a central axis. The revolving section does not need to be constant, which allows the creation of shapes with 3D curvature. It however needs to have a constant perimeter because a constant number of helices is used throughout the revolution.

The section of the shape can therefore be seen as function

$$\mathcal{S} : [0, 2\pi] \times [0, 1] \rightarrow \mathbb{R}^2.$$

We require that  $\mathcal{S}$  be everywhere continuously differentiable and satisfies the following constraints:

$$\text{(Constant perimeter:)} \quad \exists P \in \mathbb{R}, \forall \theta \in [0, 2\pi], \quad \int_0^1 \mathcal{S}(\theta, s) ds = P. \quad (10.1)$$

$$\text{(Closed section:)} \quad \forall \theta \in [0, 2\pi], \quad \mathcal{S}(\theta, 0) = \mathcal{S}(\theta, 1). \quad (10.2)$$

$$\text{(Closed shape:)} \quad \forall s \in [0, 1], \quad \frac{\partial \mathcal{S}}{\partial s}(\theta, 0) = \frac{\partial \mathcal{S}}{\partial s}(\theta, 1) \quad (10.3)$$

$$\text{(Closed shape:)} \quad \forall s \in [0, 1], \quad \mathcal{S}(0, s) = \mathcal{S}(2\pi, s). \quad (10.4)$$

$$\frac{\partial \mathcal{S}}{\partial \theta}(0, s) = \frac{\partial \mathcal{S}}{\partial \theta}(2\pi, s) \quad (10.5)$$

We will focus on the special case where  $\mathcal{S}$  rotates on itself as it revolves around the central axis, *i.e.* when  $\mathcal{S}$  is of the form

$$\mathcal{S}(\theta, s) = \begin{pmatrix} \cos(\omega\theta) & -\sin(\omega\theta) \\ \sin(\omega\theta) & \cos(\omega\theta) \end{pmatrix} \mathcal{S}(0, s), \quad (10.6)$$

where  $\omega \in \mathbb{R}$  is a parameter that must be chosen so that Equation (10.2) holds. This is for example the case when  $\omega$  is an integer, but if  $\mathcal{S}$  is invariant by rotation by an angle  $\frac{2\pi}{m}$ , then  $\omega$  can also be any integer multiple of  $\frac{1}{m}$ .

When  $\mathcal{S}$  is of the form (10.6), we call the surface obtained by revolving  $\mathcal{S}$ , a *twisted revolution surface*.

Such a surface can be described by a function  $\mathcal{S}^* : [0, 2\pi] \times [0, 1] \rightarrow \mathbb{R}^3$ . This function is given by

$$\mathcal{S}^*(\theta, s) = \mathbf{Q}_y(\theta)(R\mathbf{z} + \mathcal{S}(\theta, s)),$$

where  $R$  is a constant called the *revolution radius* and  $\mathbf{Q}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$  is the matrix representing the rotation of angle  $\theta$  around the  $\mathbf{y}$  (Figure 10.2).

**Positioning equispaced points along the perimeter of the section.** For now, we make the hypothesis that the distance between the axis of neighboring helices can be measured in a plane passing through the revolution axis. When routing helices around the shape, we want this distance to be uniform and equal to a constant  $\mathcal{H}$ . Typically, we take  $\mathcal{H} = 2.65\text{nm}$  as advised in [TLJ<sup>+</sup>17, WR11]. This implies that:

- the perimeter of the section must be equal to  $2n\mathcal{H}$
- the helices must to trough points that are equispaced along the section.

In order to cover the perimeter of the section with helices, we will have the helices go through points that are equally spaced along the section. For this reason, we define a section  $\mathcal{S}^{\leftrightarrow}$  obtained by performing a change of variable and a scaling on  $\mathcal{S}$ :

$$\begin{aligned} \mathcal{S}^{\leftrightarrow} : [0, 2\pi] \times [0, 1] &\rightarrow \mathbb{R}^2 \\ (\theta, t) &\mapsto \frac{2\mathcal{H}n}{P}\mathcal{S}(\theta, \sigma_\theta(tP)), \end{aligned} \quad (10.7)$$

where  $P$  is the perimeter of the section defined in Equation (10.1) and  $\sigma_\theta$  is the inverse curvilinear abscissa of the function  $\mathcal{S}(\theta, \cdot) : s \mapsto \mathcal{S}(\theta, s)$ .

This change of variable ensures that for  $0 \leq a \leq b \leq 1$ ,

$$\int_a^b \left\| \dot{\mathcal{S}}^{\leftrightarrow}(\theta, t) \right\|_2 dt = (b - a) \cdot 2\mathcal{H}n,$$

which means that if  $\mathcal{S}$  follows a “reasonable” trajectory, for  $n \in \mathbb{N}$ , the points  $\mathcal{S}^{\leftrightarrow}(\theta, 0), \mathcal{S}^{\leftrightarrow}(\theta, \frac{1}{n}), \dots, \mathcal{S}^{\leftrightarrow}(\theta, \frac{n-1}{n})$  are approximately equispaced along the section.

**Routing spiraling helices around the surface** One way to see methods that use concentric rings of DNA such as [HPN<sup>+</sup>11, FPNP<sup>+</sup>22] is to say that they are applied to shape with a constant section, *i.e.* where  $\mathcal{S}(\theta, s)$  does not depend on  $\theta$ . In that method, when  $n$  helices are used to cover the surface of the shape obtained by revolving  $\mathcal{S}$  around the  $y$  axis with a revolution radius of  $R$  (see Figure 10.7), the path followed by the helix of index  $i$  is given by

$$\mathcal{P}_i(t) = \mathcal{S}^*(2\pi t, s_i(t)), \quad (10.8)$$

where  $s_i$  is a constant function  $s_i(t) = \sigma(\frac{i}{n}P)$  (Figure 10.7).

In order to route *spiraling* helices around the shape, we slightly modify Equation (10.8) by having the curvilinear abscissa of each point grow linearly with  $\theta$ , and taking into account the fact that  $\mathcal{S}(\theta, s)$  might depend on  $\theta$  (Figure 10.2). The function  $s_i$  in Equation (10.8) then becomes

$$s_i(t) = \sigma \left( \left( \frac{i}{n} + kt \right) P \right), \quad (10.9)$$

where  $k \in \mathbb{Z}$  is the *spirality parameter* of the routing.

The parameter  $k$  influences the speed at which the spirals follow the perimeter of  $\mathcal{S}$  (Figure 10.2). The spirality parameter  $k$  also influences the angle at which the spiral intersect the sections of the surface. Very positive or very negative values of  $k$  will lead to hitting angles that are close to  $\pi/2$ . However, the smallest angles are not necessarily obtained when  $k$  is null. In the case of twisted revolution surfaces, this spirality parameter can also be used to “go against” the rotation of  $\mathcal{S}$ , as illustrated in the last row of Figure 10.8.

Notice that, in general, it is not the case that  $\mathcal{P}_i(0) = \mathcal{P}_i(1)$ . The consequence is that one revolution does not correspond to a full closed path. Instead,  $\mathcal{P}_i$  is connected to  $\mathcal{P}_{i+k(\bmod n)}$  which itself is connected to  $\mathcal{P}_{i+2k(\bmod n)}$  etc... We call each  $\mathcal{P}_i$  a *segment* of spiral. The concatenation of consecutive segments of spiral eventually leads to a full closed spiral.

**Influence of  $k$  on the number of segments in a spiral** The number of segment in each spiral, and therefore the number of spirals depends on  $k$ .

**Property 3.** When the surface is divided in  $n$  segments with spirality parameter  $k$ , the number of segment in each spiral is equal to  $\gcd(k, n)$

**Proof.** Let  $d = \gcd(k, n)$ . It is sufficient to show that two segments  $\mathcal{P}_i$  and  $\mathcal{P}_j$  are on the same spiral if and only if  $i = j \pmod{d}$ .

$\mathcal{P}_i$  and  $\mathcal{P}_j$  are on the same spiral if there exists  $q \in \mathbb{Z}$  so that  $i + kq = j \pmod{n}$ . In other word,

$$\exists p, q \in \mathbb{Z}, i = j + kq + pn. \quad (10.10)$$

Since  $d = \gcd(k, n)$ , by Bézout Theorem, there exists  $a, b \in \mathbb{Z}$  so that  $d = ak + bn$ .

Therefore, if  $i = j \pmod{d}$ , there exists  $c$  so that  $i = cd + j = cak + cbn + j$  and (10.10) holds.

Reciprocally, since for  $k, p \in \mathbb{Z}$ ,  $d$  divide  $kq$  and  $pn$ , (10.10) implies that  $i = j \pmod{d}$ .

### 10.3 Balancing the inter-helix distances

In our initial routing,  $\mathcal{P}_i(t)$  progresses at constant curvilinear speed on the perimeter of  $\mathcal{S}$ . This is important. If  $\mathcal{P}_i(t)$  were to progress by increasing linearly the section parameter instead of the curvilinear abscissa, the resulting routing could have inter-helices distance be heterogeneous (Figure 10.9).

When we defined the scaling factor in Equation 10.7, we implicitly made the hypothesis that the distance between neighboring helices can be measured in a plane that is orthogonal to the revolution plane. This does however not take into account the fact that the helices follow spiraling paths and are, in general, not parallel to the revolution plane. In that case, the angle at which the helices intersect the section plane must be taken into account in the estimation of the inter-helix distance as shown in Figure 10.10.



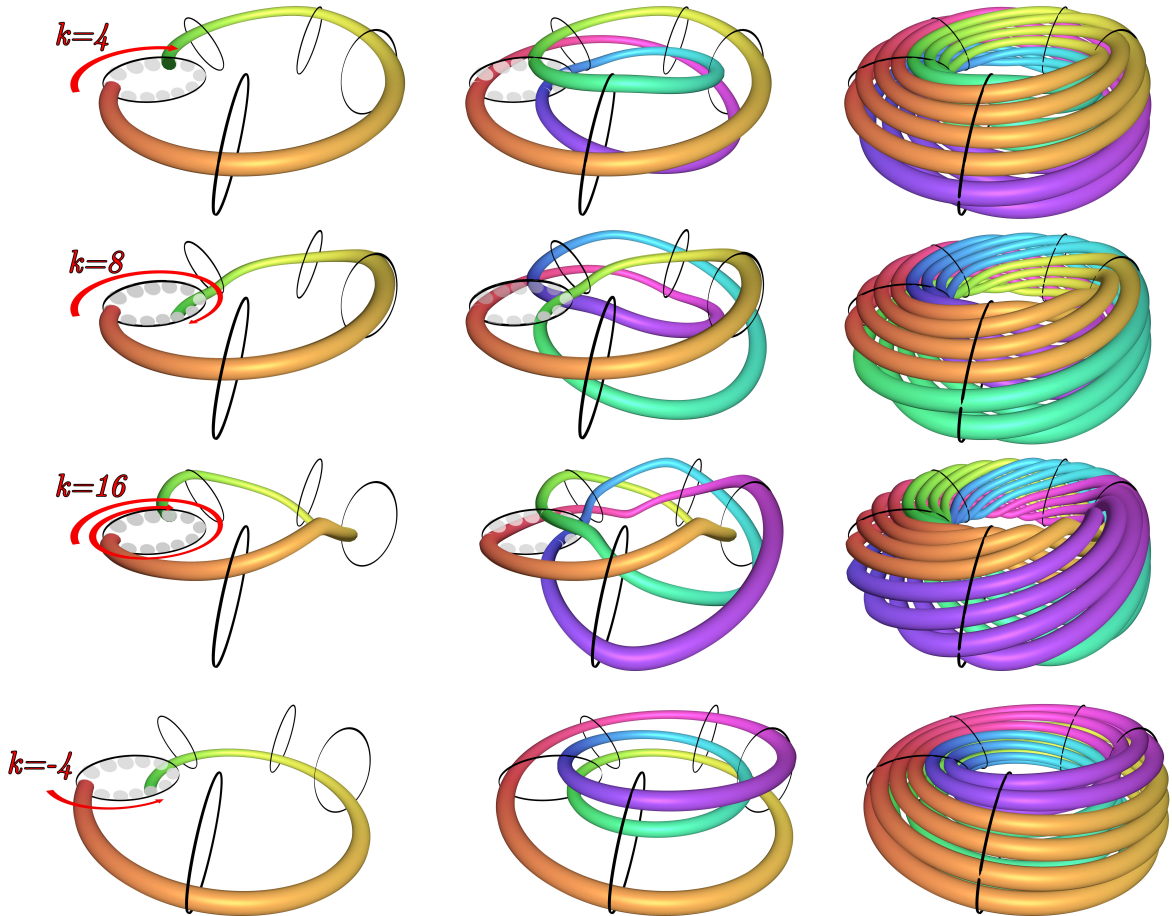
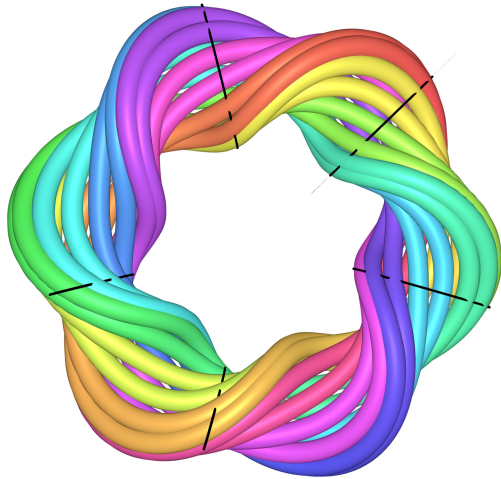


Figure 10.8: **Influence on the spirality parameter  $k$  on the routing of the helices around a revolution shape.**  $n = 12$  points have been placed at equispaced positions on the perimeter of the section. For all the values of  $k$  illustrated here,  $\gcd(k, 12) = 4$ . Each spiral is therefore made of 4 segments and there are  $12/4 = 3$  spirals. Left column: Routing of one spiral segment. Middle column: Routing of one complete spiral, making 3 turns around the revolution axis. Right column, routing of all the spirals around the shape. Notice that setting  $k$  to  $-4$  helps “going against” the rotation of  $\mathcal{S}$  and lead to spirals that are almost perpendicular to the sections.

$$\mathcal{P}_i(t) = \mathcal{S} \left( 2\pi t, \frac{i + kt}{n} \right)$$



$$\mathcal{P}_i(t) = \mathcal{S} \left( 2\pi t, \sigma \left( \frac{i + kt}{n} P \right) \right)$$

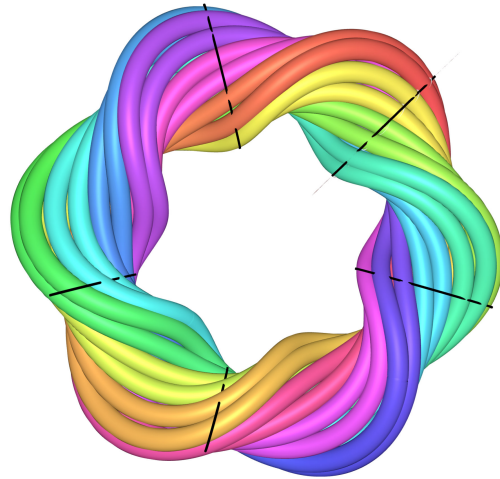


Figure 10.9: **Importance of having the spiral follow a linearly increasing curvilinear abscissa as opposed to section parameter during their revolution.** Routing made by having the helices follow a linearly increasing section parameter (Left), have a more heterogeneous inter-helix distance than routing made by having the curvilinear abscissa increase linearly (Right).

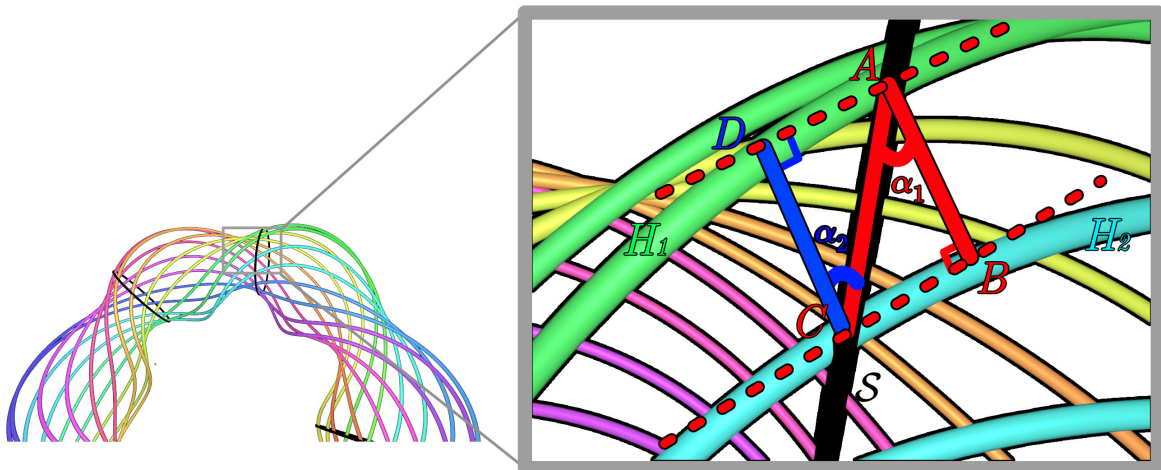


Figure 10.10: **Estimating inter helix distance when the helices intersect the section in a non-orthogonal way.** Helices  $H_1$  and  $H_2$  intersect the section  $S$  at respectively the points  $A$  and  $B$ . The orthogonal projection of  $A$  on  $H_2$ 's axis is  $C$ . When the  $H_1$  and  $H_2$  are not orthogonal to  $S$  the distance  $d$  between  $H_1$  and  $H_2$  is not equal to  $\|\vec{AC}\|_2$ , and is better estimated by the formula  $d \simeq \frac{1}{2} (\|AB\|_2 + \|AD\|_2) = \frac{1}{2} (\|AC\|_2 \cos(\alpha_1) + \|AC\|_2 \cos(\alpha_2))$ .

When a helix  $H_1$  and  $H_2$  intersects  $\mathcal{S}$  with an angle  $\alpha_1$  and  $\alpha_2$ , the distance between  $H_1$  and  $H_2$  can be estimated by the formula  $d(H_1, H_2) \simeq \frac{\cos(\alpha_1) + \cos(\alpha_2)}{2} d_{\mathcal{S}}(H_1, H_2)$ , where  $d_{\mathcal{S}}(H_1, H_2)$  is the distance between  $H_1$  and  $H_2$  when measured along  $\mathcal{S}$ . By construction, in our initial routing we have  $d_{\mathcal{S}}(H_1, H_2) = \mathcal{H}$ . If the angles  $\alpha$  were constant through the routing, we could rescale  $\mathcal{S}$  by a factor  $\frac{1}{\cos(\alpha)}$  and be done. However, in some shapes,  $\alpha$  varies too much to be reasonably approximated by a constant.

To overcome this problem, we run a physical simulation of a spring system that will modify the trajectory of the spiral segments in order to improve the inter-helix distances. We also keep track of a scaling constant  $K$ , initialized at  $K = \frac{2n\mathcal{H}}{P}$ , that we will update after each relaxation of the system.

The spring system will act on specific points that correspond to the intersections of the spiral segments with sections of the shape. The points will move on their corresponding sections, and when equilibrium is reached, the scaling constant  $K$  will be updated according to the average extension of the springs, before simulating the system again.

Note that, as the system evolves, the angle at which the helices intersect the sections of the surfaces will also change, meaning that the equilibrium length of the springs is constantly changing.

**Description of the system** We consider a constant number of sections  $\xi$  so that the  $i$ -th section is  $\mathcal{S}_i = \mathcal{S}(\theta_i)$ , where  $\theta_i = \frac{2i\pi}{\xi}$ . For  $0 \leq i < \xi$  and  $0 \leq j < n$ , we call  $A_{i,j}$  the intersection between the section  $\mathcal{S}_i$  and the segment of spiral  $\mathcal{P}_j$ . We extend this notation by writing

$$\begin{aligned} A_{-1,j} &= A_{\xi-1, j-k(\bmod n)} \text{ (the last section of } \mathcal{P}_j \text{ is connected to the first section of } \mathcal{P}_{j+k}), \\ A_{n,j} &= A_{0, j+k(\bmod n)} \text{ (the first section of } \mathcal{P}_j \text{ is connected to the last section of } \mathcal{P}_{j-k}), \\ A_{i,\xi} &= A_{i,0}, \text{ and} \\ A_{i,-1} &= A_{i,\xi-1}. \end{aligned}$$

The physical system that we simulate acts on the positions of the points  $A_{i,j}$  and consists of:

- Linear springs between  $A_{i,j}$  and  $A_{i,j+1}$  whose equilibrium length are given by

$$\ell_{i,j} = \frac{1}{2} \left( \frac{\mathcal{H}}{\cos(\alpha_{i,j})} + \frac{\mathcal{H}}{\cos(\alpha_{i,j+1})} \right),$$

where  $\alpha_{i,j} = \arccos(\langle \overrightarrow{A_{i-1,j}A_{i+1,j}} | \mathbf{Q}_y(\theta_i)\mathbf{z} \rangle)$  is an approximation of the angle at which  $\mathcal{P}_j$  intersects  $\mathcal{S}_j$  (recall that  $\mathbf{Q}_y(\theta)$  is the matrix associated to the rotation of angle  $\theta$  around axis  $y$ ).

These linear springs apply a force

$$F_{i,j}^{\text{lin}} = k^{\text{lin}} \left( 1 - \frac{\ell_{i,j}}{\| \overrightarrow{A_{i,j}A_{i,j+1}} \|_2} \right) \overrightarrow{A_{i,j}A_{i,j+1}} \quad (10.11)$$

on  $A_{i,j}$ , where  $k^{\text{lin}}$  is a constant, and the opposite of that force to  $A_{i,j+1}$ .

- Torsion springs that try to minimize the angles  $\widehat{A_{i-1,j}A_{i,j}A_{i+1,j}}$ . These springs are added to the system because we think that reducing the torsion within the DNA helices should lead to less constrained structures.

These torsion springs apply a force

$$F_{i,j}^{\text{tor}} = k^{\text{tor}} \frac{\overrightarrow{A_{i,j}A_{i+1,j}} - \overrightarrow{A_{i-1,j}A_{i,j}}}{\left\| \overrightarrow{A_{i,j}A_{i+1,j}} - \overrightarrow{A_{i-1,j}A_{i,j}} \right\|_2} \quad (10.12)$$

on  $A_{i,j}$  where  $k^{\text{tor}}$  is a constant, and half the opposite of that force to both  $A_{i-1,j}$  and  $A_{i+1,j}$ .

- A normal reaction  $R$  from the sections that keeps  $A_{i,j}$  on  $\mathcal{S}$ .

This means that the total force applied on  $A_{i,j}$  is given by

$$F_{i,j} = F_{i,j}^{\text{lin}} - F_{i,j-1}^{\text{lin}} + F_{i,j}^{\text{tor}} - \frac{1}{2}F_{i-1,j}^{\text{tor}} - \frac{1}{2}F_{i+1,j}^{\text{tor}} + R. \quad (10.13)$$

We write  $\hat{F}_{i,j}$  the force applied on  $A_{i,j}$  minus the normal reaction:

$$\hat{F}_{i,j} = F_{i,j} - R.$$

**Dynamics of the system.** Before defining the dynamic of the system, we start by rescaling the section by the average elongation of the linear springs. That is to say that we compute

$$\Gamma = \frac{1}{n\xi} \sum_{0 \leq i < \xi, 0 \leq j < n} \frac{\left\| \overrightarrow{A_{i,j}A_{i,j+1}} \right\|_2}{\ell_{i,j}}$$

and rescale  $\mathcal{S}$  by updating  $K$ :

$$K \leftarrow \frac{1}{\Gamma} K \quad (10.14)$$

We then relax the system by computing an approximate solution of the differential equation

$$\begin{cases} \Sigma(0) = \Sigma_0 \\ \dot{\Sigma}(t) = f(\Sigma(t)) \end{cases}, \quad (10.15)$$

where:

- The state of the system is given by

$$\Sigma(t) = \begin{pmatrix} s_{0,0}(t) \\ s_{1,0}(t) \\ \vdots \\ s_{\xi-1,0}(t) \\ s_{0,1}(t) \\ \vdots \\ s_{\xi-1,n-1}(t) \\ v_{0,0}(t) \\ v_{1,0}(t) \\ \vdots \\ v_{\xi-1,0}(t) \\ v_{0,1}(t) \\ \vdots \\ v_{\xi-1,n-1}(t) \end{pmatrix},$$

where  $s_{i,j}$  is the section parameter associated to  $A_{i,j}$  *i.e.* the value so that  $A_{i,j} = K\mathcal{S}^*(\theta_i, s_{i,j})$ , and  $v_{i,j}$  is the derivative of  $s_{i,j}$  with respect to time.

In our system, the point  $A_{i,j}$  is only allowed to move on its section, meaning that  $A_{i,j}$  can be seen as a function of only  $s_{i,j}$  since  $\theta_i$  is fixed. We write  $A_{i,j} = M(s_{i,j})$ , with

$$M(s_{i,j}) = K\mathcal{S}^*(\theta_i, s_{i,j})$$

- $\Sigma(0)$  is given by  $s_{i,j}(0) = \sigma_{\theta_j} \left( \frac{j}{n} + k \frac{i}{\xi} \right)$  (the value that we used in the initial routing) and  $v_{i,j}(0) = 0$ .
- The dynamics of  $\Sigma$  is defined by

$$f(\Sigma(t)) = \begin{pmatrix} v_{0,0}(t) \\ v_{1,0}(t) \\ \vdots \\ v_{\xi-1,0}(t) \\ v_{0,1}(t) \\ \vdots \\ v_{\xi-1,n-1}(t) \\ a_{0,0}(t) \\ a_{1,0}(t) \\ \vdots \\ a_{\xi-1,0}(t) \\ a_{0,1}(t) \\ \vdots \\ a_{\xi-1,n-1}(t) \end{pmatrix}, \quad (10.16)$$

where  $a_{i,j}$  is the second derivative of  $s_{i,j}$  with respect to time. To express  $a_{i,j}$ , notice that

$$\begin{aligned}\frac{d^2M}{dt^2} &= \frac{d}{dt} \left( \frac{dM}{dt} \right) \\ &= \frac{d}{dt} \left( \frac{dM}{ds} \frac{ds}{dt} \right) \\ &= \frac{d^2M}{ds^2} \left( \frac{ds}{dt} \right)^2 + \frac{dM}{ds} \frac{d^2s}{dt^2} \\ &= \frac{d^2M}{ds^2} v_{i,j}^2 + \frac{dM}{ds} a_{i,j}.\end{aligned}$$

Now, assume that all points have the same mass  $m$ . If  $F_{i,j}$  is the total force applied to  $A_{i,j}$ , Newton's first law gives

$$\begin{aligned}\frac{F_{i,j}}{m} &= \frac{d^2M}{dt^2} \\ \frac{F_{i,j}}{m} &= \frac{d^2M}{ds^2} v_{i,j}^2 + \frac{dM}{ds} a_{i,j} \\ a_{i,j} \frac{dA_{i,j}}{ds_{i,j}} &= \frac{F_{i,j}}{m} - \frac{d^2M}{ds^2} v_{i,j}^2 \\ \left\langle a_{i,j} \frac{dM}{ds} \middle| \frac{dM}{ds} \right\rangle &= \left\langle \frac{\hat{F}_{i,j}}{m} - \frac{d^2M}{ds^2} v_{i,j}^2 \middle| \frac{dM}{ds} \right\rangle \quad (\star) \\ a_{i,j} &= \frac{1}{\left\| \frac{dM}{ds} \right\|_2^2} \left( \frac{\left\langle \hat{F}_{i,j} \middle| \frac{dM}{ds} \right\rangle}{m} - \left\langle \frac{d^2M}{ds^2} v_{i,j}^2 \middle| \frac{dM}{ds} \right\rangle \right).\end{aligned} \quad (10.17)$$

Note that  $(\star)$  holds because  $R$  is normal to the section so  $\langle R \middle| \frac{dM}{ds} \rangle = 0$ .

**Simulation of the system** In order to simulate the spring system, *i.e.* solve the ODE (10.15), we use an ODE solver from the third-party `mathru`[Eih] library. This solver simply takes as input the initial state  $\Sigma_0$ , and the function  $f$  defined in Equation (10.16). To implement  $f$ , we can compute  $a_{i,j}$  using Equation (10.17). This requires that we compute the first and second derivative of  $M$  with respect to  $s$ . These derivatives are approximated numerically using the relations

$$\begin{aligned}\frac{dM}{ds} &\simeq K \mathbf{Q}_y(\theta_i) \frac{\mathcal{S}(\theta_i, s + \varepsilon) - \mathcal{S}(\theta_i, s)}{\varepsilon} \\ \frac{d^2M}{ds^2} &\simeq K \mathbf{Q}_y(\theta_i) \frac{\mathcal{S}(\theta_i, s - \varepsilon) + \mathcal{S}(\theta_i, s + \varepsilon) - 2\mathcal{S}(\theta_i, s)}{\varepsilon^2}\end{aligned}$$

for a value of  $\varepsilon$  arbitrarily set at  $10^{-6}$ . We also need to compute  $\hat{F}_{i,j}$  which we derive from Equations (10.13), (10.11) and (10.12).

Using the solver, we solve the ODE (10.15) on the intervals  $[0, T]$ ,  $[T, 2T]$ ,  $\dots, [(N-1)T, NT]$  where  $N \in \mathbb{N}$  and  $T > 0$  are parameters of the simulation. We then compute the average extensions of the linear springs in  $\Sigma(NT)$ , and rescale  $\mathcal{S}$  again by updating  $K$  as in (10.14).

**Representation of equilibrated paths** The inter-helix distances equilibrating process is considered to be done when the length of all the linear springs is close enough to the equilibrium length. The path of each segment is therefore given by

$$\mathcal{P}_j(t) = K\mathbf{Q}_y(2\pi t)[\mathcal{S}(2\pi t, \tilde{s}_j(t)) + R\mathbf{z}],$$

where  $\tilde{s}_j$  is, for efficiency reasons, a Chebyshev interpolation of the set  $\left\{\left(\frac{i}{\xi}, s_{i,j}\right)\right\}_i$ .

Final care is required in order to concatenate these paths into closed spirals. In the initial routing, we had constructed the paths so that  $\mathcal{P}_j(1) = \mathcal{P}_{(j+k) \pmod n}$ . There is however no reason for this equality to still hold now that the definition of  $\mathcal{P}_j$  depends on  $\tilde{s}_j$ .

To fix this we chose a smoothening parameter  $a$ , and in the interval  $[m - a, m + a]$  the coordinate of the  $j$ -th spiral  $\Pi_j$  is obtained by a Bézier-like linear interpolations of  $\mathcal{P}_{j+(m-1)k}$  and  $\mathcal{P}_{j+mk}$  (Figure 10.11):

$$\Pi_j(t) = \begin{cases} \frac{1+\{t\}}{2a}\mathcal{P}_{j+k\lfloor t\rfloor}(\{t\}) + \left(1 - \frac{1+\{t\}}{2a}\right)\mathcal{P}_{j+k(\lfloor t\rfloor-1)}(1 - \{t\}) & \text{if } \{t\} < a \\ \frac{\{t\}-1-a}{2a}\mathcal{P}_{j+k\lfloor t\rfloor}(\{t\}) + \left(1 - \frac{\{t\}-1-a}{2a}\right)\mathcal{P}_{j+k(\lfloor t\rfloor+1)}(1 - \{t\}) & \text{if } \{t\} > 1 - a \\ \mathcal{P}_{j+k\lfloor t\rfloor}(\{t\}) & \text{otherwise,} \end{cases}$$

## 10.4 Adjustments of the revolution radius to the desired scaffold length

When making a DNA origami, one is limited to a discrete set of scaffold lengths. Commercially available scaffold strands have a length of either 7249, 7560 or 8064 nucleotides, and while various methods exist to produce scaffold strand of other lengths, these methods have much lower yields [BSV<sup>+</sup>20].

This means that it is critical for the designer to control the number of nucleotides that is required to assemble their design. In the case of twisted revolution surfaces, the total length of the helices can be adjusted by increasing or decreasing the revolution radius.

Manually adjusting the revolution radius so that the total number of base pairs in the routed helices matches the desired scaffold length would be a tedious process requiring a lot of trials and error. This is why we chose to automatize this process in ENSnano.

Once the initial routing is computed (Section 10.2) and the inter-helix distances have been equilibrated (Section 10.3), we compute the sum of the lengths of the spirals and deduce the total number of nucleotides  $N$  in the routed helices.

If this number is different from the desired scaffold length  $N^*$ , the revolution radius is multiplied by  $\frac{N^*}{N}$ <sup>1</sup>. We then equilibrate the inter-helix distances again and repeat until the total length in nucleotides matches the scaffold length. If, after an iteration,  $N$  is close enough to  $N^*$ , the user can also choose to stop the radius optimization procedure and keep the current value of  $R$ . In that case, ENSnano adjusts geometric parameters of the helices (Section 8.2) so that the routed helices have the correct number of nucleotides. This means that the nucleotides positions around the helices will be generated using a modified rise value  $\Delta'$  so that  $\Delta' \simeq \frac{N}{N^*}\Delta$ . In practice this changes the helicity and rise by a few tenths of percent.

<sup>1</sup>This is probably not the optimal choice since the total length of the helices is an affine function and not a linear one.

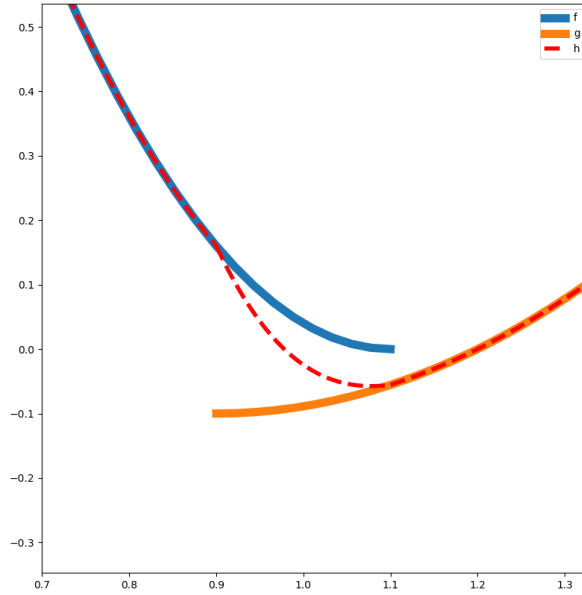


Figure 10.11: **Concatenating two curves by linear interpolation.** The graphs of  $f$  and  $g$  cannot be directly concatenated into a continuous function. To overcome this, we set a smoothing parameter  $a$  (in this case,  $a = 0.1$ ) and the concatenated function  $h$  is defined as being equal to  $f$  on the left of  $1 - a$  and to  $g$  on the right of  $1 + a$ . On the interval  $[1 - a, 1 + a]$ ,  $h(x)$  is given by  $h(x) = uf(x) + (1 - u)g(x)$ , where  $u = \frac{x - (1 - a)}{2a}$ .

When designing the origami that will be presented in the next structure, we always let the radius optimization procedure run to completion which takes less than a few minutes. However, we think that the radius optimization process could reasonably be stopped when  $N \in [0.99N^*, 1.01N^*]$ .






# Chapter 11

## Experimental validations: design and assembly of origami with complex 3D curvatures

### 11.1 Design of twisted revolution surfaces in ENSnano

#### 11.1.1 Routing the helices

**Describing the surface.** In ENSnano, twisted revolution surfaces are created in the Revolution Tab  (see Chapter 6). The Revolution Tab allows the user to choose the section  $\mathcal{S}$ , the available options are:

- An ellipse. In that case the user can input the major and minor axis of the ellipse.
- A 'C' shaped section, made of two concentric half circles connected by a straight line. This shape generates a double sphere when revolving on itself (see Figure 10.5b). The user can choose the inner and outer radii, as well as the length between the two half circles.
- A closed Bézier curve drawn by the user (see Section 9.2).

In addition to the section, the user can choose the number of half turns that the section will make during a revolution, and a revolution radius. Note that the revolution radius  $R$  is defined as follows: If  $R \leq 0$  (*resp.*  $R \geq 0$ ), then  $|R|$  is the distance from the leftmost (*resp.* rightmost) point of the section to the revolution axis.

The revolution radius can be entered in the Revolution Tab, or the revolution axis can be grabbed and moved in the 3D view (Figure 11.1). If the revolution axis is moved *inside* the section, the corresponding revolution surface is considered to be invalid.

**Setting up the routing parameters.** The section, number of half turns per revolution together with the revolution radius, define the *shape* of the twisted revolution surface, which is drawn live in the 3D view. Since a DNA origami uses a fixed given number of nucleotides, the shape will need to be rescaled. The length of the scaffold is also entered by the user. The number of spiral segments (see Chapter 10) that are used for routing the helices is calculated

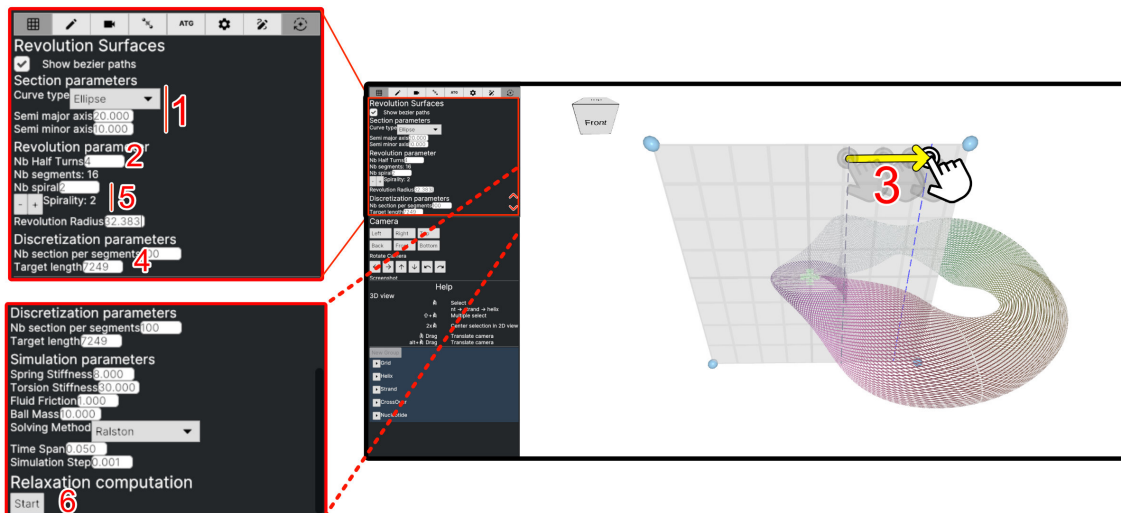


Figure 11.1: **Creating twisted revolution surfaces with ENSnano’s interface.** 1: Select the shape of the section and adjust its geometric parameters. 2: Set the number of half turns per revolution. 3: Grab and move the revolution axis. 4: Set the target scaffold length. 5: Choose the number of spirals and set the spirality parameter  $k$ . If the desired number of spirals does not divide the number of spiral segments, this can be fixed by adjusting the revolution radius. 6: Start the routing process.

so that in the initial routing, the total number of nucleotides is close enough to the target scaffold length.

Once the number of spiral segments is fixed, the user can choose the total number of spirals, and if this number divides the number of spiral segments, the user can choose the spirality parameter  $k$  among those who gives the desired number of spirals.

Finally, press the start button to initiate the whole helices routing process described in Chapter 10. Figure 11.1 summarizes how to set up the routing of helices around twisted revolution surfaces in ENSnano.

**Aborting/Interrupting the radius optimization process.** In the best case scenario, after several steps of optimization of the section scaling parameter and revolution radius (see Chapter 10), a routing that matches exactly the desired scaffold length is found. In that case the routing process stops and DNA helices are created that follow the computed routes.

As it is a possibly non-convex optimization system, we cannot provide convergence guarantees. Alternatives to the above best case scenario are:

- **Slowly converging or endless radius optimization process.** Sometimes, the radius optimization process may be stuck at a point where the total number of nucleotides in the current routing is closed to the target but does not exactly match it. When the current number of nucleotides is close enough to the target, it is possible to interrupt the radius optimization process by pressing the “Finish” button. In that case, the rise of the helices will be slightly adjusted so that the total number of nucleotides in the final routing matches the target. As discussed in Chapter 10, we believe that it is safe to

interrupt the radius optimization process if the current number of nucleotides is within 1% of the target length (that is off by  $\sim 70$ -80 nts).

- **Collapsing/Explosion of the shape.** Sometimes the helices are so twisted that reducing the revolution radius *increases* the number of nucleotides in the helices. When that happens the shape will eventually collapse on itself and the routing process will give no satisfying results. It may also happen that the physics engine that equilibrates the inter helices distances diverge, leading to an “explosion” or collapsing of the shape. In these cases, the simulation can be aborted using the “Abort” button. Adjusting the spirality parameter or the revolution radius may increase the probability of success of the routing process.

### 11.1.2 Editing the design

**2D representation of twisted revolution surfaces.** In order to create a 2D embedding of the helices routed around twisted revolution surfaces, we apply a strategy similar to the one we use for curved bundle of helices. Each segment of spiral follows a path  $[0, 1] \rightarrow \mathbb{R}^3$ , and each time  $t \in [0, 1]$  corresponds to a section of the surface. Therefore, we also ensure that each section is mapped to vertically aligned cells in the 2D view. As for curved bundle, the width of the cell representing each nucleotide  $n$  depends on the value  $w_n = t_{n+1}^f - t_n^f$  where  $t_i^f$  is defined in Equation (8.4) in Section 8.1. As an example, Figure 11.2 shows the 2D representation of a twisted torus design (see Section 11.2).

**Navigating the 3D camera around a twisted revolution surface.** ENSnano takes advantage of the parametrization of twisted revolution surface to offer an easy way to navigate the 3D camera around them. To navigate around the surface with the camera, double-click on a nucleotide, hold the Shift key and drag the mouse while holding the middle button.

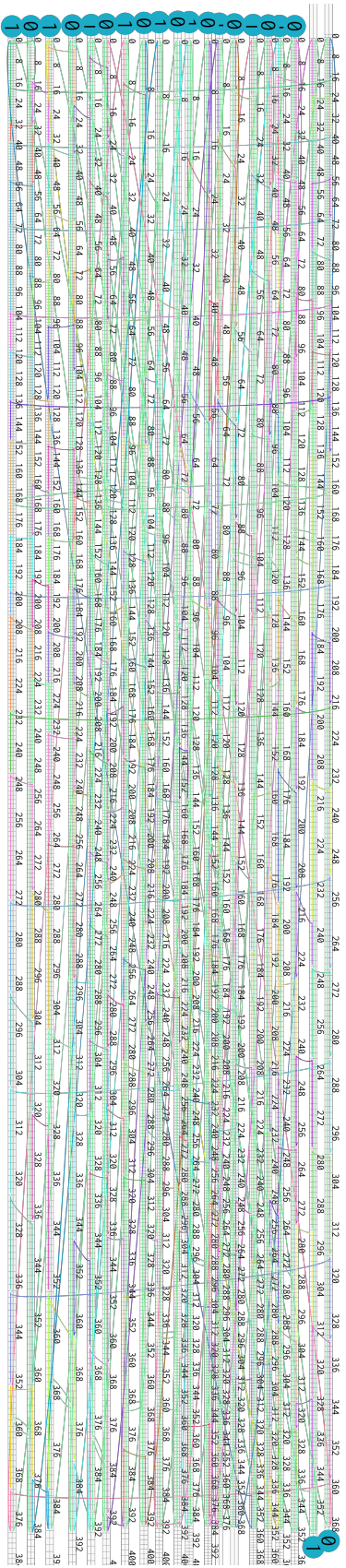
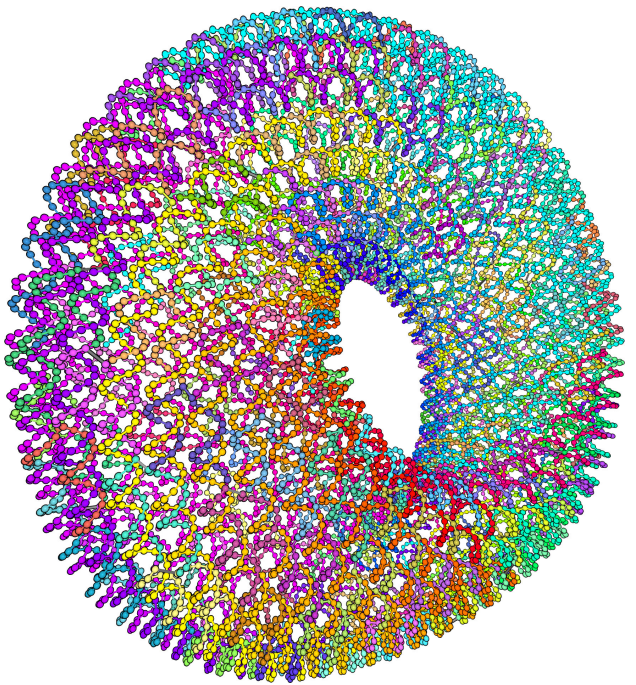
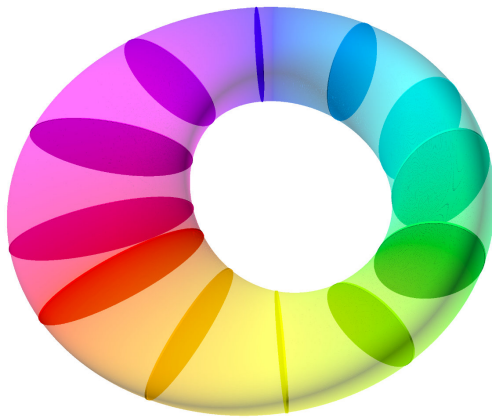
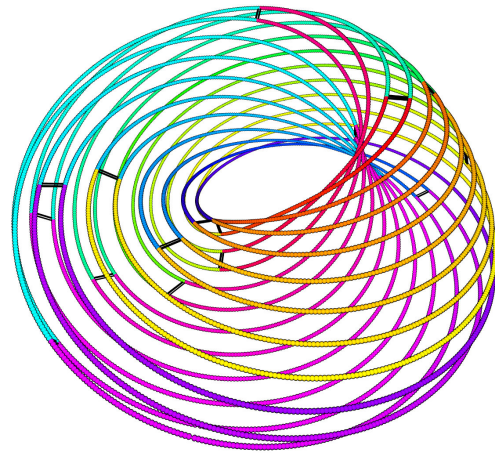


Figure 11.2: **3D (Top) and 2D (bottom) views of a twisted torus design in ENSnano.** The design is made of 2 spiraling helices, each made of 10 segments.

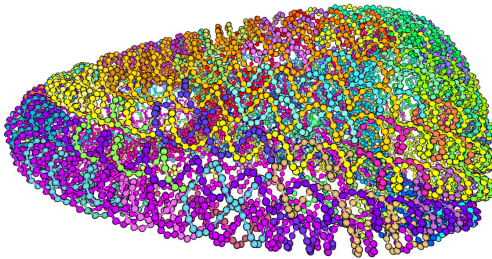




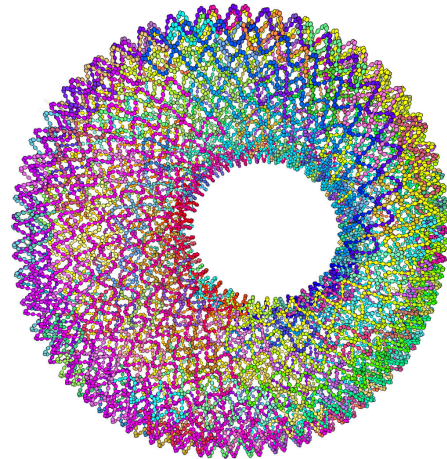
(a) Shape



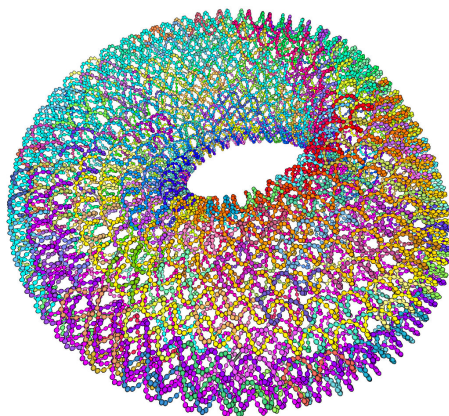
(b) Scaffold route



(c) Front view



(d) Top view



(e) Three-quarters view



(f) Three-quarters view (filled helices)

Figure 11.3: **The Möbius torus design.** The shape of this design is that of a twisted revolution surface with an elliptic section that makes half a turn in one revolution.

## 11.2 Experimental validations

Using ENSnano’s pipeline, we created several designs with complex 3D curvatures. If most design folded properly on the first try, a few did not. When this happened, we always managed to fix faulty designs by only redesigning the route of the scaffold strand, without touching the staples or making any changes in the geometry of the shape. This confirmed that the geometric model of ENSnano was not at cause, and provided some insights on what constitutes a good scaffold routing strategy.

In the first part of this section we only present the result that were obtained with the final versions of our design. We will discuss the few faulty designs and the strategy that we employed to fix the scaffold routes in Subsection 11.2.4.

### 11.2.1 Möbius torus

Our first origami with 3D curvature is a twisted revolution object whose sections are ellipses (Figure 11.3a). In a full revolution around the center of the shape, the section is rotated by half a turn. The resulting shape is that of a supercoiled tube closed on itself (Figure 11.3a) that we call a *Möbius torus*. We chose to use an elliptic section instead of a circular one to create an asymmetry that would hopefully be visible when imaging the origami. On one side of the central hole, the toric shape would be thin and tall, while being thick and small on the opposite side.

Note that we designed the Möbius torus origami before developing the more general concept of twisted revolution surfaces and the automatic pipeline presented in Chapter 10. The revolution radius was therefore optimized manually to use the p7560 scaffold.

**Assembly and characterization** Nicolas Schabanel (NS) annealed the Möbius torus origami and produced AFM images of non purified samples (Figure 11.4).

NS annealed the Möbius torus origami in  $1\times$  TAE with 12.5 mM  $\text{MgCl}_2$  using 5nM of p7560 scaffold and 50 nM of staple strands. The annealing ramp was  $-1^\circ\text{C}/5\text{min}$  from  $65^\circ\text{C}$  to  $45^\circ\text{C}$ .

Allan Mills (AM) annealed the Möbius torus, and produced TEM images of samples that were purified on 1% agarose gel (Figure 11.5). AM annealed the Möbius torus origami in  $1\times$  TAE with 12mM  $\text{MgCl}_2$  using 20nM of p7560 scaffold and 200nM of staple strands. The annealing ramp was: hold 15 min at  $65^\circ\text{C}$ , then descent from  $60^\circ\text{C}$  to  $40^\circ\text{C}$  at  $-1^\circ\text{C}/2\text{h}$ .

The Möbius torus origami was also characterized by Cryogenic electron microscopy, the collected data allowed to reconstruct the 3D structure of the origami (Figure 11.6). Here is the list of microscope that were used:

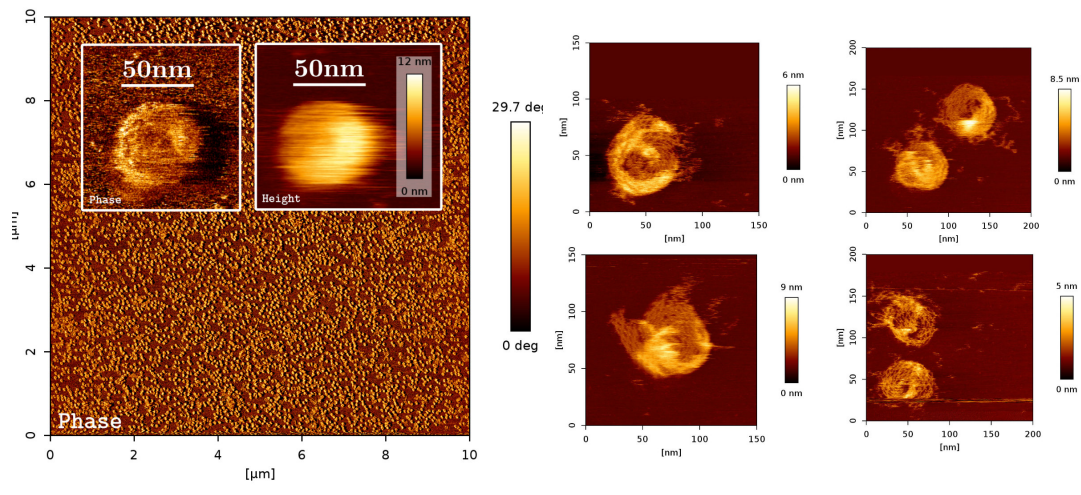
Microscopes from the Cryo-Electron Microscopy CBS facility, Montpellier, France:

- LaB6 microscope working at 120kV, equipped with a GATAN One View camera.
- Cryo Transmission Electron Microscope JEOL 2200FS FEG operating at 200 kV equipped with a GATAN direct detector K3 camera. The CBS is a member of the French Infrastructure for Integrated Structural Biology (FRISBI), a national infrastructure supported by the French National Research Agency (ANR-10-INBS-05).

Microscopes from the Cryo-Electron Microscopy C0re facility Brno, Czech Republic:

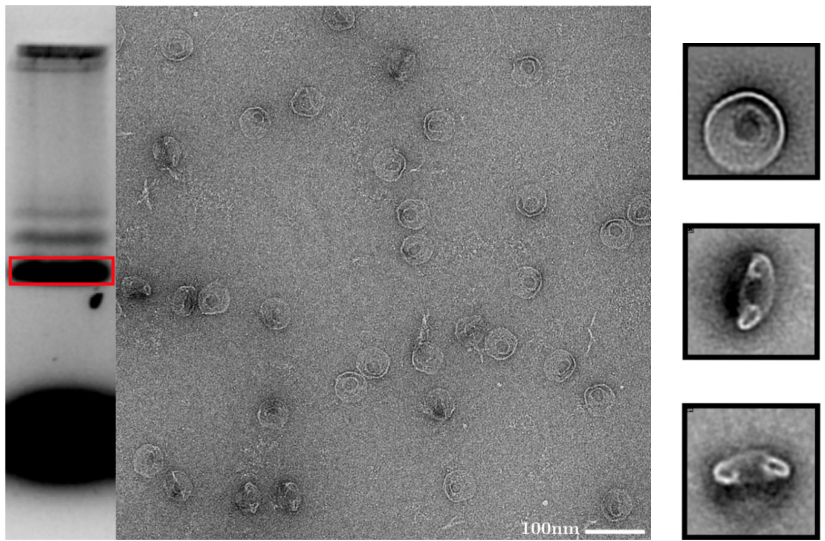
- Talos Arctica – 200 kV high-end transmission electron microscope equipped with Ceta camera, Falcon 3EC direct electron detector, and post-GIF K2 direct electron detector (Bioquantum 967).
- Titan Krios - 300kV high-end transmission electron microscope aligned for fringe free imaging (FFI) and equipped with Volta phase plate (VPP), Falcon 3EC direct electron camera, energy filter with K3 direct electron detector (Bioquantum 1067).





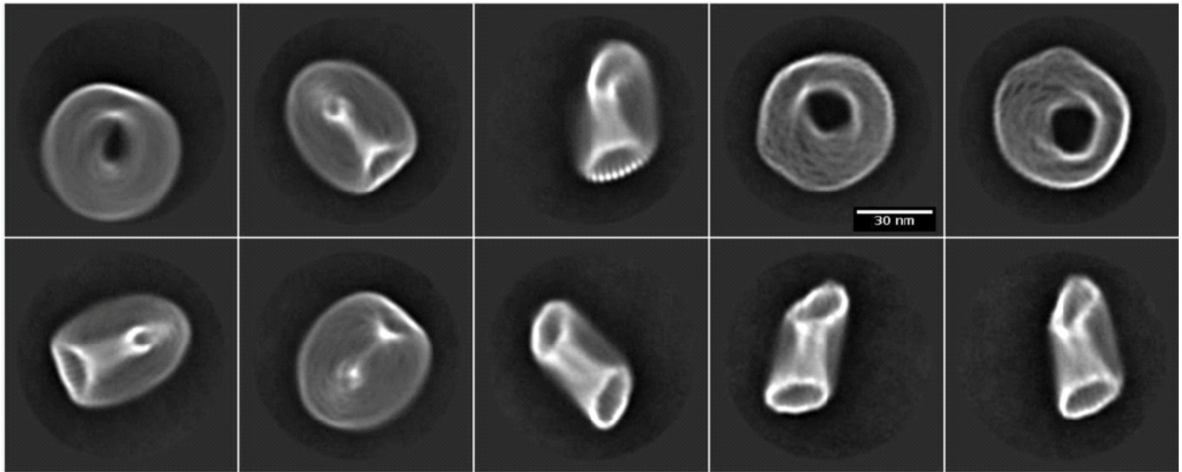
(a) AFM image of a 2.5  $\mu\text{L}$  unpurified sample (b) AFM images of “exploded” Möbius origami, revealing the underlying structure with 2.5  $\mu\text{L}$  of 22 mM  $\text{NiCl}_2$ .

Figure 11.4: **AFM images of the Möbius torus origami.** The origami were annealed and imaged by Nicolas Schabanel.

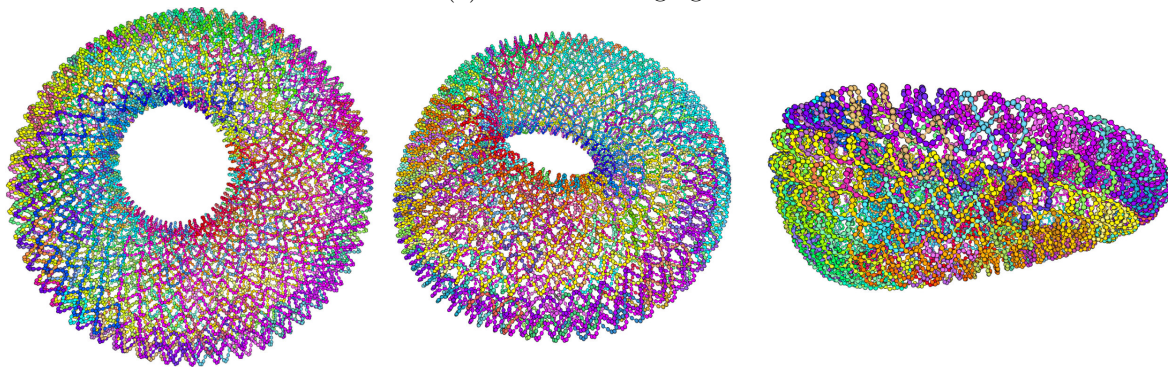


(a) Electrophoresis and TEM image of a purified sample (b) 2D class averaging

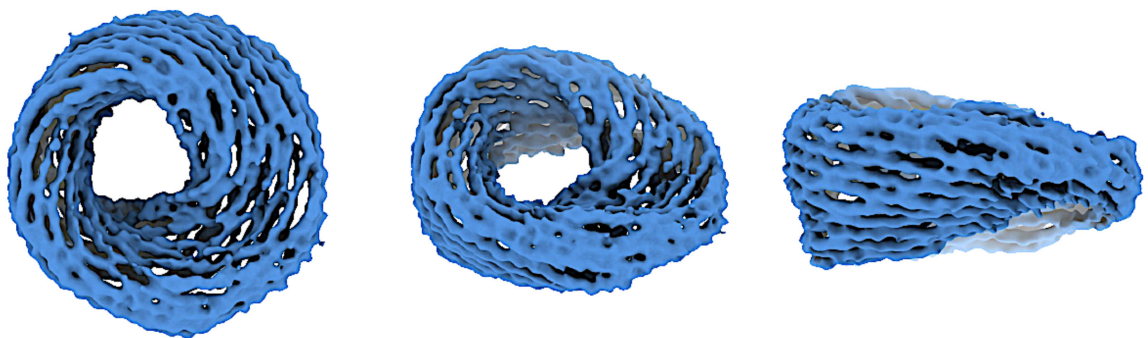
Figure 11.5: **TEM characterization of Möbius torus origami.** Annealing and TEM images by Allan Mills.



(a) 2D class averaging



(b) 3D view of the design in ENSnano



(c) 3D reconstruction from high resolution Cryo-EM data.

Figure 11.6: **High resolution Cryo-EM images of the Möbius torus origami.**

### 11.2.2 Toruses with higher number of twists

After developing our automatic pipeline for routing helices around twisted revolution surfaces, we designed 5 additional origami with the shape of twisted toruses. These origami are twisted revolution surfaces with an elliptic section that makes between 2 and 6 twists (half rotations) on itself in a revolution (Figure 11.7). In all the toruses design, the section is an ellipse whose major axis is twice as long as the minor axis. The revolution radius of all the design was optimized to use a p7560 scaffold. The geometric parameters of each twisted torus can be found in Table 11.1.

# twist	# spiral segments	spirality	# spirals	Ellipse minor axis	Revolution radius
2	18	$k - 4$	2	5.06 nm	21.13 nm
3	16	$k = -6$	2	4.54 nm	23.48 nm
4	16	$k = -10$	2	4.78 nm	22.20 nm
5	14	$k = -5$	2	4.35 nm	24.45 nm
6	14	$k = -12$	2	4.46 nm	24.00 nm

Table 11.1: **Geometric parameters of the twisted toruses design.**

**Optimization of the folding protocol.** Julie Finkel (JF) performed several rounds of annealing and gel electrophoresis to identify an optimal folding protocol for the torus with 6 twists.

The optimal folding conditions were in  $1\times$  TAE with 18 mM  $\text{MgCl}_2$  using 20 mM p7560 scaffold and 200 mM of staple strands. The folding ramp was: Hold 15 min at  $65^\circ\text{C}$  then fast descent from  $65^\circ\text{C}$  to  $55^\circ\text{C}$  at  $-1^\circ/10\text{min}$  then slower descent from  $55^\circ\text{C}$  to  $20^\circ\text{C}$  at  $-1^\circ\text{C}/1\text{h}$ . JF also followed this optimized protocol for the toruses with 2,3 and 4 twists.

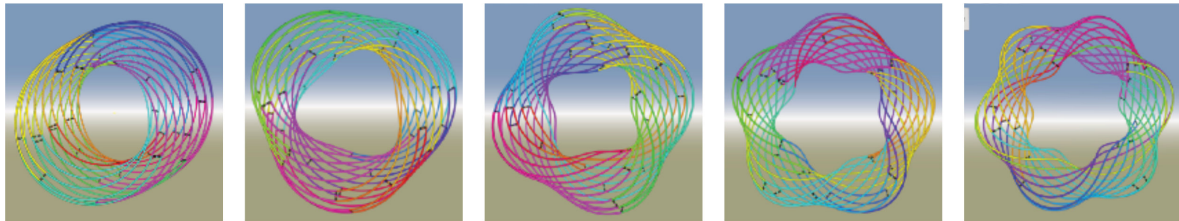
This protocol was however not optimal for folding the torus with 5 twists. After several other rounds of annealing and gel electrophoresis, JF identified the following protocol for folding the 5-twists torus: fold in  $1\times$  TAE with 18 mM  $\text{MgCl}_2$  with 5 nM p7560 scaffold with 50 nM staple strands. The folding ramp was 15 min hold at  $65^\circ\text{C}$  followed by a fast descent from  $60^\circ\text{C}$  to  $40^\circ\text{C}$  at  $-1^\circ\text{C}/10\text{min}$ . Compared to all the other annealing protocol that are used in the section, this one uses significantly lower concentrations of scaffold and staple strands. This is because we believe that at higher concentration, staples have a tendency to bind to several scaffold strands, creating undesired multimeric assemblies. We discuss this further in Subsection 11.2.4.

**Characterization** JF purified the 2,3,4,5 and 6-twists toruses on 1% agarose gel and imaged them with TEM (Figure 11.8).

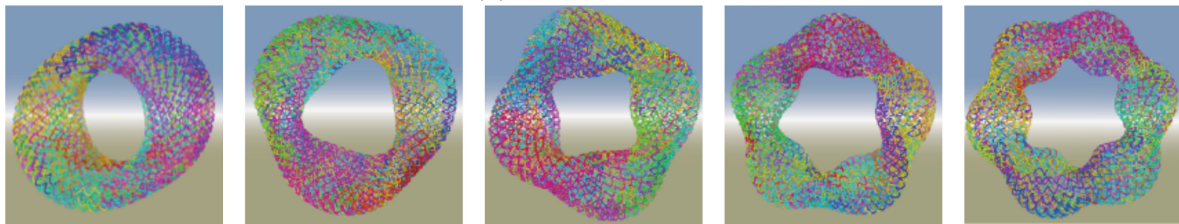




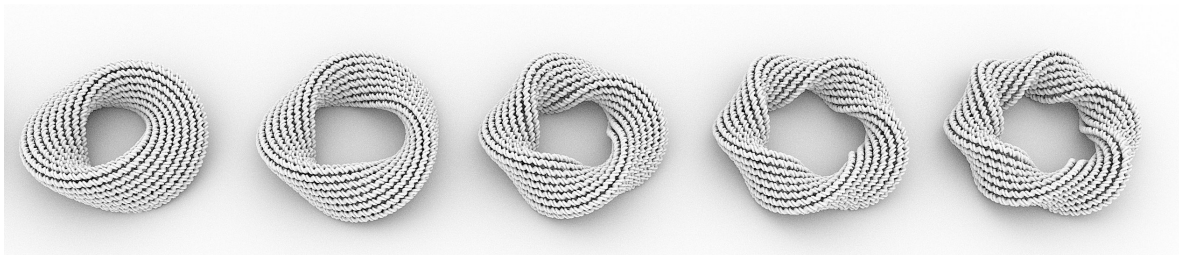
(a) Shape of the designs.



(b) Scaffold routings



(c) 3D view of the design in ENSnano



(d) 3D view of the design (filled helices)

Figure 11.7: **The 2-,3-,4-,5- and 6-twists torus designs.** The shapes of these designs are twisted revolution surfaces with an elliptic section that makes between 2 and 6 half-turns in a revolution.

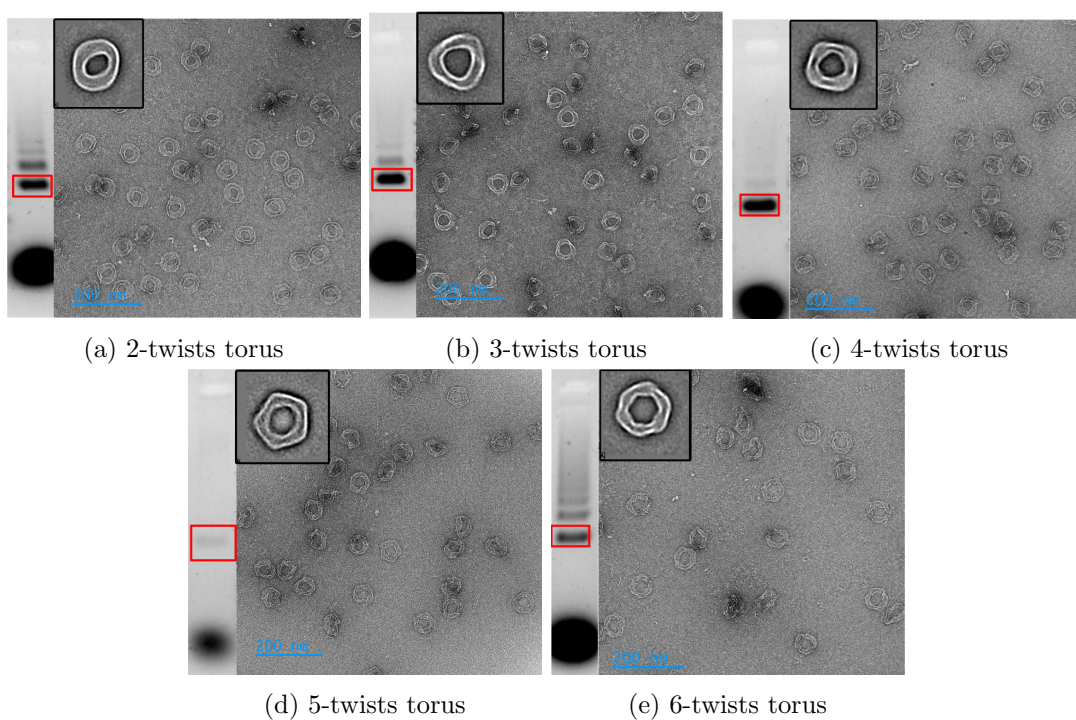
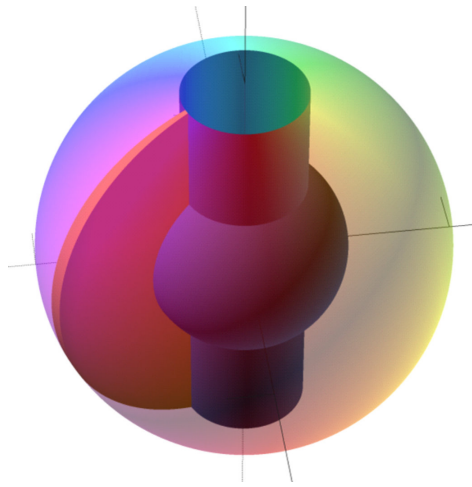
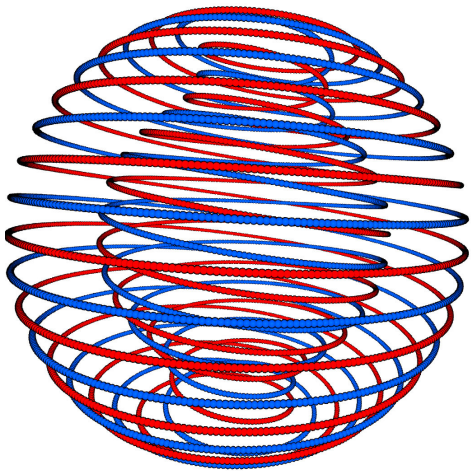


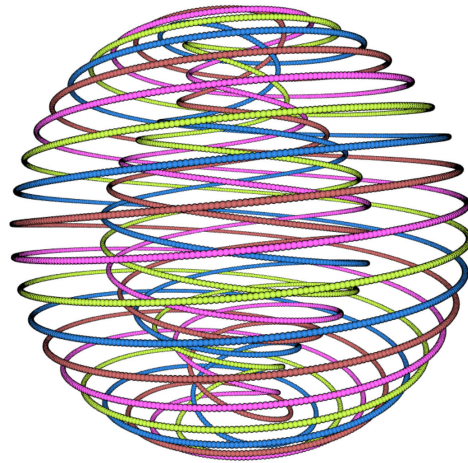
Figure 11.8: **Electrophoresis gel, and TEM characterization of the twisted toruses origami.** The origami were annealed and characterized by Julie Finkel.



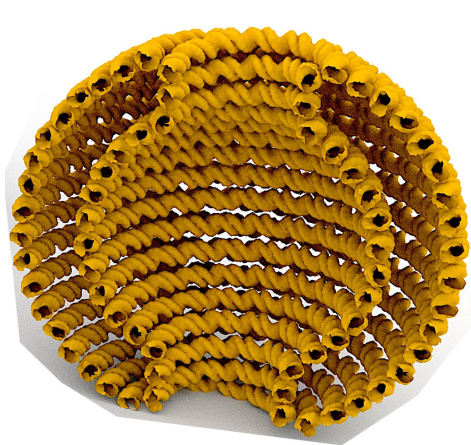
(a) Shape of the designs



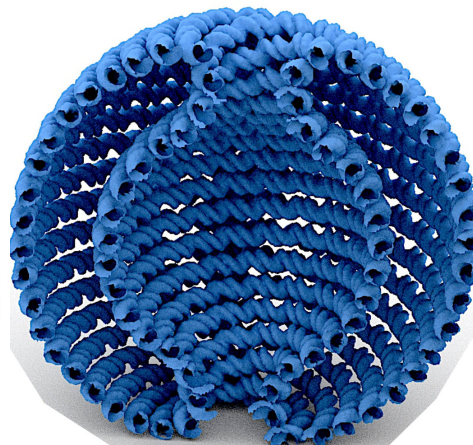
(b) Helices routing of the double-sphere V1



(c) Helices routing of the double-sphere V2



(d) Section of the double-sphere V1



(e) Section of the double-sphere V2

Figure 11.9: **The double-sphere designs.** These designs consist of two concentric spheres linked by a tubular junction. They can be seen as a revolution surface whose section has the shape of a ‘C’.



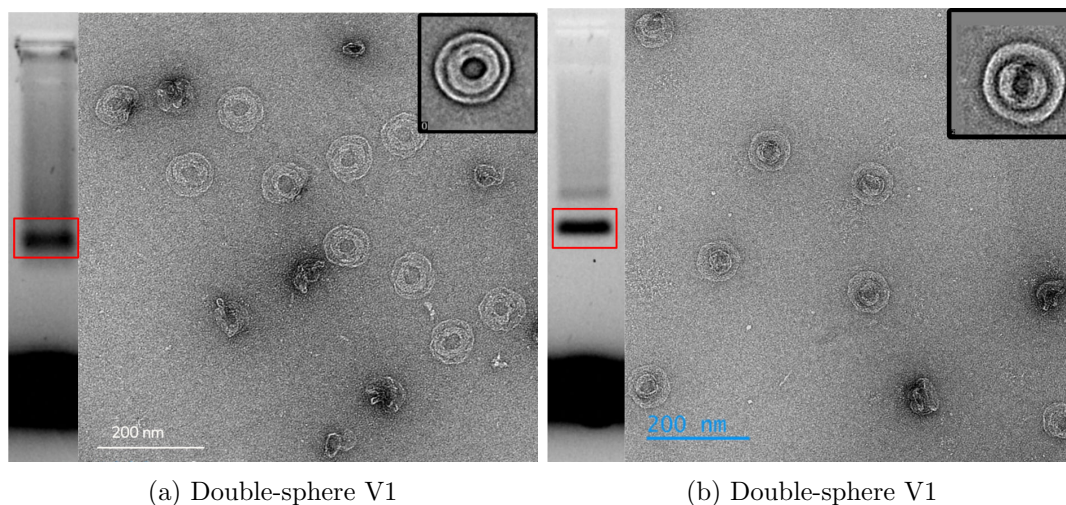


Figure 11.10: **Gel electrophoresis and TEM characterization of the double-sphere origami.** The origami were annealed and characterized by Julie Finkel.

### 11.2.3 Double-sphere origami

Using our automated pipeline for revolution surfaces, we designed origami whose shape was made of two concentric shapes linked by a tubular section (Figure 11.9).

This shape can be seen as a revolution surface whose section has a ‘C’ shape with two half-circles connected by two segments (Figure 11.9a).

We made two versions of this shape, one with 2 spirals that we call the *double-sphere V1* (Figure 11.9b), and one with 4 spirals (Figure 11.9c) that we call the *double-sphere V2*. Using 4 spirals in the double-sphere V2 allowed to create a design with a thinner tubular section without imposing the helices to have a small radius of curvature.

**Annealing and characterization.** JF annealed the double-sphere V1 origami in  $1\times$  TAE with 18 mM  $\text{MgCl}_2$  using 20 nM p8064 scaffold and 200 nM of staple strands. The folding ramp was 15 min hold at 65° C followed by a fast descent from 60° C to 40° C at  $-1^\circ\text{C}/10\text{min}$ .

JF annealed the double-sphere V2 origami with almost the same protocol. The only difference is that the optimal  $\text{MgCl}_2$  concentration for annealing this design appeared to be 16 mM.

JF characterized both double-sphere origami by TEM after purification on 1% agarose gel (Figure 11.10).

#### 11.2.4 Discussion on the importance of scaffold routing

**Möbius torus.** Our first version of the Möbius torus was designed with the goal of minimizing the number of crossovers along the scaffold strand. This design had only one double crossover on the scaffold strand (Figure 11.11a) and failed to fold.

Here is how we interpreted this failure: because of the scaffold route that we used, almost all staples of the design were complementary to two very distant domains on the scaffold (Figure 11.11b, Left). This means that almost no constraints were exerted on the scaffold strand to adopt a spiraling shape. We now believe that for this first design to fold correctly, the scaffold strand need to make the right number of turn before the staple start to attach which is extremely unlikely.

We therefore redesigned the scaffold route to avoid binding domains that should be several turns away (Figure 11.11b, Right). We notably chose the scaffold route so that the scaffold strand went through a crossover at every turn of the spiral. Note only the scaffold routing was redesigned. We did not modify the 3D helices routings or redesign the staple strands.

**Toruses with 3 and 5 twists.** The first version of the 5-twist torus was difficult to fold. Gel electrophoresis revealed that only a tiny proportion of the origami folded correctly, most of the strands aggregating in a band close to the well (Figure 11.12b).

Gel electrophoresis of the first version of the 3-twists torus revealed the existence of two species of assembly. TEM images of the most abundant one revealed that during the folding process, undesired multimeric assemblies were formed (Figure 11.13b). This was likely due to some staple strands binding several scaffold strands, instead of several domains of one single scaffold strand. We did not image such multimeric assemblies for the 5-twist torus, but we assumed that a similar phenomenon was inferring with its assembly.

We managed to significantly reduce the tendency of the 3-twists torus to form multimeric assembly by redesigning the scaffold route (Figure 11.13a). The crossovers of the scaffold strand were aligned on straight cuts in order to prevent the appearance of “finger-like” patterns that we assumed to increase the probability of multimerisation.

The 5-twists torus design was improved by increasing the density of crossovers in the scaffold strand. As for the Möbius torus, our goal was to keep domains on of the scaffold strand as straight as possible between two crossovers. However, due to geometric constraints we were not able to organize the crossovers in straight cuts as we did for the 3-twists torus. This may explain why it was more difficult to find good folding conditions for this design than for the other ones.

**Double-sphere origami.** Images of our first version of the 4 spirals double-sphere origami revealed structures presumably made of four half-spheres concatenated in various configurations (Figure 11.14a), that is: the desired structure cut open in halves.

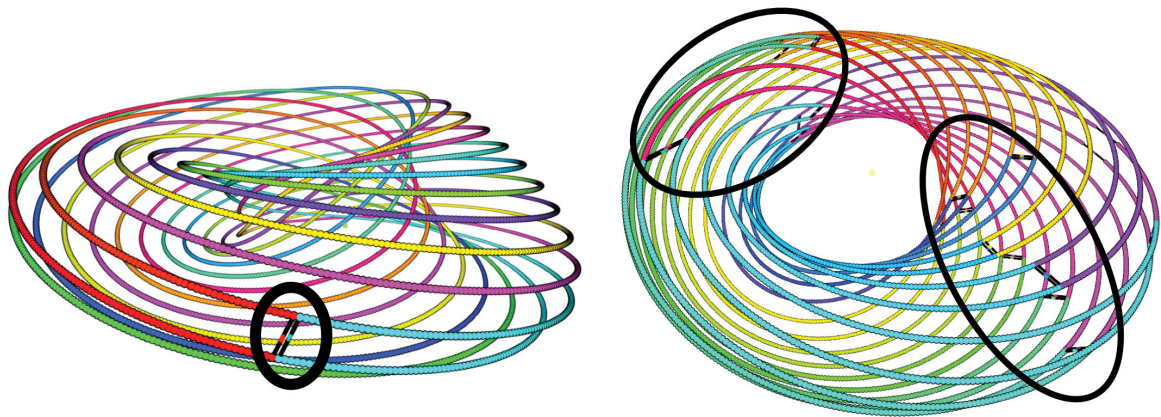
We postulated that this was due to the fact that the outer hemispheres were folding first, and that there was not enough constraint to force the concavity of the outer hemisphere to be on a specific side, leading to stable shapes with four hemispheres linked in a “necklace”.

We therefore redesigned the scaffold route so that the crossover would cut the outer sphere in four parts (Figure 11.14b). With this new scaffold route, the proportion of correctly folded origami increased significantly.

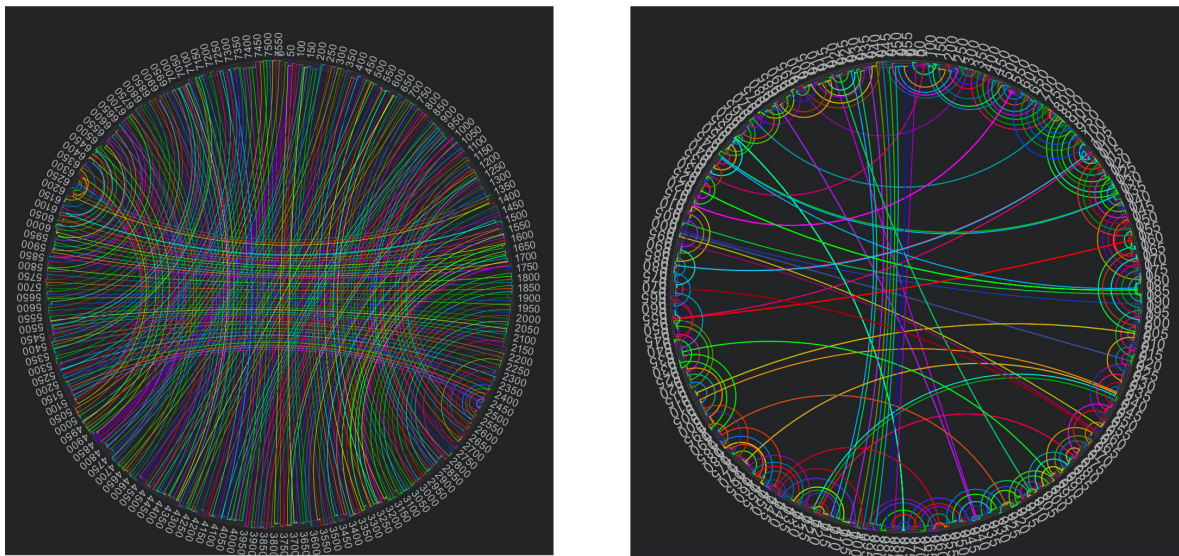
In conclusion, we found that the topological and geometric constraints exerted by the



scaffold route can significantly impact the foldability of the origami. This can be put in perspective with other studies on scaffold routing e.g. [KBV<sup>+</sup>12, DDO<sup>+</sup>15, DDB<sup>+</sup>15, SMD19].

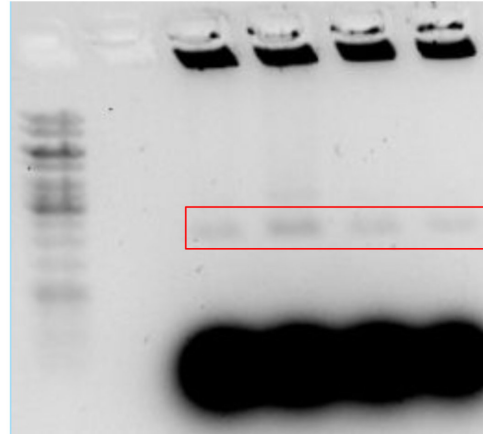


(a) Scaffold routes

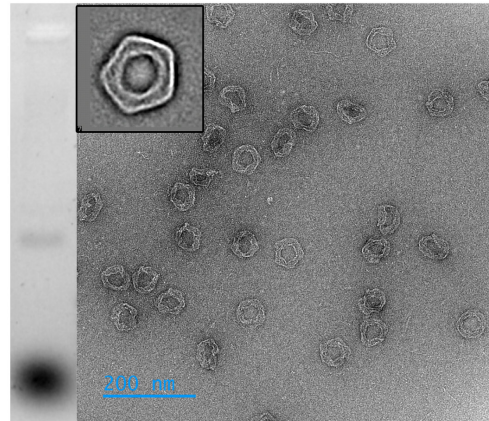
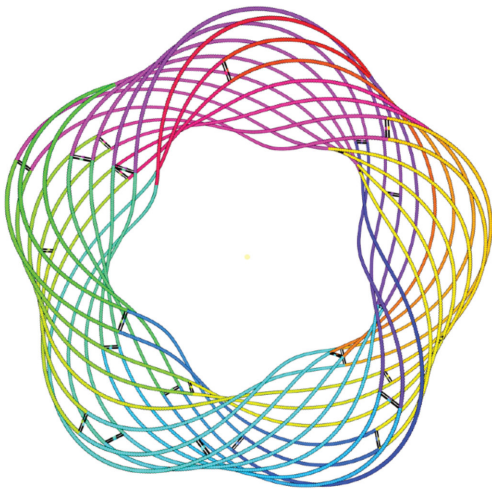


(b) Representation of the scaffold's domains bound by each staple strand.

Figure 11.11: **Importance of the scaffold route for the Möbius torus design.** a: The two scaffold route that we tried. b: Representation of the scaffold's domains bound by each staple strands. The circle represent the scaffold strand. Each domain of a staple is represented by an arc opposing its complementary region on the scaffold. The domains of a staple are connected by chords that represents the crossovers of the staple.



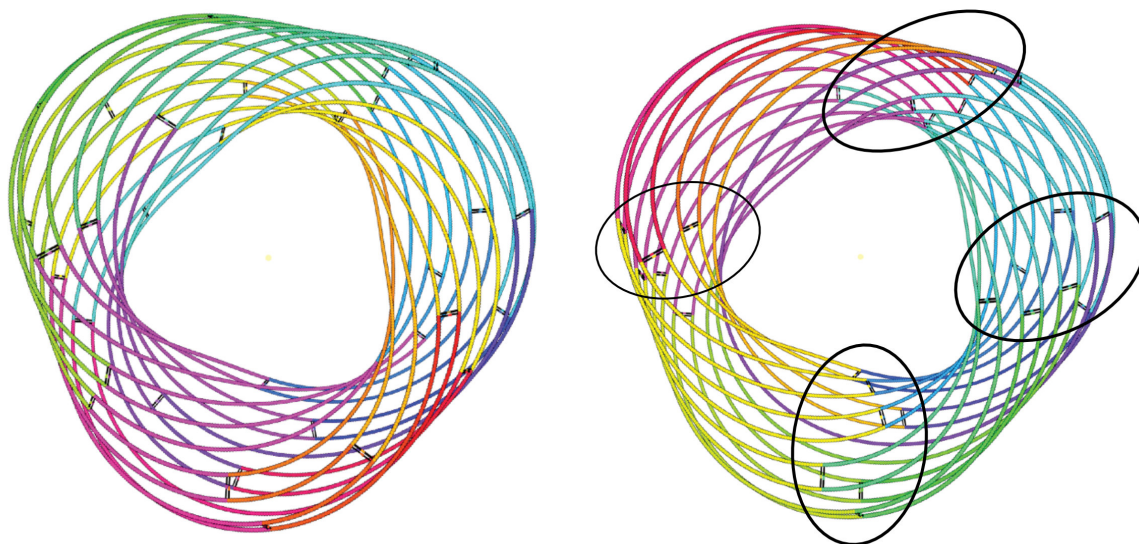
(a) Initial scaffold routing of the 5-twist torus (b) Gel electrophoresis of the origami in the initial design



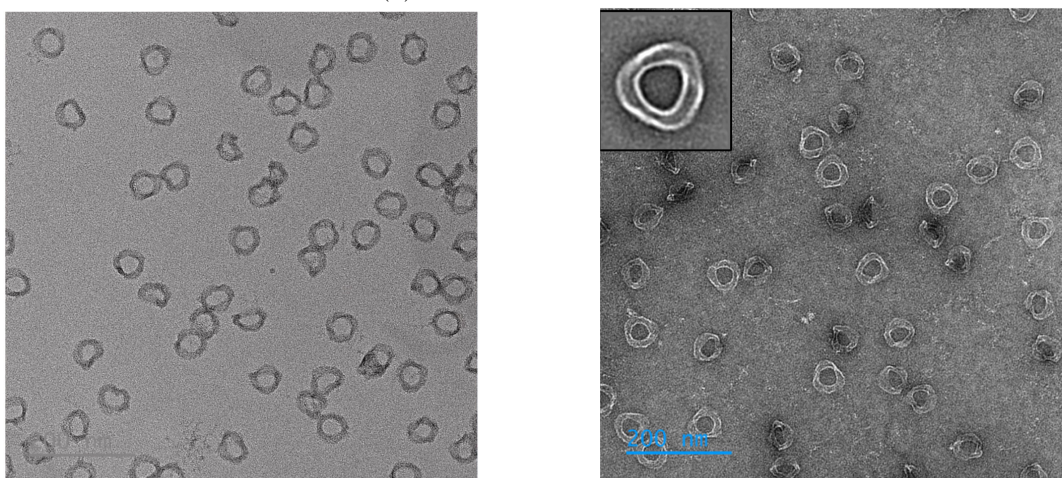
(c) Corrected scaffold routing of the 5-twists (d) TEM characterization of the corrected design

Figure 11.12: **The two scaffold route for the 5-twists torus.** In the first routing (Panel a), the scaffold route makes only two cuts in the surface. This scaffold route does not constrain the shape enough and only a few origami fold correctly. We redesigned the scaffold route so that the scaffold strand always meets a crossover before making a full turn on the section (Panel c) this significantly improved the yield (Panel 11.12d).



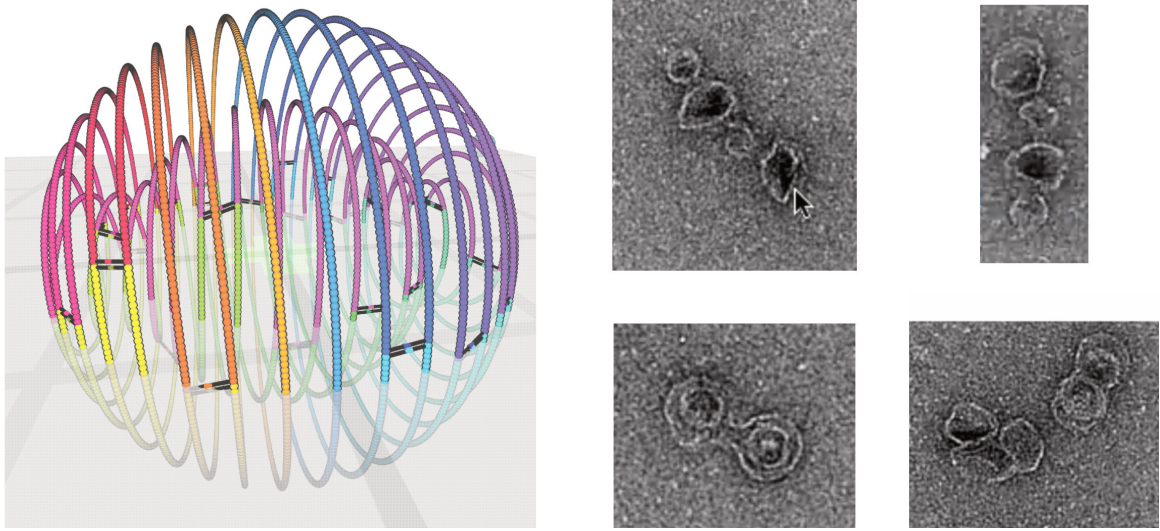


(a) The two scaffold routes

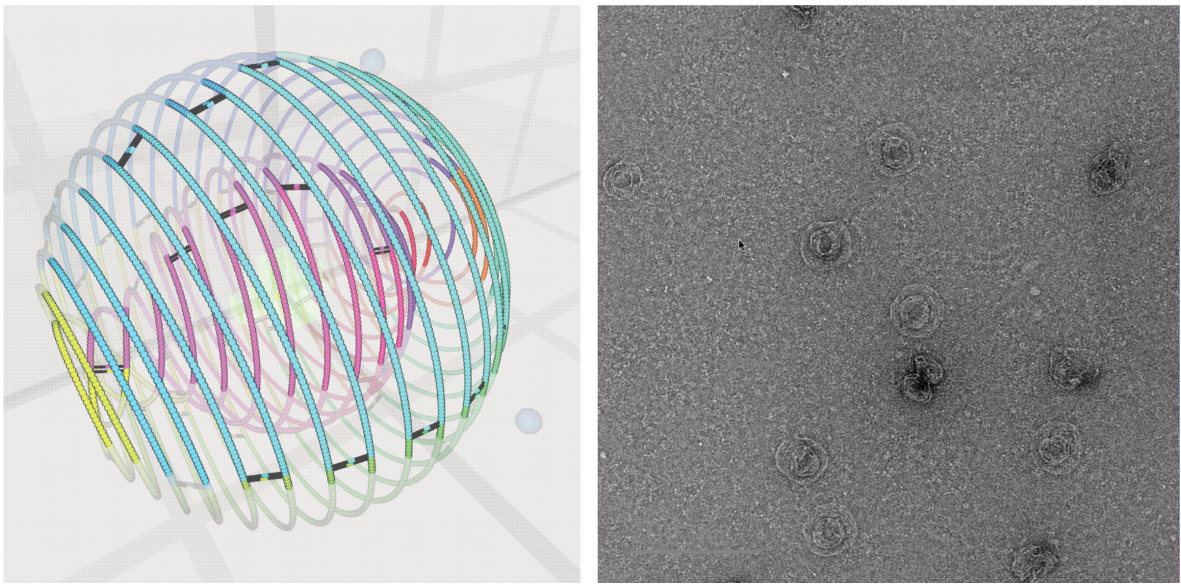


(b) TEM images of the origami

Figure 11.13: **The two scaffold routes that we tried for the 3-twists torus.** On the left, the scaffold's crossovers are not aligned. This may lead to the formation of finger-like patterns, grabbing finger from other origamis, that would explain the formation of multimeric structures. We fixed the design by aligning the scaffold's crossovers on sections.



(a) First scaffold route with only two cuts on the outer sphere.



(b) Second scaffold route with four cuts on the outer sphere.

Figure 11.14: **The two scaffold routes that we tried for the double-sphere V2 origami.** a: This first route had only two cuts on the outer sphere and lead to various shapes made of 4 concatenated hemispheres. b: The design was fixed by having the scaffold route separate the outer sphere in 4 parts instead of two.

# Conclusion and perspectives

The primary objective of our work was to set the basis for a complete GUI based platform for designing high fidelity 3D DNA nanostructures. The current version of ENSnano is close enough to that goal as it allowed us to fully design several complex DNA origami without using any other software during the process. Compared to standard design methods, we believe that ENSnano offers a significant simplification of the design pipeline.

## Summary of contributions



In Part II, we developed a hierarchical data structure for representing DNA origami. This data structure is based on the concept of grids to group parallel helices. Grids in ENSnano significantly ease the design of modular structures. Like other software (cadnano, scadnano) we separate the topology of the design from its geometric structure in a dedicated 2D view. ENSnano however stands out by transparently exposing the geometry of the design via a 3D view.

We also developed an integrated physics engine to automatically adjust the roll of the helices, perform quick stability tests and infer the geometry of cadnano designs.

All this was implemented in the first version of our software ENSnano. ENSnano offers both 2D and 3D views of the design. The 2D interface is inspired from cadnano's and allows to seamlessly edit the topology of the design. The 3D interface is used to visualize and edit the geometry of the design. It is also well suited to perform some operations that require information about the 3D position of nucleotides like creating crossovers between non-parallel or curved helices. The 2D and 3D view of the design are fully synchronized: modifications of the design made in one view are immediately visible in the other one.

We concluded Part II by an experimental validation of our software. To do so, we designed in ENSnano an origami made of two non-parallel layers and successfully annealed it, without prior validation from an external physics simulation software.

In Part III, we developed a *new geometric model for curved DNA helices*. This model based on a “lazily adaptive frame” algorithm allows to generate the 3D positions of nucleotides in DNA helices with arbitrary 3D curved axis.

In order to design curved DNA origami in ENSnano, we first designed an interface for creating curved bundle of helices with the shape of Bézier curves. This allowed us to design in ENSnano a 6-helices bundle origami with the shape of a looped square . This origami was successfully annealed, and several variants of it were tried. Characterization of the variants of the  design showed that the crossover pattern that we designed with ENSnano was enough to constrain locally the curvature of the design. However, special staples were required to

globally enforce the desired  $\infty$  shape, which seems anyway unavoidable as demonstrated by our plumber pipes experiment.

Our final contribution is a *spiral-based method for routing DNA helices* around twisted revolution surfaces. This method relies on a physics system to balance inter-helices distances and a radius optimization procedure that outputs a routing whose total length matches exactly the desired scaffold length. Using this method, we designed a series of design with 3D curvature: twisted toruses whose section makes between 1 and 6 half rotations in a revolution, and shapes made of two concentric spheres linked together by a thin tubular junction. These designs with unprecedentedly complex shapes were successfully annealed and characterized.

Again, all these origami were designed entirely in ENSnano, without relying on physics simulation software for validation. Their successful assembly is a strong evidence of the validity of our model for curved DNA.

## Future work

Our long term goal for ENSnano is to provide a versatile tool for the design of DNA nanostructures.

Throughout this work, we have demonstrated the efficiency of ENSnano's interface and the reliability of its geometric model of DNA. However, our software could still benefit from several features that would make it more powerful and versatile.

**Automation tools.** Several software such as cadnano, offer automatic staples generation algorithms. This feature is, currently, notably absent from ENSnano. We believe that a simple local search approach should be enough for that purpose, and we hope to be able to implement one in ENSnano very soon. We also want to stress that, in our opinion, automatic strands generation algorithms should only be used together with a *correct geometric model of DNA*. Otherwise, the outputs needs to be adjusted anyway, using feedbacks from simulations software or experimental results.

Generating automatically a scaffold route seems however to be a more challenging task. There exists several software that offer automatic scaffolding, these are however always only applicable to specific classes of shape. Software from the deadalus suite are restricted to wireframe origami, MagicDNA only deals with design made of concatenated bundles, and DNAxis is restricted to axially symmetric structures. Moreover, the outputs of these software often needs to be re-imported in cadnano for manual adjustment.

Among the issues that need to be addressed by a general-purpose automatic scaffolding algorithm is a way to evaluate the quality of the proposed routing. As discussed in Chapter 11, an important criterion seems to be the distribution of the distance between the scaffold domains bound by individual staples. A high-enough crossover density on the scaffold strand seem to be required in order to enforce enough geometric constraints. Moreover, it seems that crossovers on the scaffold strand have a collaborative effect when it comes to preventing multimerisation of origami during folding. Looking at each crossover individually may therefore not be enough, and global constraints may be required to express evaluation criteria for the scaffold routing. A general purpose scaffolding algorithm should therefore take into account geometric and topological constraints as well.



**Multiple design in one file.** In Chapter 9, we designed several variants of a single origami. Currently, each of these variants needs to be stored in its specific file. Implementing a layering system in ENSnano would allow to “factor” these variants in a single file. The strands shared by all variants would belong to the base layer while variant-specific strands would live in their own layer.

Such a layer-based organization of the design would also allow to automatically organize the 96-wells plates of staples in a way that makes it easy for the experimentalist to mix the staples corresponding to the desired variant.

**Sequence generation for non-scaffolded designs.** Generating the sequences of the staples in a DNA origami is trivial, as they can simply be deduced from the sequence of their complementary domains on the scaffold.

Generating sequences for the strands of a non-scaffolded design is more difficult. This is because big sets of strands may contain pairs of strands that were not *designed* to be complementary but have an “accidentally” high affinity for each other.

Our current plan is to extend ENSnano’s data structure with a notion of subdivision of strands into abstract domains. These abstract domains would each belong to some “domain family” having its own constraints (affinity with other family, orthogonality etc...). This division into abstract domains would be independent of the region of virtual helices that serve as support for the strand, and we would use this domain-based representation to generate inputs for specialized sequence generators.

**Automatic layout of the 2D view.** The possibility to freely organize the 2D view facilitates the design of structures with non-parallel layers.

It is however not always easy to manually find an ergonomic organization of the 2D view. The optimal organization of the 2D view may also depend on the specific region of the design that is being edited.

The designer experience may be greatly improved by the possibility to automatically organize the 2D view of a set of helices. Being able to save several organizations and to quickly change from one to another is also a desirable feature.

Possible leads to develop these features include the use of stereographic projections to map the 3D coordinates of the helices to 2D, and the use of linear programming to organize the helices so that their 2D representations do not collide.

**Design of single-stranded domains** In multi-components design, single stranded “bridges” of DNA are sometimes used to connect the different component of a design [TPQ17a, TPQ17b]. It would be helpful to have a way to automatically create those single-stranded connections with the correct length. This could be done by using ENSnano’s geometric model to compute the physical distance between the two connection points and infer the appropriate number of nucleotides between them.

**Improvement to the stability testing features** The simulations of single stranded loops or connections by our physics engine could be improved to assist the designer in designing multi-components structures.

It would be helpful to extend the model of our physics engine developed in Chapter 5 to include fluctuations at nick that would highlight open seam within a DNA origami that are

left open instead of being properly connected by bridges staples.

# Bibliography

- [Adl94] Leonard M Adleman. Molecular computation of solutions to combinatorial problems. *science*, 266(5187):1021–1024, 1994.
- [Adl98] Leonard M Adleman. Computing with DNA. *Scientific american*, 279(2):54–61, 1998.
- [ADN<sup>+</sup>08] Ebbe S Andersen, Mingdong Dong, Morten M Nielsen, Kasper Jahn, Allan Lind-Thomsen, Wael Mamdouh, Kurt V Gothelf, Flemming Besenbacher, and Jørgen Kjems. DNA origami design of dolphin-shaped structures with flexible tails. *ACS nano*, 2(6):1213–1218, 2008.
- [AT12] Marco Abate and Francesca Tovena. *Curves and surfaces*. Springer Science & Business Media, 2012.
- [Bar01] David Baraff. Physically based modeling: Rigid body simulation. *SIGGRAPH Course Notes, ACM SIGGRAPH*, 2(1):2–1, 2001.
- [BDLS96] Dan Boneh, Christopher Dunworth, Richard J Lipton, and Jiri Sgall. On the computational power of DNA. *Discrete Applied Mathematics*, 71(1-3):79–94, 1996.
- [BI92] Serge L Beaucage and Radhakrishnan P Iyer. Advances in the synthesis of oligonucleotides by the phosphoramidite approach. *Tetrahedron*, 48(12):2223–2311, 1992.
- [BMG<sup>+</sup>15] Erik Benson, Abdulmelik Mohammed, Johan Gardell, Sergej Masich, Eugen Czeizler, Pekka Orponen, and Björn Högberg. DNA rendering of polyhedral meshes at the nanoscale. *Nature*, 523(7561):441–444, 2015.
- [BMP<sup>+</sup>22] Joakim Bohlin, Michael Matthies, Erik Poppleton, Jonah Procyk, Aatmik Mallya, Hao Yan, and Petr Šulc. Design and simulation of DNA, RNA and hybrid protein–nucleic acid nanostructures with oxView. *Nature protocols*, 17(8):1762–1788, 2022.
- [BSV<sup>+</sup>20] Joshua Bush, Shrishti Singh, Merlyn Vargas, Esra Oktay, Chih-Hsiang Hu, and Remi Veneziano. Synthesis of DNA origami scaffolds: current and emerging strategies. *Molecules*, 25(15):3386, 2020.
- [CKK<sup>+</sup>11] Carlos Ernesto Castro, Fabian Kilchherr, Do-Nyun Kim, Enrique Lin Shiao, Tobias Wauer, Philipp Wortmann, Mark Bathe, and Hendrik Dietz. A primer to scaffolded DNA origami. *Nature methods*, 8(3):221–229, 2011.

- [CQ18] Kevin M Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370–376, 2018.
- [CS91] Junghuei Chen and Nadrian C Seeman. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature*, 350(6319):631–633, 1991.
- [CTP<sup>+</sup>08] Valentino R Cooper, Timo Thonhauser, Aaron Puzder, Elsebeth Schröder, Bengt I Lundqvist, and David C Langreth. Stacking interactions and the twist of DNA. *Journal of the American Chemical Society*, 130(4):1304–1308, 2008.
- [DBC12] Shawn M Douglas, Ido Bachelet, and George M Church. A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070):831–834, 2012.
- [DBG<sup>+</sup>16] Cheikh Tidiane Diagne, Christophe Brun, Didier Gasparutto, Xavier Baillin, and Raluca Tiron. DNA origami mask for sub-ten-nanometer lithography. *ACS nano*, 10(7):6458–6463, 2016.
- [DCS07] Shawn M Douglas, James J Chou, and William M Shih. DNA-nanotube-induced alignment of membrane proteins for nmr structure determination. *Proceedings of the National Academy of Sciences*, 104(16):6644–6648, 2007.
- [DDB<sup>+</sup>15] Frits Dannenberg, Katherine E Dunn, Jonathan Bath, Marta Kwiatkowska, Andrew J Turberfield, and Thomas E Ouldridge. Modelling DNA origami self-assembly at the domain level. *The Journal of chemical physics*, 143(16), 2015.
- [DDL<sup>+</sup>09] Shawn M Douglas, Hendrik Dietz, Tim Liedl, Björn Högberg, Franziska Graf, and William M Shih. Self-assembly of DNA into nanoscale three-dimensional shapes. *Nature*, 459(7245):414–418, 2009.
- [DDO<sup>+</sup>15] Katherine E Dunn, Frits Dannenberg, Thomas E Ouldridge, Marta Kwiatkowska, Andrew J Turberfield, and Jonathan Bath. Guiding the folding pathway of DNA origami. *Nature*, 525(7567):82–86, 2015.
- [DDS09] Hendrik Dietz, Shawn M Douglas, and William M Shih. Folding DNA into twisted and curved nanoscale shapes. *Science*, 325(5941):725–730, 2009.
- [DF96] SD Durbin and G Feher. Protein crystallization. *Annual review of physical chemistry*, 47(1):171–204, 1996.
- [DFG<sup>+</sup>21] Swarup Dey, Chunhai Fan, Kurt V Gothelf, Jiang Li, Chenxiang Lin, Longfei Liu, Na Liu, Minke AD Nijenhuis, Barbara Saccà, Friedrich C Simmel, et al. DNA origami. *Nature Reviews Methods Primers*, 1(1):13, 2021.
- [dLMA<sup>+</sup>20] Elisa de Llano, Haichao Miao, Yasaman Ahmadi, Amanda J Wilson, Morgan Beeby, Ivan Viola, and Ivan Barisic. Adenita: interactive 3D modelling and visualization of DNA nanostructures. *Nucleic acids research*, 48(15):8269–8275, 2020.
- [DLS20] David Doty, Benjamin L Lee, and Tristan Stérin. scadnano: a browser-based, scriptable tool for designing DNA nanostructures. *arXiv preprint arXiv:2005.11841*, 2020.

- [DM04] Zhaoxiang Deng and Chengde Mao. Molecular lithography with DNA nanostructures. *Angewandte Chemie*, 116(31):4160–4162, 2004.
- [DMT<sup>+</sup>09] Shawn M Douglas, Adam H Marblestone, Surat Teerapittayanon, Alejandro Vazquez, George M Church, and William M Shih. Rapid prototyping of 3D DNA-origami shapes with cadnano. *Nucleic acids research*, 37(15):5001–5006, 2009.
- [DOL<sup>+</sup>13] Jonathan PK Doye, Thomas E Ouldridge, Ard A Louis, Flavio Romano, Petr Šulc, Christian Matek, Benedict EK Snodin, Lorenzo Rovigatti, John S Schreck, Ryan M Harrison, et al. Coarse-graining DNA for simulations of DNA nanotechnology. *Physical Chemistry Chemical Physics*, 15(47):20395–20414, 2013.
- [Eih] Matthias Eiholzer. Mathru: Mathematics library written in rust.
- [Eva14] Constantine Glen Evans. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*. PhD thesis, California Institute of Technology, 2014.
- [FG53] Rosalind E Franklin and Raymond G Gosling. Molecular configuration in sodium thymonucleate. *Nature*, 171:740–741, 1953.
- [FPNP<sup>+</sup>22] Daniel Fu, Raghu Pradeep Narayanan, Abhay Prasad, Fei Zhang, Dewight Williams, John S Schreck, Hao Yan, and John Reif. Automated design of 3D DNA origami with non-rasterized 2D curvature. *Science Advances*, 8(51):eade4455, 2022.
- [FS93] Tsu Ju Fu and Nadrian C Seeman. DNA double-crossover molecules. *Biochemistry*, 32(13):3211–3220, 1993.
- [GA14] Cody W Geary and Ebbe Sloth Andersen. Design principles for single-stranded RNA origami structures. In *International Workshop on DNA-Based Computers*, pages 1–19. Springer, 2014.
- [GDS<sup>+</sup>21] Martin Glaser, Sourav Deb, Florian Seier, Amay Agrawal, Tim Liedl, Shawn Douglas, Manish K Gupta, and David M Smith. The art of designing DNA nanostructures with CAD software. *Molecules*, 26(8):2287, 2021.
- [GLB<sup>+</sup>08] Andreas R Gruber, Ronny Lorenz, Stephan H Bernhart, Richard Neuböck, and Ivo L Hofacker. The vienna RNA websuite. *Nucleic acids research*, 36(suppl\_2):W70–W74, 2008.
- [Gri28] Fred Griffith. The significance of pneumococcal types. *Epidemiology & Infection*, 27(2):113–159, 1928.
- [GWND15] Thomas Gerling, Klaus F Wagenbauer, Andrea M Neuner, and Hendrik Dietz. Dynamic DNA devices and assemblies formed by shape-complementary, non-base pairing 3D components. *Science*, 347(6229):1446–1452, 2015.
- [HKJ<sup>+</sup>21] Chao-Min Huang, Anjelica Kucinic, Joshua A Johnson, Hai-Jun Su, and Carlos E Castro. Integrated computer-aided engineering and design for DNA assemblies. *Nature Materials*, 20(9):1264–1271, 2021.

- [Hol64] Robin Holliday. A mechanism for gene conversion in fungi. *Genetics Research*, 5(2):282–304, 1964.
- [HPN<sup>+</sup>11] Dongran Han, Suchetan Pal, Jeanette Nangreave, Zhengtao Deng, Yan Liu, and Hao Yan. DNA origami with complex curvatures in three-dimensional space. *Science*, 332(6027):342–346, 2011.
- [HPY<sup>+</sup>13] Dongran Han, Suchetan Pal, Yang Yang, Shuoxing Jiang, Jeanette Nangreave, Yan Liu, and Hao Yan. DNA gridiron nanostructures based on four-arm junctions. *Science*, 339(6126):1412–1415, 2013.
- [HYS<sup>+</sup>08] Yu He, Tao Ye, Min Su, Chuan Zhang, Alexander E Ribbe, Wen Jiang, and Chengde Mao. Hierarchical self-assembly of DNA into symmetric supramolecular polyhedra. *Nature*, 452(7184):198–201, 2008.
- [IKJ<sup>+</sup>14] Ryosuke Iinuma, Yonggang Ke, Ralf Jungmann, Thomas Schlichthaerle, Johannes B Woehrstein, and Peng Yin. Polyhedra self-assembled from DNA tripods and characterized with 3D DNA-PAINT. *science*, 344(6179):65–69, 2014.
- [JHES07] Prashant K Jain, Wenyu Huang, and Mostafa A El-Sayed. On the universal scaling behavior of the distance decay of plasmon coupling in metal nanoparticle pairs: a plasmon ruler equation. *Nano Letters*, 7(7):2080–2088, 2007.
- [JSZ<sup>+</sup>19] Hyungmin Jun, Tyson R Shepherd, Kaiming Zhang, William P Bricker, Shanshan Li, Wah Chiu, and Mark Bathe. Automated sequence design of 3D polyhedral wireframe DNA origami with honeycomb edges. *ACS nano*, 13(2):2083–2093, 2019.
- [JWP<sup>+</sup>21] Hyungmin Jun, Xiao Wang, Molly F Parsons, William P Bricker, Torsten John, Shanshan Li, Steve Jackson, Wah Chiu, and Mark Bathe. Rapid prototyping of arbitrary 2D and 3D wireframe DNA origami. *Nucleic acids research*, 49(18):10265–10274, 2021.
- [JZS<sup>+</sup>19] Hyungmin Jun, Fei Zhang, Tyson Shepherd, Sakul Ratanalert, Xiaodong Qi, Hao Yan, and Mark Bathe. Autonomously designed free-form 2D DNA origami. *Science advances*, 5(1):eaav0655, 2019.
- [KBV<sup>+</sup>12] Yonggang Ke, Gaëtan Bellot, Niels V Voigt, Elena Fradkov, and William M Shih. Two design strategies for enhancement of multilayer–DNA–origami folding: underwinding for specific intercalator rescue and staple-break positioning. *Chemical Science*, 3(8):2587–2597, 2012.
- [KKDB12] Do-Nyun Kim, Fabian Kilchherr, Hendrik Dietz, and Mark Bathe. Quantitative prediction of 3D solution shape and flexibility of nucleic acid nanostructures. *Nucleic acids research*, 40(7):2862–2868, 2012.
- [KLM<sup>+</sup>18] Enzo Kopperger, Jonathan List, Sushi Madhira, Florian Rothfischer, Don C Lamb, and Friedrich C Simmel. A self-assembled nanoscale robotic arm controlled by electric fields. *Science*, 359(6373):296–301, 2018.

- [KMS83] Neville R Kallenbach, Rong-Ine Ma, and Nadrian C Seeman. An immobile nucleic acid junction constructed from oligonucleotides. *Nature*, 305(5937):829–831, 1983.
- [Kor77] Roger D Kornberg. Structure of chromatin. *Annual review of biochemistry*, 46(1):931–954, 1977.
- [KOSY12] Yonggang Ke, Luvena L Ong, William M Shih, and Peng Yin. Three-dimensional structures self-assembled from DNA bricks. *science*, 338(6111):1177–1183, 2012.
- [KSF<sup>+</sup>12] Anton Kuzyk, Robert Schreiber, Zhiyuan Fan, Günther Pardatscher, Eva-Maria Roller, Alexander Högele, Friedrich C Simmel, Alexander O Govorov, and Tim Liedl. DNA-based self-assembly of chiral plasmonic nanostructures with tailored optical response. *Nature*, 483(7389):311–314, 2012.
- [LCP<sup>+</sup>99] Colin J Loweth, W Brett Caldwell, Xiaogang Peng, A Paul Alivisatos, and Peter G Schultz. DNA-based assembly of gold nanocrystals. *Angewandte Chemie International Edition*, 38(12):1808–1812, 1999.
- [LMD] Nicolas Levy, Pierre-Étienne Meunier, and Woods Damien. Codenano, a tool for designing DNA nanostructures.
- [LMR<sup>+</sup>97] Karolin Luger, Armin W Mäder, Robin K Richmond, David F Sargent, and Timothy J Richmond. Crystal structure of the nucleosome core particle at 2.8 Å resolution. *Nature*, 389(6648):251–260, 1997.
- [LS21] Nicolas Levy and Nicolas Schabanel. ENSnano: a 3D modeling software for DNA nanostructures. In *DNA27-27th International Conference on DNA Computing and Molecular Programming*, 2021.
- [LYJ<sup>+</sup>14] Wei Li, Yang Yang, Shuoxing Jiang, Hao Yan, and Yan Liu. Controlled nucleation and growth of DNA tile arrays within prescribed DNA origami frames and their dynamics. *Journal of the American Chemical Society*, 136(10):3724–3727, 2014.
- [MA20] Christopher Maffeo and Aleksei Aksimentiev. MrDNA: a multi-resolution model for predicting the structure and dynamics of DNA systems. *Nucleic acids research*, 48(9):5135–5146, 2020.
- [MAM<sup>+</sup>22] A Mills, N Aissaoui, D Maurel, J Elezgaray, F Morvan, JJ Vasseur, E Margeat, RB Quast, J Lai Kee-Him, N Saint, et al. A modular spring-loaded actuator for mechanical activation of membrane proteins. *Nature Communications*, 13(1):3182, 2022.
- [MH02] John C Mason and David C Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.
- [MKS<sup>+</sup>22] Alba Monferrer, Jessica A Kretzmann, Christian Sigl, Pia Sapelza, Anna Liedl, Barbara Wittmann, and Hendrik Dietz. Broad-spectrum virus trapping with heparan sulfate-modified DNA origami shells. *ACS nano*, 2022.
- [Mor99] Michael E Mortenson. *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.



- [MVC<sup>+</sup>17] AM Mohammed, L Velazquez, A Chisenhall, D Schiffels, DK Fygenson, and R Schulman. Self-assembly of precisely defined DNA nanotube superstructures using DNA origami seeds. *Nanoscale*, 9(2):522–526, 2017.
- [MZSC15] Alexander E Marras, Lifeng Zhou, Hai-Jun Su, and Carlos E Castro. Programmable motion of DNA origami mechanisms. *Proceedings of the National Academy of Sciences*, 112(3):713–718, 2015.
- [OHY<sup>+</sup>17] Luvena L Ong, Nikita Hanikel, Omar K Yaghi, Casey Grun, Maximilian T Strauss, Patrick Bron, Josephine Lai-Kee-Him, Florian Schueder, Bei Wang, Pengfei Wang, et al. Programmable self-assembly of three-dimensional nanostructures from 10,000 unique components. *Nature*, 552(7683):72–77, 2017.
- [PD79] H Potter and D Dressler. DNA recombination: in vivo and in vitro studies. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 43, pages 969–985. Cold Spring Harbor Laboratory Press, 1979.
- [PKZ<sup>+</sup>14] Keyao Pan, Do-Nyun Kim, Fei Zhang, Matthew R Adendorff, Hao Yan, and Mark Bathe. Lattice-free prediction of three-dimensional structure of programmed DNA assemblies. *Nature communications*, 5(1):5578, 2014.
- [PO07] Richard T Pomerantz and Mike O’Donnell. Replisome mechanics: insights into a twin DNA polymerase machine. *Trends in microbiology*, 15(4):156–164, 2007.
- [Por91] Dietmar Porschke. Persistence length and bending dynamics of DNA from electrooptical measurements at high salt concentrations. *Biophysical chemistry*, 40(2):169–179, 1991.
- [QWB11] Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *nature*, 475(7356):368–372, 2011.
- [Riv06] TJ Rivlin. The Lebesgue constants for polynomial interpolation. In *Functional Analysis and its Applications: International Conference, Madras, 1973*, pages 422–437. Springer, 2006.
- [Rot96] Paul WK Rothmund. A DNA and restriction enzyme implementation of Turing machines. 1996.
- [Rot06] Paul WK Rothmund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006.
- [RPW04] Paul W K Rothmund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS biology*, 2(12):e424, 2004.
- [SAA<sup>+</sup>11] Daniele N Selmi, Roslin J Adamson, Helen Attrill, Alan D Goddard, Robert JC Gilbert, Anthony Watts, and Andrew J Turberfield. DNA-templated protein arrays for single-molecule imaging. *Nano letters*, 11(2):657–660, 2011.
- [SBKK11] Sunaina Surana, Jaffar M Bhat, Sandhya P Koushika, and Yamuna Krishnan. An autonomous DNA nanomachine maps spatiotemporal pH changes in a multicellular living organism. *Nature communications*, 2(1):340, 2011.

- [See82] Nadrian C Seeman. Nucleic acid junctions and lattices. *Journal of theoretical biology*, 99(2):237–247, 1982.
- [SL65] Carl Schildkraut and Shneior Lifson. Dependence of the melting temperature of DNA on salt concentration. *Biopolymers: Original Research on Biomolecules*, 3(2):195–208, 1965.
- [SM] Josh Stone and Niko Matsakis. Rayon: A data parallelism library for Rust.
- [SMD19] Fabian Schneider, Natalie Möritz, and Hendrik Dietz. The sequence of events during folding of a DNA origami. *Science advances*, 5(5):eaaw1412, 2019.
- [SS17] Nadrian C Seeman and Hanadi F Sleiman. DNA nanotechnology. *Nature Reviews Materials*, 3(1):1–23, 2017.
- [THS<sup>+</sup>14] Vivek V Thacker, Lars O Herrmann, Daniel O Sigle, Tao Zhang, Tim Liedl, Jeremy J Baumberg, and Ulrich F Keyser. DNA origami based assembly of gold nanoparticle dimers for surface-enhanced raman scattering. *Nature communications*, 5(1):3448, 2014.
- [TLJ<sup>+</sup>17] Anupama J Thubagere, Wei Li, Robert F Johnson, Zibo Chen, Shayan Doroudi, Yae Lim Lee, Gregory Izatt, Sarah Wittman, Niranjan Srinivas, Damien Woods, et al. A cargo-sorting DNA robot. *Science*, 357(6356):eaan6558, 2017.
- [TPQ17a] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns. *Nature*, 552(7683):67–71, 2017.
- [TPQ17b] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Programmable disorder in random DNA tilings. *Nature nanotechnology*, 12(3):251–259, 2017.
- [Tre19] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.
- [VMS<sup>+</sup>20] Rémi Veneziano, Tyson J Moyer, Matthew B Stone, Eike-Christian Wamhoff, Benjamin J Read, Sayak Mukherjee, Tyson R Shepherd, Jayajit Das, William R Schief, Darrell J Irvine, et al. Role of nanoscale antigen organization on B-cell activation probed using DNA origami. *Nature nanotechnology*, 15(8):716–723, 2020.
- [VZGG13] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.
- [Wan61] Hao Wang. Proving theorems by pattern recognition—ii. *Bell system technical journal*, 40(1):1–41, 1961.
- [WC53] James D Watson and Francis HC Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.
- [WDM<sup>+</sup>19] Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature*, 567(7748):366–372, 2019.

- [WDT<sup>+</sup>80] Richard Wing, Horace Drew, Tsunehiro Takano, Chris Broka, Shoji Tanaka, Keiichi Itakura, and Richard E Dickerson. Crystal structure analysis of a complete turn of B-DNA. *Nature*, 287(5784):755–758, 1980.
- [WES<sup>+</sup>17] Klaus F Wagenbauer, Floris AS Engelhardt, Evi Stahl, Vera K Hechtel, Pierre Stömmer, Fabian Seebacher, Letizia Meregalli, Philip Ketterer, Thomas Gerling, and Hendrik Dietz. How we make DNA origami. *ChemBioChem*, 18(19):1873–1885, 2017.
- [Whi13] David Whitford. *Proteins: structure and function*. John Wiley & Sons, 2013.
- [Win98] Erik Winfree. *Algorithmic self-assembly of DNA*. California Institute of Technology, 1998.
- [WLL<sup>+</sup>09] Sean Williams, Kyle Lund, Chenxiang Lin, Peter Wonka, Stuart Lindsay, and Hao Yan. Tiamat: a three-dimensional editing tool for complex DNA structures. In *DNA Computing: 14th International Meeting on DNA Computing, DNA 14, Prague, Czech Republic, June 2-9, 2008. Revised Selected Papers 14*, pages 90–101. Springer, 2009.
- [WLWS98] Erik Winfree, Furong Liu, Lisa A Wenzler, and Nadrian C Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544, 1998.
- [WME<sup>+</sup>22] Christopher M Wintersinger, Dionis Minev, Anastasia Ershova, Hiroshi M Sasaki, Gokul Gowri, Jonathan F Berengut, F Eduardo Corea-Dilbert, Peng Yin, and William M Shih. Multi-micron crisscross structures grown from DNA-origami slats. *Nature Nanotechnology*, pages 1–9, 2022.
- [WR11] Sungwook Woo and Paul WK Rothemund. Programmable molecular recognition based on the geometry of DNA nanostructures. *Nature chemistry*, 3(8):620–627, 2011.
- [WSD17] Klaus F Wagenbauer, Christian Sigl, and Hendrik Dietz. Gigadalton-scale shape-programmable DNA assemblies. *Nature*, 552(7683):78–83, 2017.
- [YHS<sup>+</sup>08] Peng Yin, Rizal F Hariadi, Sudheer Sahu, Harry MT Choi, Sung Ha Park, Thomas H LaBean, and John H Reif. Programming DNA tube circumferences. *Science*, 321(5890):824–826, 2008.
- [ZJL<sup>+</sup>14] Qian Zhang, Qiao Jiang, Na Li, Luru Dai, Qing Liu, Linlin Song, Jinye Wang, Yaqian Li, Jie Tian, Baoquan Ding, et al. DNA origami as an in vivo drug delivery vehicle for cancer therapy. *ACS nano*, 8(7):6633–6643, 2014.
- [ZNLY12] Fei Zhang, Jeanette Nangreave, Yan Liu, and Hao Yan. Reconfigurable DNA origami to generate quasifractal patterns. *Nano letters*, 12(6):3290–3295, 2012.
- [ZSB<sup>+</sup>11] Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1):170–173, 2011.

- [ZZX<sup>+</sup>20] Hanliang Zhu, Haoqing Zhang, Ying Xu, Soňa Laššáková, Marie Korabečná, and Pavel Neuzil. PCR past, present and future. *Biotechniques*, 69(4):317–325, 2020.



# Index

- Angular velocity, *see* Spin
- aTAM, **10**
- Bézier curves, **87**
  - Drawing of  $\sim$  in ENSnano, 89–90
  - Piecewise  $\sim$ , **88**
- Chebyshev
  - Interpolant, **68**
    - As a sum of Chebyshev Polynomials, 68
  - Interpolation, **72**, 76, 85
  - Points, **68**
  - Polynomial, **67**
    - Evaluation of  $\sim$ , **70**
- Crossover, **18**
- Crossover suggestions in ENSnano, 60
- Curvature, **78**
- Curvilinear Abscissa, **75**, 76
  - Inverse  $\sim$ , **75**, 76, 108
- DNA bricks, *see* Single-stranded tiles
- DNA origami, **17**
  - Applications, 10
  - Design technique, 20
- Domain (of a DNA strand), **6**
- Double-crossover tiles, 10
- ENSnano interface, 51
  - Getting started, 53
  - Tool panel, 52
- Experimental realization
  - Double-sphere origami, 132
  - Looped square origami, 92–93
  - Möbius torus, 124
  - Rocket origami, 63
  - Toruses with 2-6 half rotations, 128
- Fixed, **36**
- Moving, **75**
  - Construction of a Continuous  $\sim$ , 85–86
  - Construction of a Discrete  $\sim$ , 78–80
- Gel electrophoresis, **5**
- Helicity (of a DNA double helix), 37
- Holliday junction, 6
- Inclination (of a DNA double helix), 38
- Lattice associated to a grid, **36**, 85
- Lebesgue Constant, **68**
- Minor Groove Angle (of a DNA double helix), 37
- Molecular computing, 9
- P-stick Model
  - For curved helices, **75**
  - For straight helices, **36**, 75
  - Geometric Parameters, **37**
    - Adjustment of the  $\sim$ , 77
- PCR, **6**
- Rigid body, **44**
- Rise (of a DNA double helix), 37
- Single-stranded tiles, 10
- Spin, 45
- Stacking Interactions (in a DNA double helix), 78
- Translated Curve, **85**
- Twisted Revolution Surface, **108**

Frame





# Index of Citations

- [AT12], 78  
[Adl94], 9, 10  
[Adl98], 9  
[ADN<sup>+</sup>08], 26
- [Bar01], 44  
[BI92], 17  
[BMG<sup>+</sup>15], 22, 26, 27  
[BMP<sup>+</sup>22], 41  
[BDLS96], 10  
[BSV<sup>+</sup>20], 116
- [CKK<sup>+</sup>11], 18  
[CS91], 9  
[CQ18], 10  
[LMD], 27  
[CTP<sup>+</sup>08], 78
- [DDB<sup>+</sup>15], 134  
[dLMA<sup>+</sup>20], 26  
[DM04], 13  
[DFG<sup>+</sup>21], 17  
[DBG<sup>+</sup>16], 13  
[DDS09], 21, 26, 29, 90  
[DLS20], 27, 35  
[DCS07], 15  
[DMT<sup>+</sup>09], 26, 90  
[DDL<sup>+</sup>09], 21  
[DBC12], 22  
[DOL<sup>+</sup>13], 27  
[DDO<sup>+</sup>15], 134  
[DF96], 13
- [Eva14], 10
- [FG53], 3  
[FS93], 10  
[FPNP<sup>+</sup>22], 27, 101, 108
- [GA14], 37, 38
- [GWND15], 10, 22, 36  
[GDS<sup>+</sup>21], 26  
[Gri28], 3  
[GLB<sup>+</sup>08], 27
- [HPN<sup>+</sup>11], 21, 27, 29, 101, 107, 108  
[HPY<sup>+</sup>13], 22  
[HYS<sup>+</sup>08], 22  
[Hol64], 6  
[HKJ<sup>+</sup>21], 27
- [IKJ<sup>+</sup>14], 29
- [JHES07], 13  
[JSZ<sup>+</sup>19], 27  
[JZS<sup>+</sup>19], 27  
[JWP<sup>+</sup>21], 27
- [KMS83], 9  
[KOSY12], 10, 13, 61  
[KBV<sup>+</sup>12], 134  
[KKDB12], 27  
[KLM<sup>+</sup>18], 22  
[Kor77], 81  
[KSF<sup>+</sup>12], 13
- [LS21], 51  
[LYJ<sup>+</sup>14], 10  
[LCP<sup>+</sup>99], 13  
[LMR<sup>+</sup>97], 81, 84
- [MA20], 27  
[MZSC15], 22  
[MH02], 67–70  
[Eih], 115  
[MAM<sup>+</sup>22], 10, 36  
[MVC<sup>+</sup>17], 10  
[MKS<sup>+</sup>22], 15  
[Mor99], 87

[OHY<sup>+</sup>17], 13

[PKZ<sup>+</sup>14], 29  
[PO07], 6  
[Por91], 4  
[PD79], 8

[QWB11], 10

[SM], 72  
[Riv06], 68  
[Rot96], 10  
[RPW04], 10  
[Rot06], 10, 17, 18, 20

[SL65], 5  
[SMD19], 134  
[See82], 8  
[SS17], 10  
[SAA<sup>+</sup>11], 15  
[SBKK11], 15

[THS<sup>+</sup>14], 13  
[TLJ<sup>+</sup>17], 10, 29, 36, 48, 57, 108  
[TPQ17a], 10  
[Tre19], 67, 68

[VMS<sup>+</sup>20], 15  
[VZGG13], 70

[WSD17], 10  
[WES<sup>+</sup>17], 22  
[Wan61], 10  
[WC53], 3, 6  
[Whi13], 13  
[WLL<sup>+</sup>09], 26  
[Win98], 10  
[WLWS98], 10  
[WDT<sup>+</sup>80], 4, 5  
[WME<sup>+</sup>22], 10  
[WR11], 20, 26, 38, 47, 60, 61, 80, 108  
[WDM<sup>+</sup>19], 10, 61

[YHS<sup>+</sup>08], 61

[ZSB<sup>+</sup>11], 27  
[ZNLY12], 22  
[ZJL<sup>+</sup>14], 15  
[ZZX<sup>+</sup>20], 6