Hierarchical Assembly of DNA Nanostructures for Signal Transmission

Andres David Serrano Paladines

Undergraduate Thesis, Fall 2017

Bachelor of Science in Mechanical Engineering

The Ohio State University

Department of Mechanical and Aerospace Engineering

Nanoengineering and Biodesign Laboratory

Thesis Defense Committee:

Dr. Carlos Castro, Department of Mechanical and Aerospace Engineering

Dr. Jonathan Song, Department of Mechanical and Aerospace Engineering

Mentor:

Dr. Alexander Marras, Department of Mechanical and Aerospace Engineering

Presented in partial fulfillment of the requirements for graduation with Honors Research Distinction at The Ohio State University

Abstract

In recent years scaffolded DNA origami has emerged as a novel technique for the construction of programmable nanostructures via molecular self-assembly. This technology provides unprecedented control over geometry and mechanical properties. These structures have demonstrated potential for a range of biomedical applications such as drug delivery, force measurement, and biomarker detection. Recent advancements have focused on the design of dynamic structures that can be triggered by DNA or other biological or environmental inputs to undergo actuated motion of the structure into different conformations. This work aims to exand on this foundation by developing of material systems where local conformational changes can be physically communicated to other parts of the material through propagated motion. We have designed a dynamic DNA nanostructure that can be assembled into arrays that can reach length scales ~10-100 times larger than the individual structure and can propagate conformational changes across the arrays. DNA strands specific to one end of the array initiate motion for the "trigger" structure at that end, which in turn propagates motion to subsequent structures in a sequential manner. This propagated motion is designed to transmit a signal across large distances. In the future, the ability to transmit a signal across micron-scale distances could lead to customizable molecular transport systems, programmable circuits, and long-range directional communication in biological environments.

Dedication

To my parents and siblings, for supporting and advising me through all the endeavors I have set out for myself since taking the path of engineering.

Acknowledgements

Firstly, I would like to thank my advisor, Dr. Carlos Castro, for giving me the invaluable opportunity to work in his Nanoengineering and Biodesign Laboratory. His vision and approach to academia and research have left a lasting impression on me by molding my perspective of what it constitutes to be an engineer. The research environment that he has cultivated in his laboratory will always be a source of admiration, and his mentorship and support serve for constant motivation.

I would also like to thank Dr. Alexander Marras and Patrick Halley, for accepting me as their mentee at different points of my research career and challenging me to become a better scientist. They are exceptional engineers and good friends that have set benchmarks I aspire to reach and pass. I also must thank Dr. Jonathan Song, not only for accepting to be in my thesis committee, but also for being a dedicated professor that motivated me to expand my vision of engineering. The class atmosphere that he promotes led to many stimulating discussions that have shaped my research interests.

My fellow lab partners have filled not only my research career but also my college experience with unforgettable moments. I could never have asked for a better undergraduate research team, OhioMOD 2016, who are with no doubt the best nerds in the game. I look forward to reuniting with all of you at Tommy Rudibaugh's place in North Carolina for happy hour. Amjad Chowdhury and Nick Andrioff, I'll always cherish our long-lasting conversations on what Anjelica Kucinic dubs as "boys stuff", and I commend her for being patient enough with us. Going to Cazuelas on Friday afternoons with the three of you was the best way to start the weekend. Thank you, Jenny Le, for also being a great mentor, friend, anime enthusiast, lifesaver, brutally honest, the list continues. I could never have spent so much time in the fourth-floor office if it wasn't for your company and that of Melika Shahhosseini. Everyone else in NBL, I have enjoyed working with all of you and I appreciate the diligent but entertaining atmosphere fomented by you.

Finally, I thank my family and friends, both here in Columbus and in Ecuador for making my college journey far less stressful through all the moments shared together. Alex Schilling, thank you for listening to all my ramblings as my roommate for these last two years.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Biological Nanotechnology

Nanotechnology encompasses the design, manufacturing, and implementation of materials at the scale of one to hundreds of nanometers (nm). Currently, nanoscale materials play pivotal roles in the development of a wide array of technologies such as chemical catalysts, cancer screening, and electronics [1]. The potential applications of nanotechnology clearly have far-reaching impact on different engineering and science fields. Arguably, one of the most exciting prospects of nanotechnology is the ability to work on the scale of biology for medical applications or in probing or controlling molecular and cellular systems.

Biological nanotechnology integrates disciplines such as medicine, engineering, chemistry, among others, for the interaction of nanomaterials with biological systems. Current research focuses on the development of organic and inorganic nanodevices for drug delivery, therapeutics, diagnostics and imaging [2, 3, 4]. One significant advantage of engineering such nanodevices is their versatility to exploit the functionality of biological systems to mimic, manipulate, and measure biological processes [5, 6, 7]. These devices rely on the ability to design nanostructures with precisely controlled geometry, mechanical properties, and interaction capabilities (e.g. binding to biomolecules). Structural DNA nanotechnology, the major focus behind this work, gives the opportunity to design nanoscale machines with unparalleled control over complex nanoscale geometry and mechanical and dynamic properties that can perform multiple tasks.

1.2 DNA Structure

Deoxyribonucleic acid (DNA) is well known as the genetic material of all living organisms, and it functions to store and express information. This information is encoded in the DNA sequence, which is the ordering of organic molecules known as nucleotides within a DNA strand. As seen in Figure 1, every nucleotide contains in their structure one of the following nitrogenous bases:

Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). As seen if in Figure 1, these bases can bind to each other through hydrogen bonds according to Watson-Crick base pairing rules; adenine with thymine (A-T) and cytosine with guanine (C-G) [8].

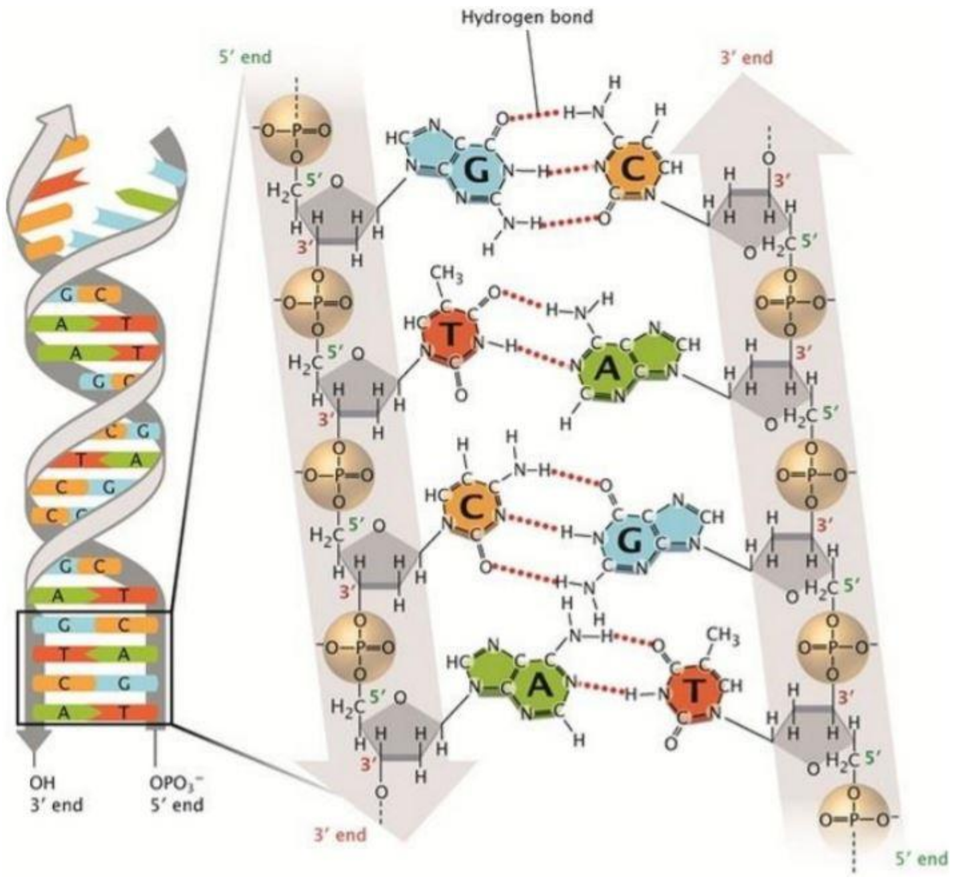


Figure 1: B-DNA, the most common double helical structure. Two hydrogen bonds connect A to T while three hydrogen bonds connect C to G. The sugar-phosphate backbones run anti-parallel to each other. [9]

One of the key aspects of DNA deduced by James Watson and Francis Crick is that the complementary pairing of A to T and C to G leads to a mechanism for biological replication [8], and currently, custom DNA strands can be chemically synthesized by a number of commercial vendors by connecting nucleotides in a programmed sequence order. Following complementary base-pairing strands of DNA can be designed to be complementary, keeping in mind that two strands of single-stranded DNA (ssDNA) bind in an anti-parallel fashion (Fig. 1). For example, a strand with the sequence 5'-AGTC-3' (5' and 3' denote the ends with and without a terminal phosphate group, respectively) would be complementary to a strand with the sequence 5'-GACT-3'. These complementary strands are therefore referred to as reverse complements. The binding strength between DNA strands depends on the length of the strands; their sequence, because C-G pairs from three hydrogen bonds while A-T pairs from two; and solution conditions, specifically strand concentration and cation concentration. Cations screen the electrostatic repulsions between the negatively charged phosphate groups in the sugar-phosphate backbones.

1.3 Structural DNA Nanotechnology

To build nanoscale machines with DNA, it is also necessary to modify the linearity of the helix axis by creating branches that can be combined into larger constructs [10]. In result, branching customizable DNA molecules to form lattices in two or three dimensions is the fundamental concept behind structural DNA nanotechnology. Structural DNA nanotechnology was founded by Nadrian C. Seeman in the early 1980's, after envisioning the organization of nucleic acids into crystalline arrangements through a designed self-assembly process [10]. He proposed the production of nucleic acid sequences that form immobile junctions (Figure 2) by minimizing the sequence symmetry among Watson-Crick base pairs [11]. In turn, the construction of DNA immobile junctions implied the possibility to assemble highly specific geometrical lattices that are periodic in space. Early work on the field led to the assembly of cubes, octahedrons, and other topologies that are considered the basis for DNA-based nanomechanical devices [12].

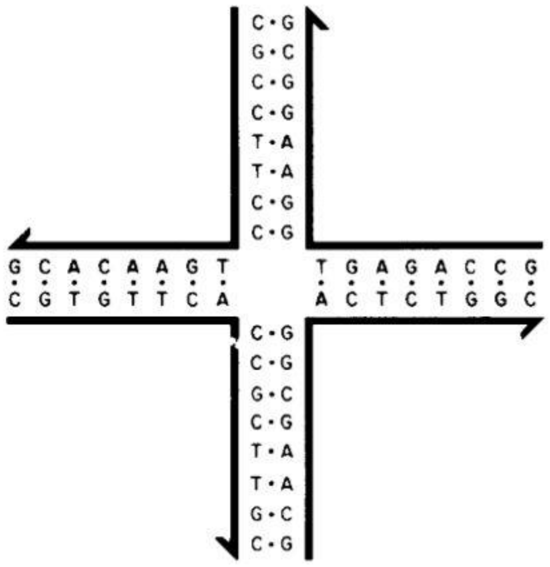


Figure 2: Sequence design of a 4-arm DNA branched immobile junction. The directions of the sugar-phosphate backbones, 5'→3', are indicated by the half-arrowheads. [11]

This immobile junction could be connected into larger structures via the use of so-called "sticky ends," which are sequences of ssDNA that extend beyond the double-stranded DNA (dsDNA) duplex to enable base-pairing to a neighboring unit or structure.

1.4 DNA Origami

Seeman's DNA immobile junction became the building block for the development of structural DNA nanotechnology. In 2006, the invention of scaffolded DNA origami by Paul Rothemund was a major advance in enabling the creation of 2D nanostructures with high complexity [13]. By combining a ~7000-8000 bases long single-stranded DNA (ssDNA) scaffold around two hundred ~15-60 bases long ssDNA strands, referred to as staples, shapes composed of DNA double helices can be assembled. The scaffold is usually derived from the M13mp18 viral genome, while the staples can be synthesized with custom sequences by a number of commercial vendors [13, 14]. Specifically, the sequences of the short staples are designed to be piecewise complementary to the sequence of the long scaffold. In result, they bind together according to Watson-Crick base pairing rules forming geometries that contain parallel double helices as shown in Figure 3. When the helical rotation of either the scaffold or staple strands in a double helix is directed towards a neighboring helix, crossovers can be made between the two that hold the helices together and give stability to the desired nanostructure [14]. The DNA origami design process can be facilitated by computer-aided-design (CAD) software.

DNA origami fabrication [14] is carried out via molecular self-assembly by mixing the scaffold strands with the staple strands that correspond to a desired structure. Typically, the staples are present in excess relative to the scaffold to allow for their correct binding while also organizing the scaffold for the binding of subsequent staples [13].

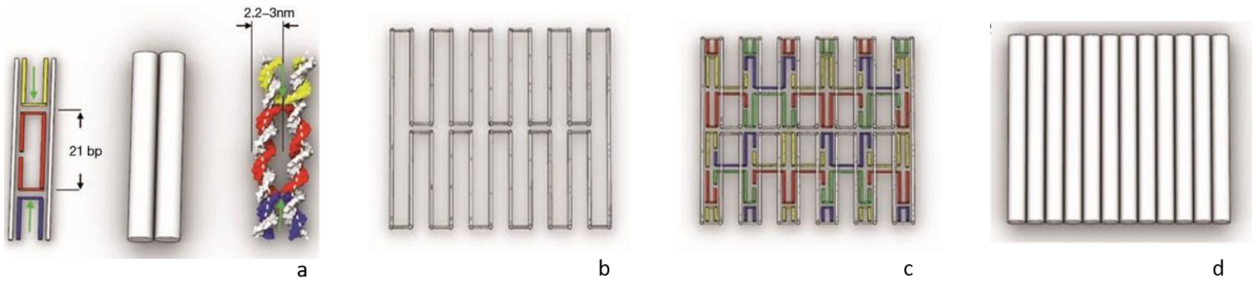


Figure 3: Design of a rectangular tile with DNA origami. a) The staples (colored strands) binding to the scaffold segments (white strands) form parallel double helices that are connected due to crossovers (green arrows). b) Scaffold blueprint for rectangular tile. c) Staples are colored differently to highlight their paths through the scaffold blueprint. d) Double helices in cylinder representation. [14]

As seen in Figure 4, Rothemund initially demonstrated the versatility of DNA origami by designing and manufacturing two-dimensional structures such as stars and smiley faces, among other shapes, that contained a high level of detail [13]. Further development of the field led to the creation of three-dimensional structures that contained twisted and curved geometries [15, 16].

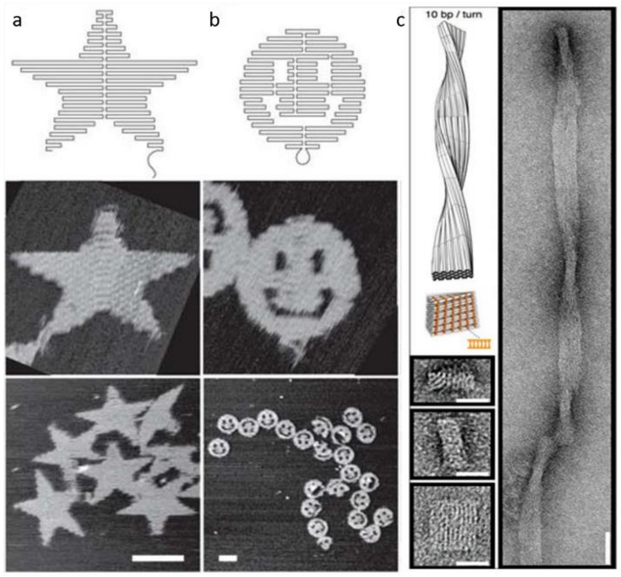


Figure 4: 2D and 3D DNA origami Structures. a, b) Scaffold blueprint and AFM images of star and smiley two-dimensional structures. Scale bars: 100 nm. [13] c) CAD design and TEM images of a 10 by 6-helix DNA bundle with 10 bp/turn average double-helical twist density. Scale bars: 50 nm. [16]

A major current direction for the field is the creation of DNA origami structures that can be programmed to change conformations in response to specific signals introduced into the local environment [17-19]. This ability to dynamically control nanostructures gives the opportunity to use them for single-molecule experiments to characterize biological processes [20, 21]. Furthermore, these nanostructures have also shown promise in cancer research for their potential as novel drug delivery carriers [22-24], having already demonstrated their viability in *in vivo* mice trials [4]. Our lab has demonstrated the use of DNA nanostructures to circumvent daunorubicin drug resistance at clinically relevant doses in a leukemia model [24] as illustrated in Figure 5. These applications for DNA origami have emerged in the last decade, achieving and exceeding the possibilities of structural DNA nanotechnology predicted by Seeman decades earlier.

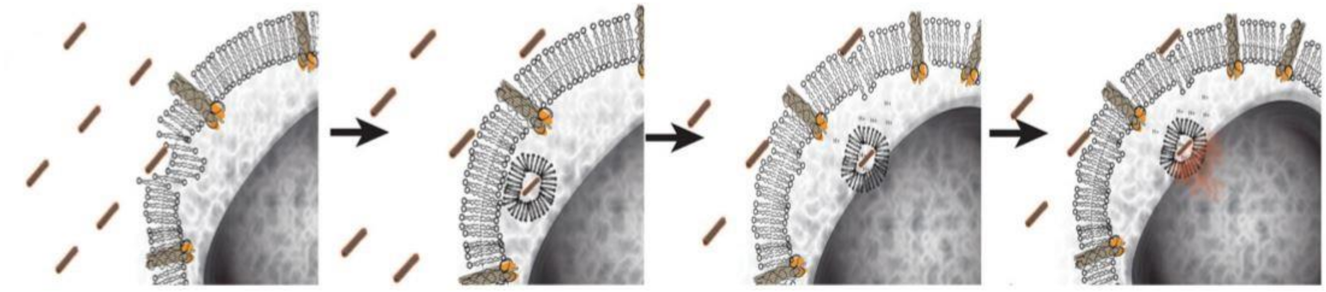


Figure 5: Daunorubicin-loaded DNA nanostructures circumvent efflux pumps expressed on a cell's surface. Once inside the cell, daunorubicin molecules are released and enter the nucleus to disrupt DNA replication and impair cellular growth. [24]

1.5 DNA Strand Displacement

This work seeks to expand the functional capabilities of dynamic DNA origami nanostructures to by enabling hierarchical DNA origami systems that can be controlled dynamically to produce long-range propagated motion. To achieve this, it is necessary to understand DNA strand displacement, a technique widely used to actuate DNA nanostructures.

Although DNA Origami has been successful in facilitating the creation of DNA nanostructures with structural complexity, the ability to actuate conformational changes requires the inclusion of some actuation mechanism. The most commonly used actuation mechanism is DNA strand displacement to allow dynamic transition between conformational states. Most often, DNA strand displacement is used to release a connection that transitions the structure from being latched in a specific conformation (e.g. a closed container) to a state with some component or components that undergo constrained fluctuation (e.g. pivoting open or closed of an arm or lid). In this process, two ssDNA strands with partial or full complementary sequences hybridize by displacing (i. e. removing) another pre-hybridized shorter ssDNA strand (Figure 6). This technique allows for molecular-scale changes to be controlled in a sequence-specific manner [25].

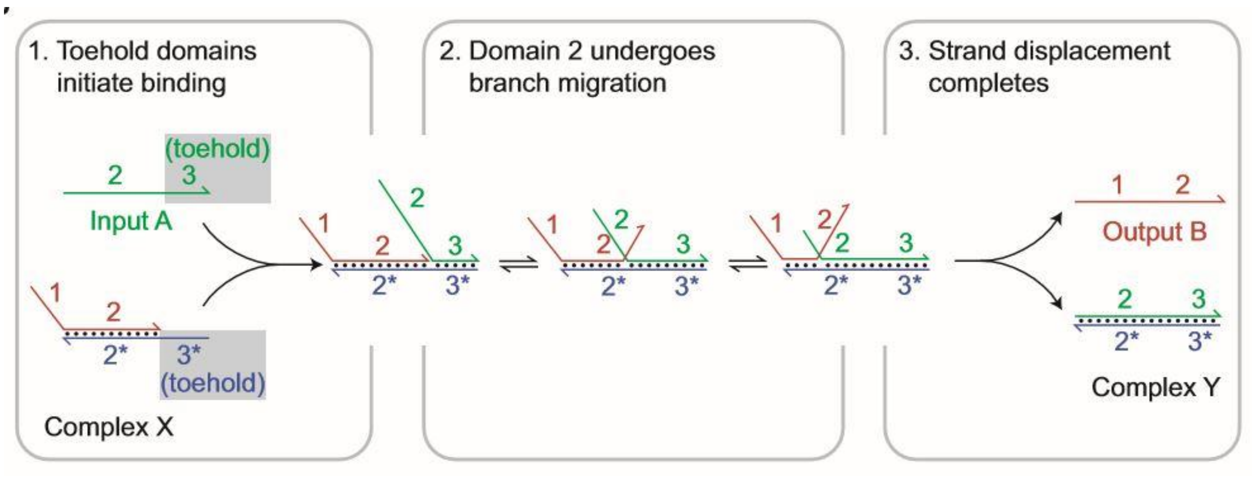

1. Toehold domains initiate binding
2. Domain 2 undergoes branch migration
3. Strand displacement completes
(toehold)
2
3
Input A
1
2
2*
3*
(toehold)
Complex X
1
2
2*
3
3*
Output B
Complex Y

Figure 6: Strand displacement reaction facilitated by "toehold" sequences. [25]

A relevant example is shown in Figure 6, where the two pre-hybridized ssDNA strands have different overall sequences, with the sequence of the red strand being only partially complementary to the sequence of the blue strand. Meanwhile, the green strand that initiates the reaction is fully complementary to the blue strand and both have “toehold” sequences (or ssDNA overhangs that extend beyond the duplex region) that start binding. The hybridization of these toeholds will align the blue and green strands while the red strand starts to be displaced. The bases from the green strand displace the bases from the red strand in a random-walk process.

DNA strand displacement also allows for conformational changes on DNA nanostructures to be reversed, as the hybridization and displacement of strands can be made in a cyclic manner [26]. The next step is to review these dynamic nanostructures and how they can be reconfigured based on their structural design.

1.6 Dynamic DNA Origami Nanostructures

Some of the fundamental practices behind Mechanical Engineering are the design, manufacturing, and operation of mechanical systems. Our lab has pioneered the translation of these practices to the nanoscale with the creation of DNA origami mechanisms and machines that parallel the design of macroscopic mechanisms [27, 28, 29, 30]. Specifically, these can be given multiple degrees of freedom by coupling rigid double helical DNA bundles with flexible ssDNA connections. Figure 7 shows a DNA Origami compliant joint can imitate a hinge with a torsional spring [27]. By increasing or decreasing the length of the ssDNA "springs", the compliant joint's angle can be increased or decreased accordingly. This change in length also leads to a change in the torsional stiffness of the compliant joint.

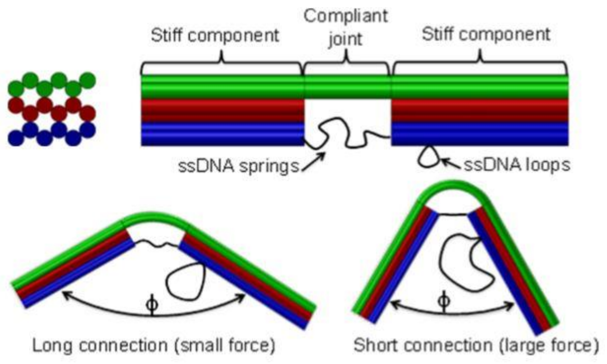


Figure 7: DNA origami compliant joint. Short ssDNA springs have a high torsional stiffness and in result apply a large force, while long ssDNA springs have a low torsional stiffness and in result apply a small force. [27]

The mechanical properties that control the behavior of rigid double helical DNA bundles coupled with flexible ssDNA connections can also be practically used for the control of dynamic nanostructures. By adopting the approach behind macroscopic mechanism design, the fabrication of joints with predefined angular or linear motion can be achieved. Coupling precisely designed double helical bundles with selectively placed ssDNA connections ensures motion in specific degrees of freedom [28]. For instance, DNA origami hinge and slider joints can be designed that only move in rotational and linear directions respectively [29]. These joints can then be combined to create mechanisms that exhibit both types of motion (Figure 8).

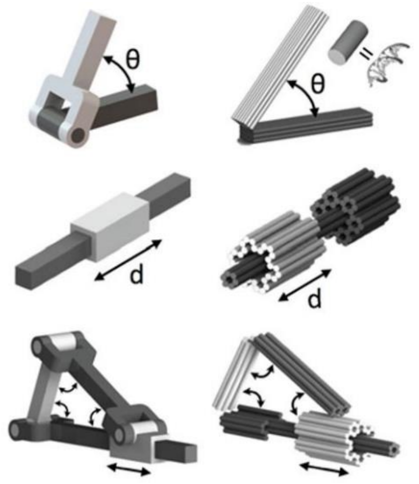


Figure 8: Macroscopic mechanisms along with their corresponding DNA origami counterparts. Joints that demonstrate angular (top) and linear (middle) motion can be integrated to achieve complex motion (bottom). [29]

DNA strand displacement, as explained in the previous section, allows for the dynamic control of DNA nanostructures through a sequence-specific process. This technique and the mechanical properties of DNA serve as powerful tools to initiate and reverse conformational changes. An example of this is the Bennett linkage mechanism shown in Figure 9 that transitions between an open frame conformation and a compact bundle conformation [29]. It is designed with several ssDNA overhangs along the length of its four links so that ssDNA inputs with complementary sequences bridge the overhangs to actuate the structure into the closed bundle configuration. The ssDNA inputs contain toehold sequences that allow for their removal via strand displacement after their binding to another set of complementary ssDNA inputs.

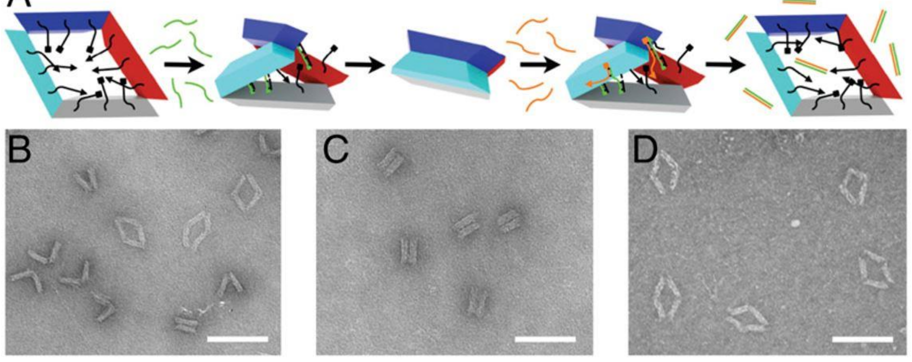


Figure 9: Reversible conformational change of DNA origami Bennett linkage. A) Actuation of the mechanism to its compact bundle shape is achieved via the addition of ssDNA inputs that bridge ssDNA overhangs. The mechanism can be reverted to its open frame shape via strand displacement after a second addition of ssDNA inputs. B-D) TEM images showing the reversible transition between open frame and compact bundle conformations. Scale bars: 100 nm. [29]

DNA origami nanostructures based on macroscopic mechanisms can also be used to design features of the energy landscapes that dictate the dynamics of these nanostructures. For example, our lab previously developed the 4-bar mechanism shown in Figure 10 that incorporates a jointed beam in a buckled configuration and a deformable, or compliant, link to form a bistable a DNA origami mechanism [30]. The jointed beam can "snap-through" between two stable positions via the bending of the compliant link. As seen in Figure 10, it's energy landscape can be characterized by analyzing the mechanical deformation of the compliant link since it has the lowest bending stiffness. The two stable positions (S1 and S2) correspond to configurations where the crank link (e.g. lower portion of the jointed beam) is either at a small angle or in a vertical position relative to the frame link. When the crank and coupler links are collinear (U), the compliant link reaches its maximum deformation and maximum energy.

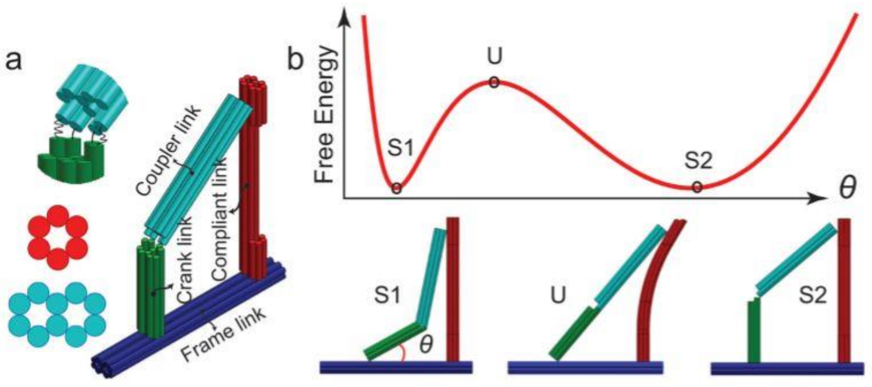


Figure 10: Design and energy landscape of DNA origami bistable mechanism. a) Schematic of the nanostructure, which is composed of four links. b) Energy landscape determined by the deformation of the compliant link, which results in two stable positions (S1 and S2) where there is no deformation and one unstable position (U) where there is maximum deformation. [30]

1.7 Signal Transmission with DNA Origami

It has been shown how DNA origami nanostructures can be actuated by using a variety of designs that reconfigure rigid double helical bundles with flexible ssDNA connections. However, all these mechanisms have been synthesized from one ssDNA scaffold and, in result, have been restricted to short-range motion. To address this limitation, it is necessary to develop DNA origami mechanisms capable of propagating motion across longer distances.

The transmission of mechanical and chemical signals at the molecular level is a predominant phenomenon in various biological processes such as receptor-mediated cell signaling or materials transport via molecular motors. The ability to reconfigure DNA origami nanostructures gives the opportunity to develop dynamic systems that can transmit signals. Initial applications with DNA origami for signal transmission consisted in two-dimensional tiles with precisely defined features that allow for the directional control of DNA walkers [31], which are nanomachines that mimic motor proteins such as kinesin (Figure 11). Further development of DNA origami tiles has led to assembly lines where DNA walkers can collect cargo [32], and the improvement of their range of motion by using microfluidic devices that introduce and remove ssDNA inputs [33].

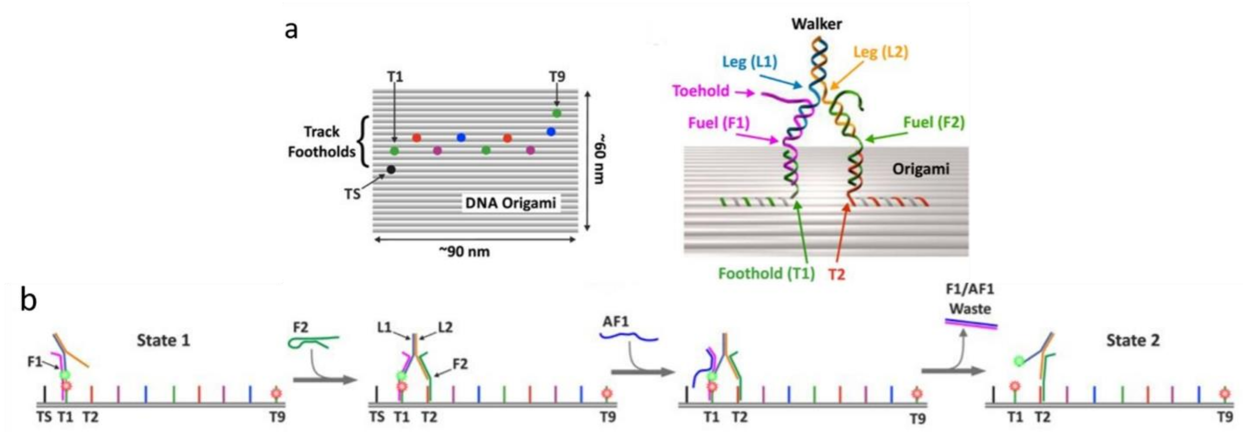


Figure 11: DNA walker design and operating principle. a) Left, DNA origami tile containing 10 foothold strands (TS and T1-T9; different colors indicate different DNA sequences). Right, each of the two legs of the walker connects to the footholds via "fuel" strands (F1 and F2). b) Motion from T1 to T2: Fuels have four sequences (F1-F4), such that each fuel attaches one specific leg to one specific foothold. Introducing antifuel strands (AF1-AF4) detach the fuel and release the leg from its foothold [33].

The DNA origami platforms explored above provide the means to transmit signals for the manipulation of biological processes. Nevertheless, these mechanisms are still only capable of propagating motion across a range of up to ~100 nanometers [31-33]. By assembling large and dynamic DNA origami structures via the polymerization of monomeric units, long-range signal transmission can be achieved. For this end, the transformation of DNA origami arrays that can be initiated at selected units and then propagated to subsequent units throughout the array has been demonstrated [34]. In another study, two nanostructures were connected by geometric complementarity and sequence-specific DNA linkages to transfer a signal throughout their combined lengths [35]. DNA hairpins were immobilized at specific points along the nanostructures, and upon the addition of an initiator strand, they polymerized to form a continuous DNA duplex. Another DNA origami mechanism highly relevant to this work was the polymerization of the pseudorotaxane filament shown in Figure 12 where rings passively slid along the filament's axis after being released via DNA strand displacement [36].

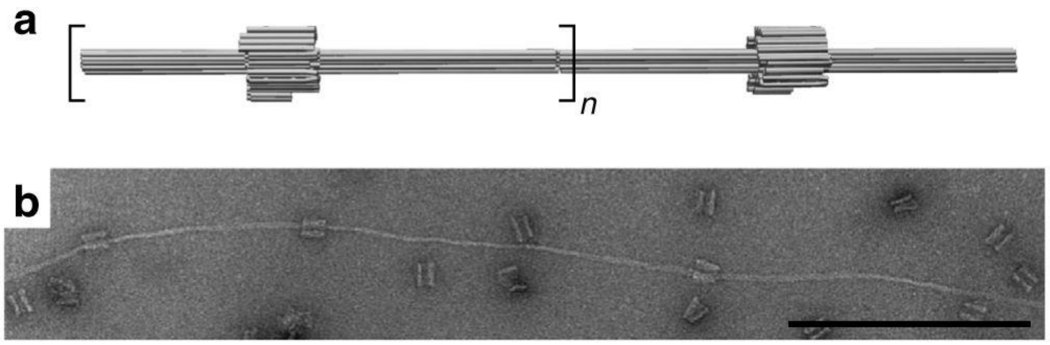


Figure 12: Pseudorotaxane filaments with multiple rings. a) CAD model of filaments composed of polymerized units with rings attached. b) TEM image of a polymeric filament with rings attached at their starting position. Scale bar: 500 nm. [35]

This review shows the progression through which structural DNA nanotechnology has evolved. From the characterization of the DNA double helix to the transmission of signals with DNA origami nanostructures, the developments presented here serve as a basis for the work of this thesis. Chapter 2 focuses on DNA origami design, manufacturing, and strategies to ensure structural stability. Chapter 3 discusses the nanostructure that is the primary focus of this work and its hierarchical assembly for long-range signal transmission. Finally, Chapter 4 provides concluding thoughts on the potential that the nanostructure presented here should have for structural DNA nanotechnology.

Chapter 2: DNA Origami Design, Manufacturing, and Stability

2.1 Design Process

DNA origami facilitates the self-assembly of higher-order molecular structures due to the programmable nature of DNA. Furthermore, parameters such as complementary geometry, internal connections, and annealing conditions influence the self-assembly of these nanostructures [13, 14, 37]. DNA origami design begins by modeling double helical DNA bundles from the hybridization of the long ssDNA scaffold with multiple short ssDNA staples. These bundles can be designed with precisely controlled cross-sectional geometries where the helices fall on two- or three-dimensional geometries.

To build a three-dimensional nanostructure, bundles may be packed onto a honeycomb lattice [38] or a square lattice [39] as seen in Figure 13. As discussed in Section 1.4, double helical bundles can be packed onto a particular lattice by being connected to neighboring bundles using crossovers. Each strand in a double helical bundle rotates by 240° about the helical axis every 7 base-pairs (bp), allowing for a cross-over placement every 7 bp to create a honeycomb cross-sectional lattice (Fig .13a, right). Meanwhile, in a square lattice a cross-over is placed every 8bp to one of four neighboring helices.

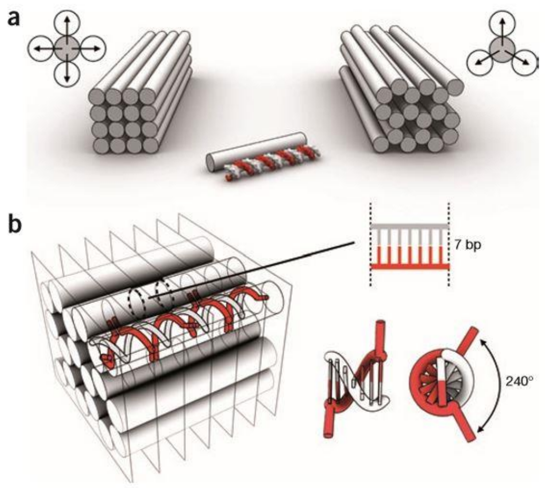


Figure 13: Packing and crossover rules of DNA origami lattices. a) Cross-sectional view of square lattice and honeycomb lattice packing. b) Crossovers in honeycomb lattice, which are allowed every 7 bp due to the 240° helix rotation. [14]

B-DNA has a natural twist density of 10.5 bp/turn, but in a square lattice its twist density is altered to 10.67 bp/turn, resulting in twisting torques that are transmitted by the crossovers throughout the entire lattice [39]. This creates a global twist deformation, but it can be eliminated by departing from the constant placement of crossovers every 8 bp to achieve twist densities closer to 10.5 bp/turn, or by creating objects with large torsional stiffness in the helix axis [39].

Once the desired lattice packing is selected, the ssDNA scaffold is routed to resemble the shape taken by the packed double helical bundles. Scaffold routing and the whole DNA origami design process is facilitated by using the open-source design software caDNAno [38]. As illustrated in Figure 14, caDNAno presents the user a two-dimensional schematic of the double helical bundles in the shape taken, and the scaffold can then be routed throughout this schematic. Next, the ssDNA staples are also routed to create the double helical bundles and to insert the majority of crossovers that serve as connections between neighboring bundles. Once the staple routing is complete, the caDNAno design can be exported for verification to canDo, a computational tool that uses the finite element method to predict the three-dimensional shape of DNA origami nanostructures [14]. Finally, if the caDNAno design meets the user's specifications, a list of staple sequences can be generated based on their complementarity to the sequence of the scaffold. These staples can then be synthesized by a number of commercial suppliers.

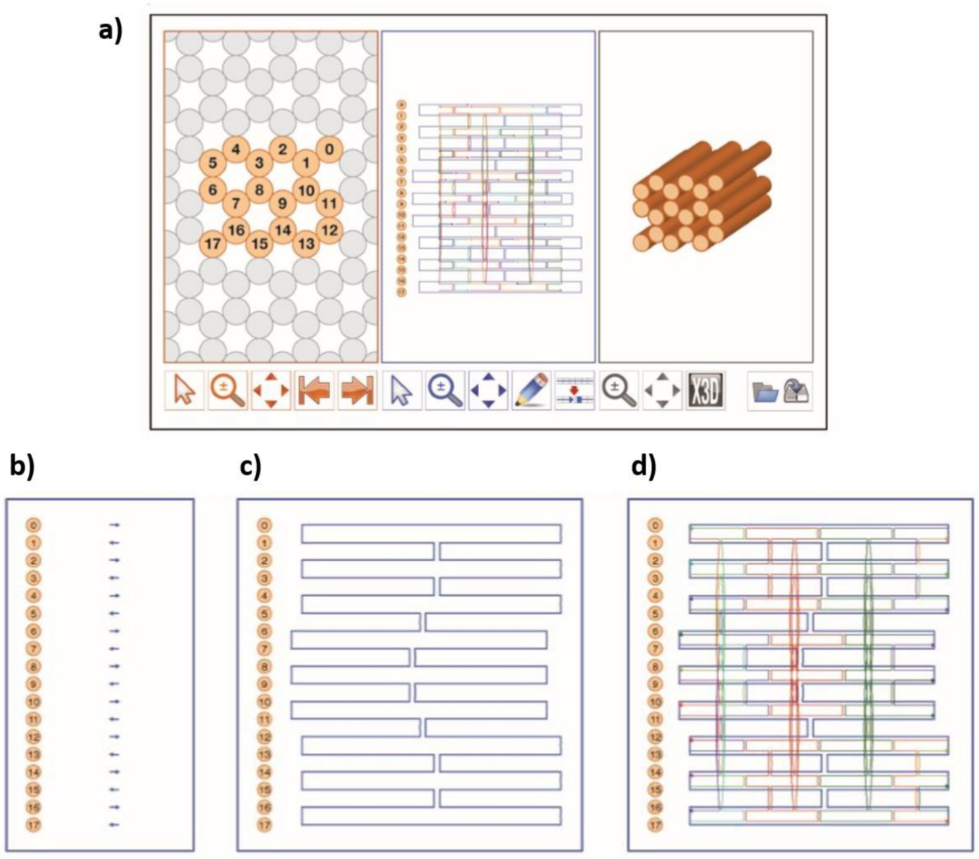


Figure 14: caDNAno interface and design process. a) caDNAno interface. Left, cross-sectional view of the desired structure based on a honeycomb lattice. Middle, 2D schematic of scaffold and staples routing. Right, 3D schematic of the desired structure. b) Middle panel snapshot during the first step of the design process. c) Scaffold routing step. d) Finalized design process after staple routing. [38]

2.2 Manufacturing Process

DNA origami nanostructures are fabricated through a self-assembly process in which an excess of short staple strands bind to the long scaffold strand when subjected to an annealing process. Specifically, there is a cooperative process in which the probability of staples folding to the scaffold increases in the presence of staples already folded [40]. The scaffold and excess staples are mixed with 1 mM EDTA, 5 mM Tris, 5 mM NaCl, and a variable concentration of $MgCl2$, typically 10-20 mM, in a solution that is subjected to an annealing thermal ramp. This annealing process starts by heating the solution to 65 °C to ensure melt interactions so all the DNA is melted into ssDNA. The solution is then cooled down over a range of temperatures typically over the course of several hours to a few days. This is done because the synthesis of different structures is dependent on the configuration of the thermal ramp. By taking this approach, it is possible to manufacture three-dimensional nanostructures with high yields within ten to hundreds of minutes [41], and direct folding pathways to program the sequence in which the components of higher-order structures are assembled [42].

Folded DNA origami nanostructures in solution are purified using agarose gel electrophoresis [14] or polyethylene glycol (PEG) purification [43]. For gel electrophoresis, the solution is mixed with a loading dye and introduced into the empty wells of a 2% agarose gel that contains 0.5x TBE, EtBr, and 11 mM MgCl2. A voltage is applied across the gel for 2-3 hours to allow the negatively charged nanostructures travel towards the negative to a positive electrode, separating them from misfolded or aggregated DNA with higher molecular weight. The results of the folding reaction are usually run on a gel along the ssDNA scaffold as a control to qualitatively assess its folding by comparing the run speed of the folded structures relative to the scaffold as illustrated in Figure 15. The gel is then placed over an ultraviolet transilluminator for imaging of the EtBr-stained DNA to excision the folded nanostructures and resuspend them in solution. For PEG purification, the solution is mixed with an equal volume of 15% PEG 8000 in a centrifuge for 25 minutes at 4 °C. PEG promotes the separation of folded nanostructures from misfolded excess DNA based on their molecular weights. The nanostructures are accumulated into a pellet while the excess DNA is suspended in a supernatant. This supernatant is then removed for the resuspension of the pellet in solution. Finally, purified DNA origami nanostructures can be structurally characterized by using a transmission electron microscope (TEM).

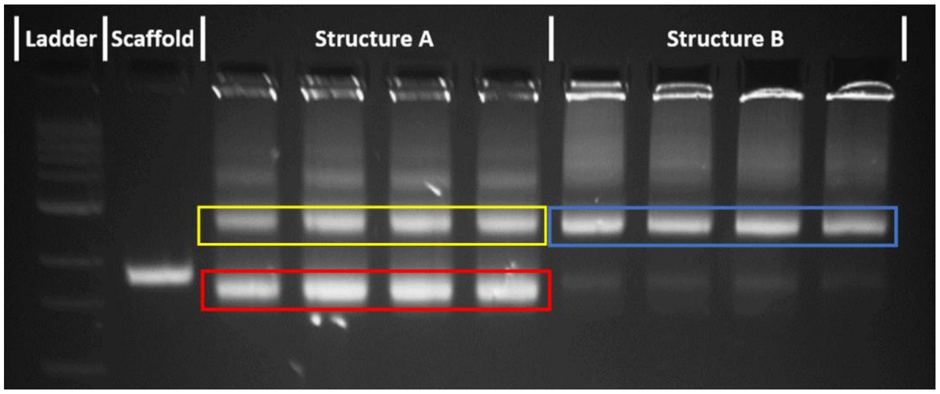


Figure 15: Agarose gel image after electrophoresis. The leftmost lane is a DNA ladder used for reference. The scaffold band is used for assessing the folding of structures. Structure A, whose bands are boxed in red, travelled farther than the scaffold, thus indicating well folding. Structure B, whose bands are boxed in blue, travelled less than the scaffold and in result is not well folded. Bands boxed in yellow indicate misfolded aggregated DNA. Both structures were folded under the same conditions.

2.3 Stability of DNA Origami Nanostructures

The design and synthesis of DNA origami nanostructures are only the first steps to take for their implementation in biological systems as bioengineering tools. Foreign DNA introduced into higher organisms is very likely to elicit immunological mechanisms for its detection and disposal [44]. In addition, numerous parameters such as temperature, lattice packing, and crossovers placement influence their structural stability in the presence of physiological agents. For instance, denaturing agents such as urea or GdmCl can disrupt the hydrogen bonds in DNA origami nanostructures at elevated temperatures [45].

One physiologically-relevant condition for the stability of DNA origami nanostructures is the concentration of cations in solution. Because of the negatively charged sugar-phosphate backbone of DNA, electrostatic repulsions can disrupt the binding of the staples to the scaffold if the concentration of cations is significantly low. One structural parameter that can influence a structure's stability in solutions with low cation concentrations is the lattice packing. As discussed in Section 2.1, the structural nature of the square lattice allows for the creation of densely packed objects with rectangular shapes that may experience undesired global twist deformations, while the honeycomb lattice allows for the creation of straight structures that are overall more porous [14]. Nevertheless, the alternatives to countermeasure the natural deformation of square lattices addressed in Section 2.1 and their inherent higher density with respect to honeycomb lattices can make them more suitable for the creation of nanostructures with surface modifications or increased stability.

To assess the stability provided by these lattice packings, an 18-helix bundle (hb) symmetrical rod with honeycomb lattice and an 18hb rectangular rod with square lattice(Figure 16) were incubated under decreasing concentrations of MgCl2. These structures have geometrical dimensions that are dependent on design and lattice packing. For instance, the 18hb rectangular rod has a larger length than the 18hb symmetrical rod, but because of the latter's lattice packing, it has the greater surface area. Further differences between these structures that are inherent to their designs include the scaffold and staple routing and the placement of crossovers.

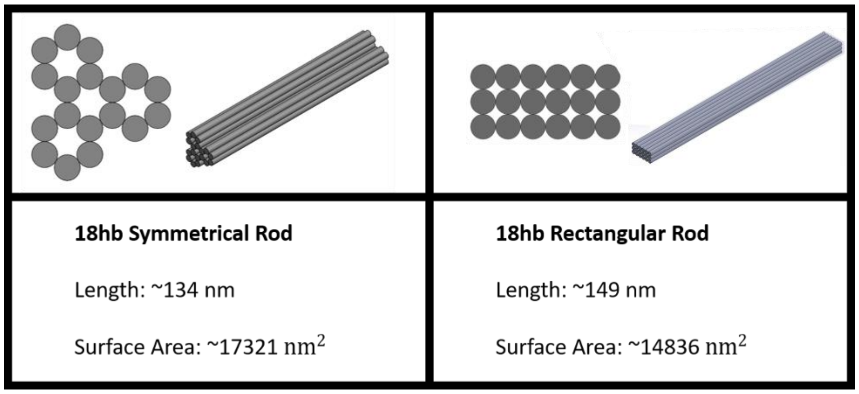


Figure 16: Schematics and structural parameters of nanostructures with honeycomb and square lattices. Top left, cross-sectional view and 3D schematic of the 18hb Symmetrical Rod. Bottom left, length and surface area of the symmetrical rod. Top right, cross-sectional view and 3D schematic of 18hb Rectangular Rod. Bottom right, length and surface area of rectangular rod.

To evaluate stability under decreasing salt concentrations, solutions with well folded nanostructures were mixed with an equal volume of 15% PEG 8000 in a centrifuge for 25 minutes at 4 °C. After removing the supernatant, the pellet containing the structures was resuspended in buffers containing 20-0 mM MgCl2 for 24 hours at room temperature. The structures were then loaded into an agarose gel for electrophoresis and their folding was qualitatively analyzed. Figure 17 shows that the 18hb symmetrical rod degrades at 10 mM MgCl2 while the 18hb rectangular rod degrades at 1 mM MgCl2. This would suggest that despite the larger length of the rectangular rod, its smaller surface area promotes the preservation of the double helical DNA bundles under low cation concentrations. The agarose gels were then quantitatively analyzed with a MATLAB code that normalizes the band intensities relative to the total integrated intensity. The gel intensity for the 18hb symmetrical rod remained almost constant until complete structure degradation occurred at 10 Mm MgCl2, while that for the 18hb rectangular rod gradually increased until 3 mM MgCl2. For well folded structures, the gel intensity may increase until reaching a peak as the concentration of salt decreases. This is due to the aggregation of structures at higher cation concentrations, which inhibit their movement from the negative to positive cations when the gel is charged. Because square lattice structures are densely packed as opposed to the porous honeycomb lattice structures, their propensity to degrade under adverse physiological conditions is reduced.

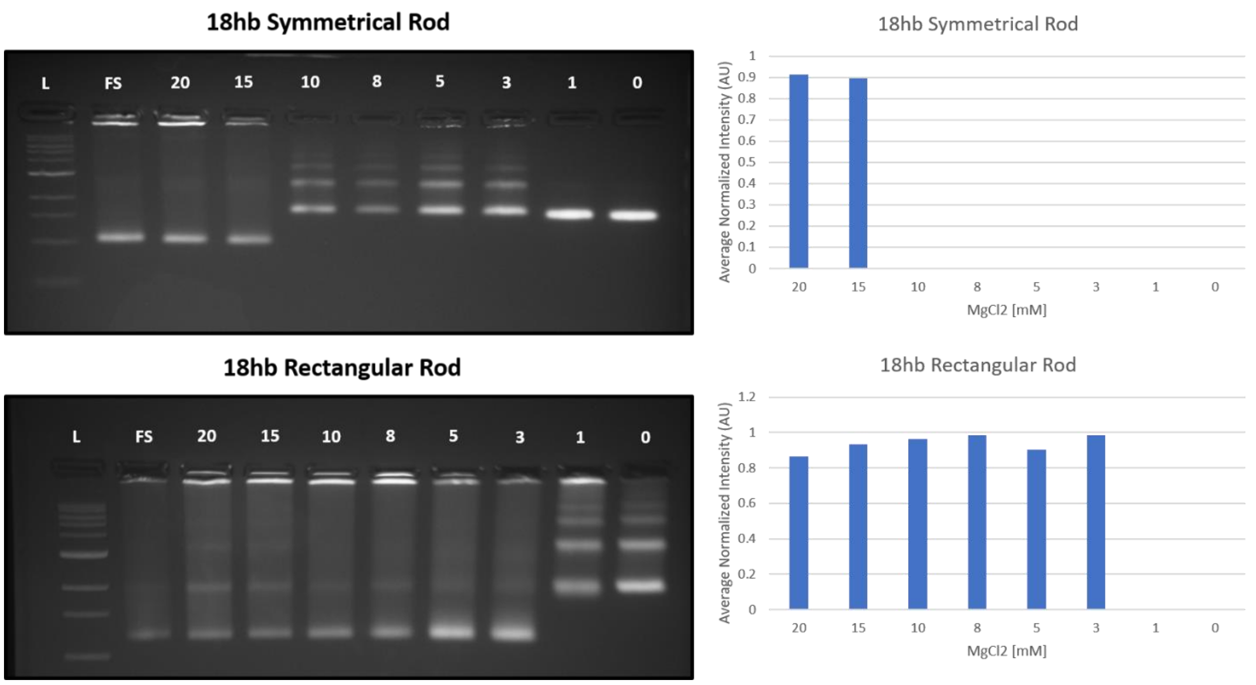


Figure 17: Agarose gel images and normalized gel intensities. Top left, L is the DNA ladder, and FS contains well folded 18hb symmetrical rods for comparison to the ones resuspended in 20-0 mM MgCl2. Structure degradation can be seen at 10 Mm MgCl2. Top right, the structure's band intensity remains almost constant until 15 mM MgCl2. Bottom left, the same conditions were applied to the 18hb rectangular rod, but structure degradation occurs at 1 mM MgCL2. Bottom right, the structure's band intensity gradually increases until 3 mM MgCl2.

The results presented here with respect to structural stability are not comprehensive, but they still provide insight into one of the parameters that influence the stability of DNA origami nanostructures. If DNA origami is to be implemented for biological applications, a more in-depth study of the strategies that could be executed to circumvent immunological responses is needed. Examples of such strategies are virus-inspired membranes [46] and oligolysine-based coatings [47] that achieve stability *in vivo* and increase pharmacokinetic bioavailability, thus indicating that the resilience of nanostructures can be enhanced with novel biological materials.

Chapter 3: Hierarchical Assembly for Signal Transmission

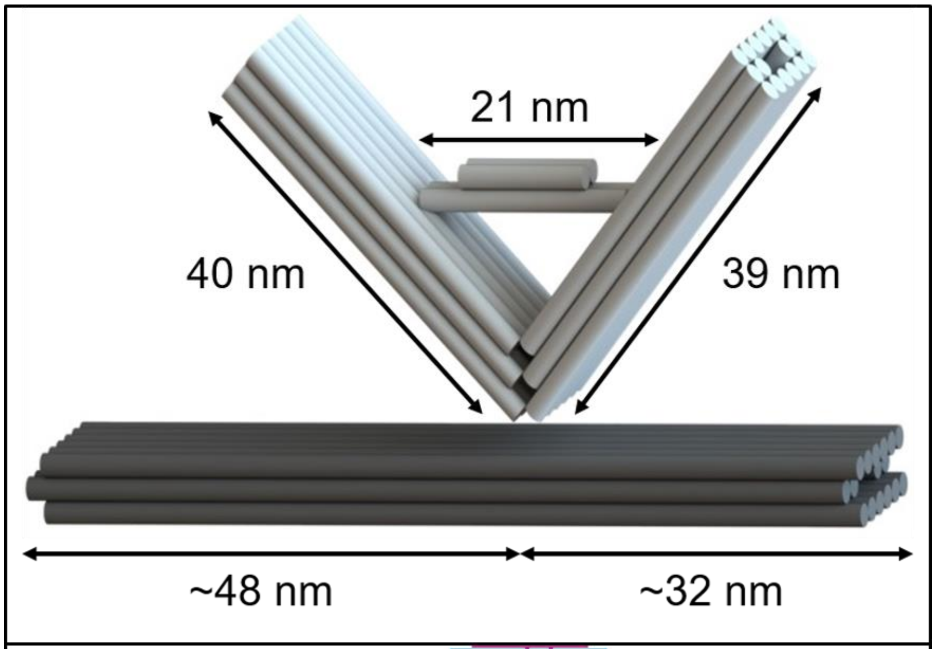
3.1 Introduction to Signal Transmission

Structural DNA nanotechnology also enables the creation of dynamic systems to transmit signals for the manipulation of biological processes. It has been shown that with DNA origami, it is possible to polymerize nanostructures to assemble dynamic mechanisms to propagate motion [34-36]. Nevertheless, these systems have either achieved signal transmission between lengths of only ~100-200 nm [34,35], or have propagated motion over several hundreds of nanometers by non-directional diffusion [36]. We expect that the DNA origami nanostructure proposed here will address these limitations by transmitting a signal across the microscale in a directional manner.

Here we focused on the development of an oscillator mechanism comprised of a V-shaped structure attached to a platform has been designed for long-range signal transmission. The two components of the oscillator mechanism are coupled by ssDNA connections that allow for one degree of rotational freedom back and forth. This oscillator nanostructure can be polymerized into an array that can propagate a mechanical signal throughout its length. Using DNA strand displacement, we can initiate actuation in a sequence-specific manner by inducing a conformational change on an initial oscillator that subsequently actuates oscillators throughout the array.

3.2 Design of Oscillator Nanostructure

As seen in Figure 18, the V-shaped structure consists of two rigid 16 double helical bundles arranged in a 3×6 square lattice that lack 2 double helical bundles in the center. The V-shaped structure also contains a rigid 4 double helical strut between the 3×6 bundles to keep them mechanically stable. The platform is a 16 double helical bundle with a similar cross-section. The V-shaped structure is connected to the platform at three locations that contain two neighboring 3 nucleotides (nt) long ssDNA scaffold for a total of 6 flexible connections, allowing for one degree of rotational motion. The oscillator nanostructure is asymmetrical, making it possible to visually differentiate whether the V-shaped structure is constrained to the left or right side of the platform.



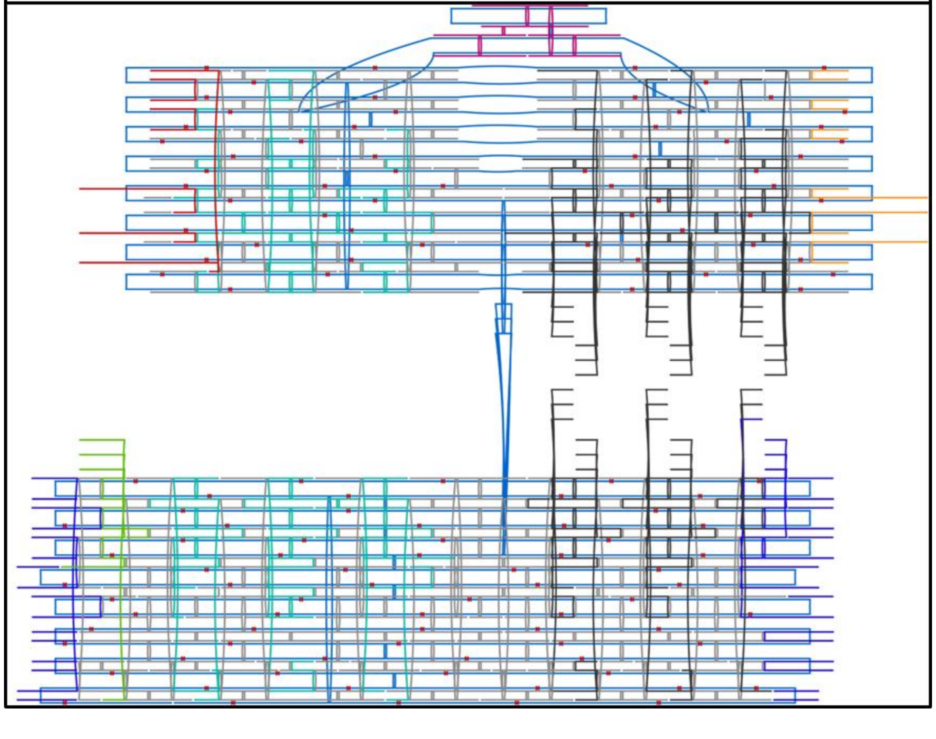


Figure 18: 3D Schematic with dimensions and caDNAno schematic of oscillator. Top, Isometric view showing the 3x6 square packing of the V-shaped structure and dimensions highlighting the asymmetry of the oscillator. Bottom, CaDNAno schematic where the light blue lines represent the scaffold while other colored lines represent the staples. The red x's represent deleted base pairs to correct the inherent global twist deformation of square lattice structures.

This nanostructure uses an 8064-base long scaffold that is mixed with an excess of staples in the ionic solution discussed in Section 2.2. The solution is heated in a thermal ramp to 65 °C and is then cooled down at 54-51 °C for three hours per degree before quenching it to 4 °C. The structural stability of the oscillator was evaluated under decreasing salt concentrations by following a similar procedure to that explained in Section 2.3. Characterization of the structure via TEM has led us to conclude that folding conditions are optimized at a concentration of 20 mM MgCl2.

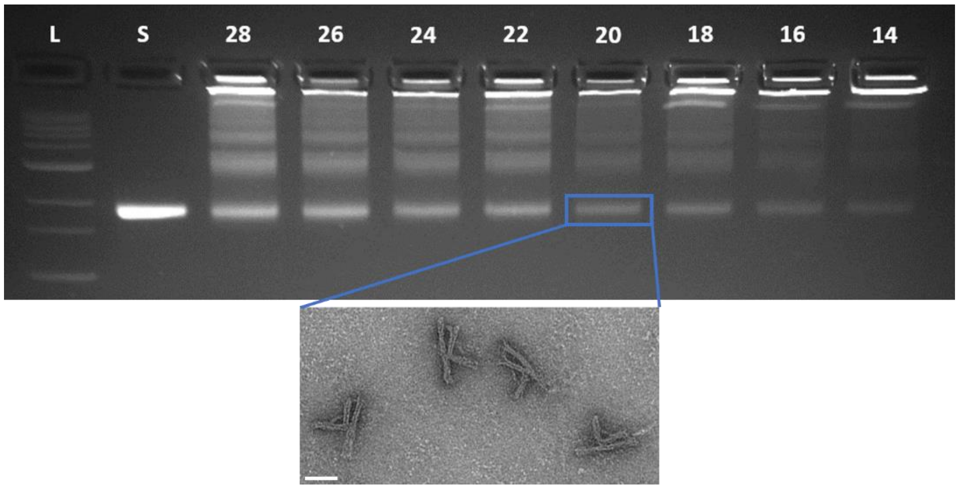


Figure 19: Agarose gel image and TEM image of folded oscillators. Top, the oscillator was folded in solutions at 28-14 mM MgCl2 and compared to its scaffold (8064) after gel electrophoresis. Bottom, TEM image of well folded oscillators at 20 mM MgCl2. Scaler bar: 50 nm.

The oscillator can be constrained to the left or right side of the platform by extending ssDNA staples at specific locations beyond the double helical bundles. Figure 20 shows that on the left, three unique 15 nt long ssDNA latches located on the V-shaped structure can bind to partially complementary 22 nt long ssDNA latches located on the platform. On the right, eighteen weakly complementary 4 nt long ssDNA overhangs located on the V-shaped structure and platform can bind to each other.

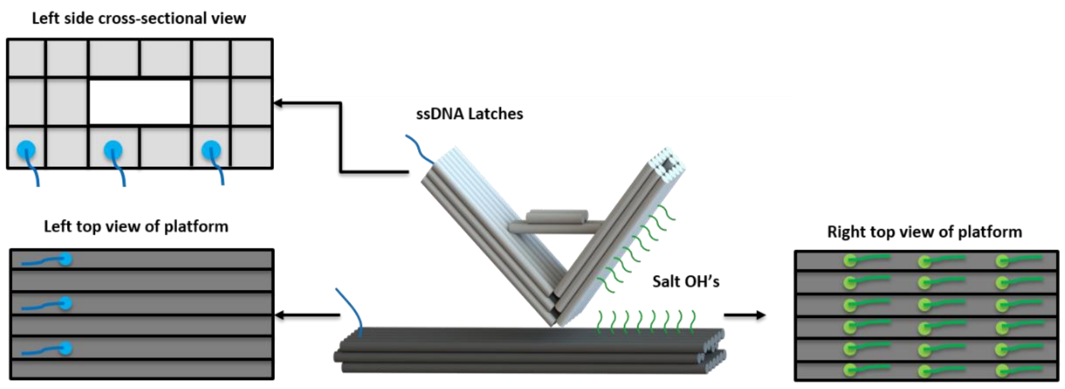

Left side cross-sectional view
ssDNA Latches
Left top view of platform
Salt OH's
Right top view of platform

Figure 20: 3D schematic of oscillator and 2D schematics of the placement of ssDNA latches and overhangs. Top left, cross-sectional view of V-shaped structure with the three latches located on specific double helical bundles. Bottom left, top view of platform with the three latches complementary to those on the V-shaped structure. Center, 3D schematic of oscillator with ssDNA latches and overhangs. Top right, top view of platform with the 18 overhangs weakly complementary to those on the V-shaped structure.

3.3 Actuation of Oscillator Monomers

The latches hold the oscillator constrained to the left side indefinitely, but for actuation, 22 nt long ssDNA inputs fully complementary to the bottom latches can be introduced into solution. As seen in Figure 21, these inputs initially hybridize with the unpaired 7 nt long toehold sequences in the bottom latches and then displace the top latches via strand displacement. Once the top latches are displaced, the hybridized bottom latches become inert and the oscillator is released to its unconstrained state. By increasing the cation concentration in solution, the electrostatic repulsions between the weakly complementary overhangs on the right side can be screened and these can bind. In result, the oscillator rotates and is constrained to the right by the overhangs.

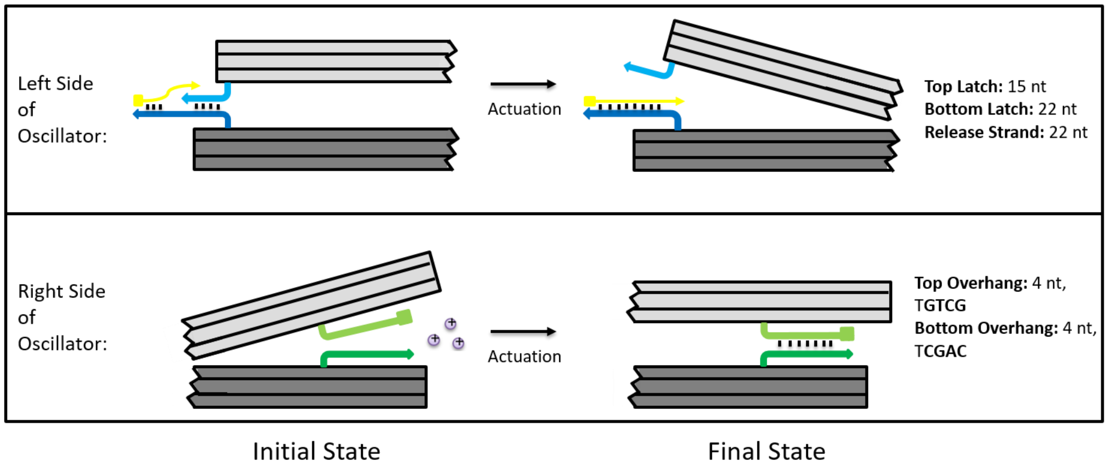


Figure 21: Strand displacement reactions on the left and right sides of the oscillator during actuation. Top, a 22 nt long ssDNA input fully complementary to the bottom left latch hybridizes and displaces the top left latch, thus releasing the oscillator. Bottom, increasing the cation concentration in solution promotes binding of the weakly complementary overhangs and in result rotate the oscillator to the right.

Figure 22 demonstrates the efficiency of our procedure for the actuation of the oscillator monomer. Through TEM imaging we determined that approximately 95% of well folded oscillators were constrained to the left side by the single-stranded (ss) DNA latches. ssDNA inputs and MgCl2 were then introduced into solution to induce the displacement of the top latches and screen the electrostatic repulsions of the overhangs, resulting in the 90° rotation of the oscillators. Approximately 80% of the oscillators were constrained to the right side after actuation. The actuation procedure was also repeated with MgCl2 but no ssDNA inputs to confirm that the actuation of the oscillator only occurs after the strand displacement reaction takes place.

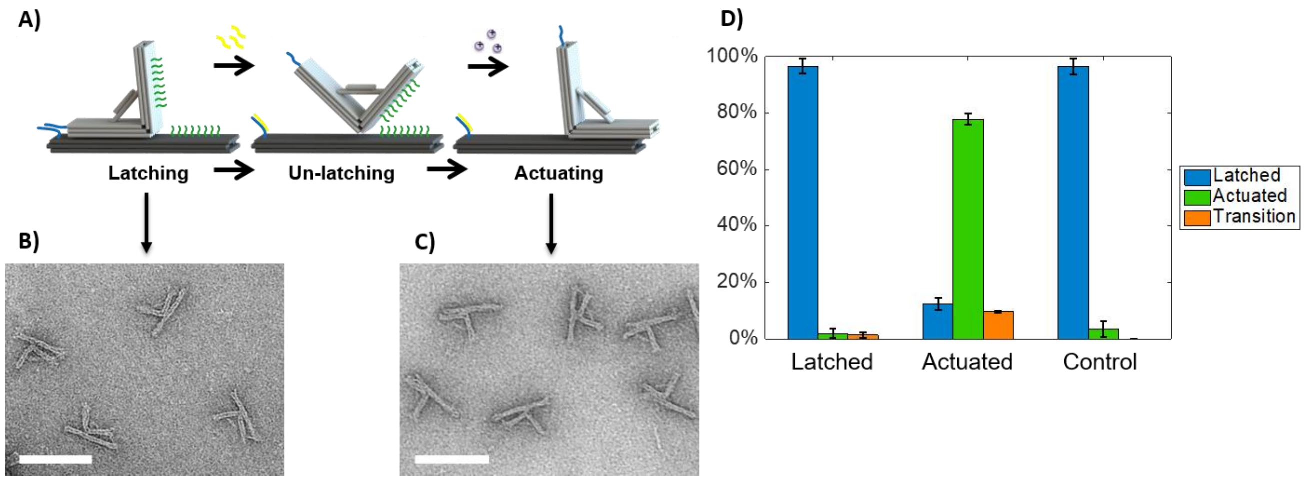


Figure 22: Actuation of oscillator monomers. A) 3D schematic of the actuation process where the introduction of ssDNA inputs and cations in solution unlatch the oscillator and rotate it to the right. B) TEM image of oscillators before actuation. C) TEM image of oscillators after actuation. D) Data from the actuation process, indicating efficiency of latching and actuation relative to a control. Scale bars: 100 nm.

The actuation of the oscillator monomer was also characterized via bulk fluorescence resonance energy transfer (FRET) using a spectrofluorometer. FRET is the transfer of energy between a pair of fluorescent molecules, usually denominated donor and acceptor. These molecules can interact over distances smaller than 10 nanometers [3, 48], which in result allows us to integrate them to specific ssDNA overhangs on the oscillator to characterize its actuation. As shown in Figure 23, FRET is a function of the spectral overlap between the donor's emission spectrum and the acceptor's excitation spectrum.

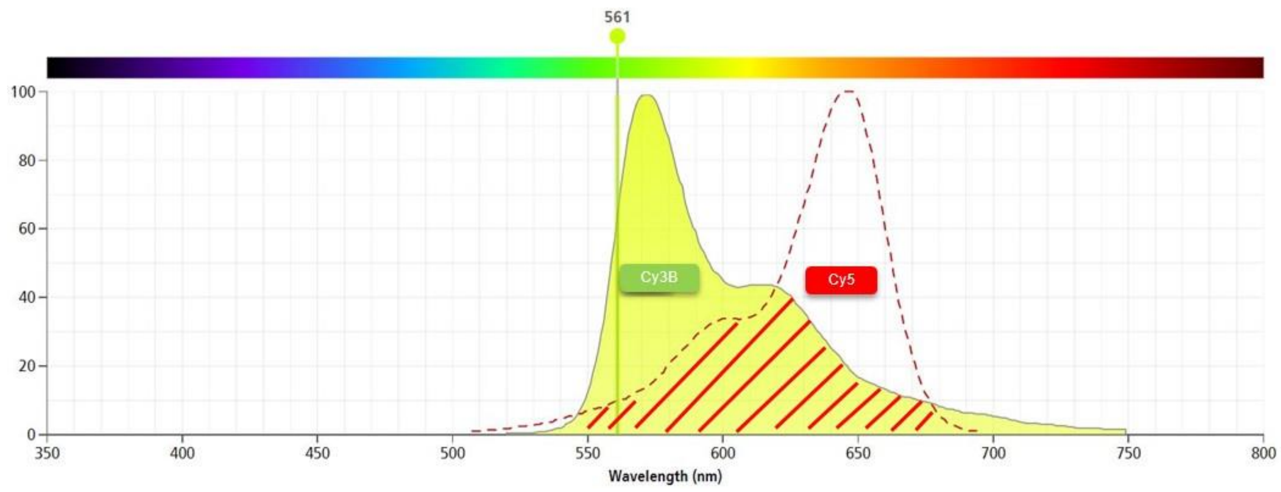


Figure 23: Spectral overlap between Cy3B and Cy5 fluorophores. At a wavelength of 561 nm, Cy3B's emission spectrum overlaps Cy5's excitation spectrum over the area highlighted in red. Adapted from http://www.bdbiosciences.com/us/s/spectrumviewer.

A sample of oscillator monomers was inserted into a quartz cuvette that has three clear windows for a beam of light to pass through, and this was placed inside a dark chamber in the spectrofluorometer. A 520 nanometers laser was used to excite a Cy3B donor fluorophore located on the V-shaped structure. Before actuation, this fluorophore emits green light, but after actuation, its energy is transferred to a Cy5 acceptor fluorophore located on the platform, which in result emits red light. The emission and excitation fluorescence data was recorded by the spectrofluorometer, and this was processed with a MATLAB code to calculate a normalized FRET efficiency before and after actuation. An increase in FRET efficiency from 17.68% before actuation to 38.98% after actuation was determined.

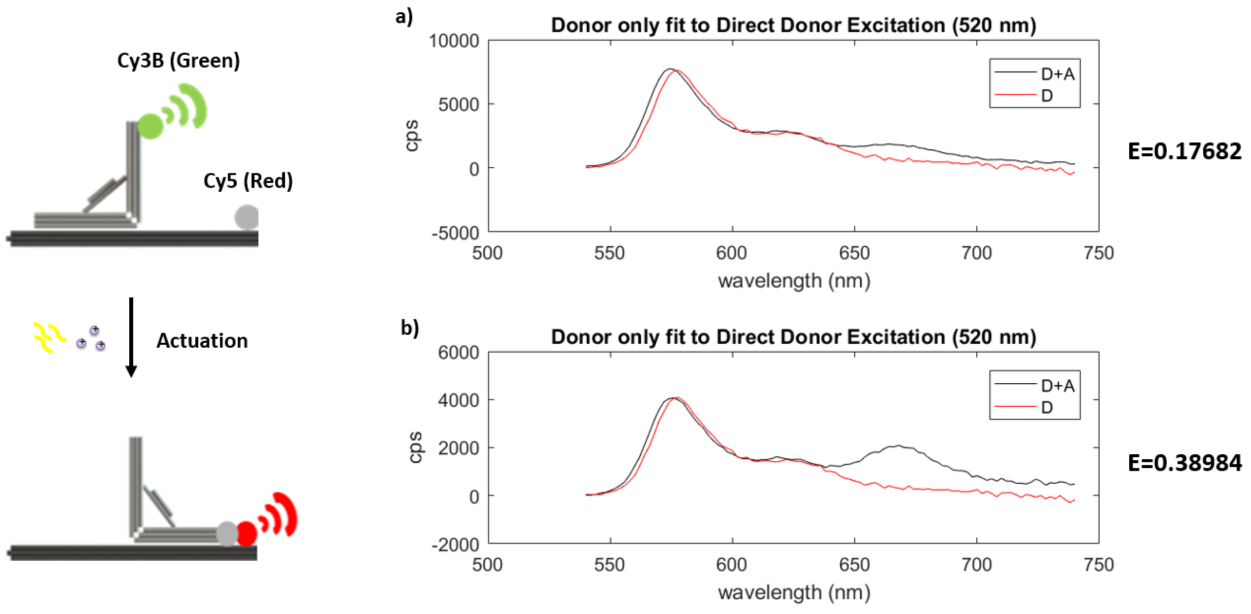


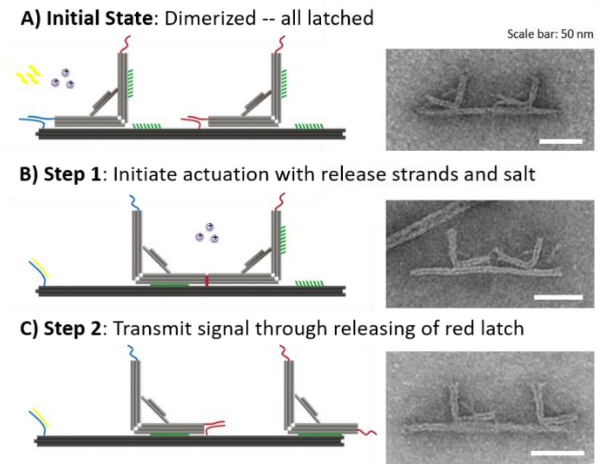
Figure 24: 3D schematics of actuation with fluorophores and FRET efficiencies before and after actuation. Left, the actuation of the oscillator decreases the distance between Cy3B and Cy5, thus allowing for FRET to occur. a) before actuation the donor (D) and FRET (D+A) signals overlap as there is no FRET. b) the FRET peak over the 650-500 nm range indicates an increase in the signal of Cy5.

3.4 Polymerization of Oscillator for Signal Transmission

To achieve the propagation of a mechanical signal across long-range distances, oscillators can be polymerized so that the platforms of multiple monomers are coupled to create arrays that span several hundreds of nanometers. This is possible due to the complementary geometry of these nanostructures and the hybridization of ssDNA scaffolds on the left and right side of the platforms. ssDNA blocking units can also be added that hybridize to the scaffold on either side of the platform to prevent polymerization on that side, thus allowing us to control initial and final monomers.

A starter monomer and a polymerization monomer have been designed to start actuation and transmit a signal respectively. The difference between these monomers is the DNA sequence of the ssDNA latches located on both sides of the V-shaped structure. As previously explained, the starter monomer has left latches with unique sequences, but in addition, it has top right latches whose sequences are fully complementary to the left latches on the platform of the polymerization monomer. These top right latches are also included on the polymerization monomer. After ssDNA inputs are introduced for the actuation of the starter monomer, this will rotate to the right by the increase in salt and release the next oscillator via strand displacement. Once this oscillator rotates to the right, it will release a third oscillator, and so on. In result, the top right latches are the inputs for each subsequent polymerization monomer. The complementarity between the right latches on the V-shaped structure and the left latches on the platform of a polymerization monomer is the key to propagating a mechanical signal.

Figure 25 demonstrates the efficiency of our procedure for signal transmission by coupling one starter oscillator and one polymerization oscillator. The ssDNA blocking units discussed earlier were included to ensure that only oscillator dimers assembled. Through TEM imaging we determined that approximately 90% of well folded dimers were constrained to the left side by the ssDNA latches. ssDNA inputs and MgCl2 were then introduced into solution to actuate the starter oscillator, which rotates to the right and presents the inputs to induce the displacement of the top left latches on the polymerization oscillator. This oscillator then rotates due to the binding of the ssDNA overhangs. Approximately 60% of the dimers were constrained to the right side after actuation.


A) Initial State: Dimerized -- all latched
Scale bar: 50 nm
B) Step 1: Initiate actuation with release strands and salt
C) Step 2: Transmit signal through releasing of red latch

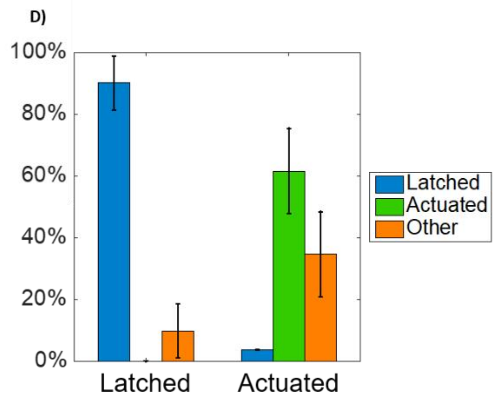


Figure 25: Actuation of oscillator dimers. A) Initial state before actuation. B) Intermediate state after actuation. C) Signal transmission after complete actuation. D) Data from the actuation process, indicating efficiency of latching and actuation.

The actuation of the oscillator dimers demonstrates the process through which a mechanical signal can be transmitted, and this process translates for the actuation of larger polymers. As discussed earlier in this section, their assembly is facilitated by complementary geometry and the binding of ssDNA scaffolds on the platforms, but the ssDNA blocking units are not included. This allows the polymerization of a starter oscillator with multiple polymerization oscillators to create arrays that span over the microscale. The actuation of the starter oscillator triggers the next oscillator, which in turn rotates and triggers subsequent oscillators in a sequential manner.

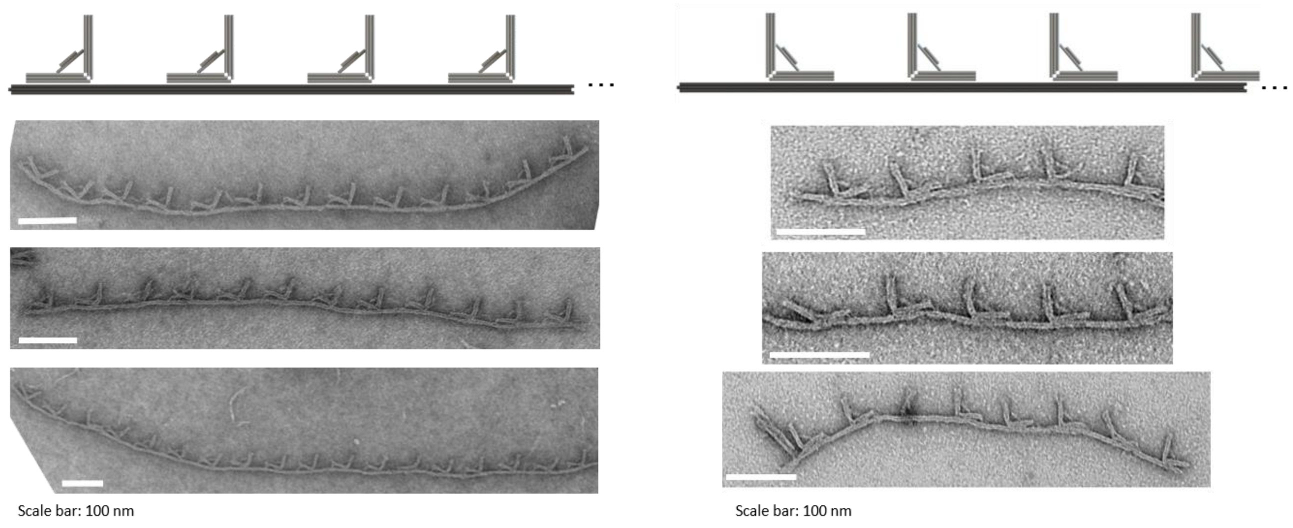


Figure 26: Oscillator polymers before and after actuation.

3.5 Discussion and Future Work

DNA Origami has allowed us to control higher-order dynamic arrays from the self-assembly and polymerization of an oscillator nanostructure. By employing DNA strand displacement reactions and increasing the salt in solution, a starting oscillator is actuated and triggers the next oscillator in line, resulting in the propagation of the actuation mechanism throughout the array. The results presented in this work would indicate that the limitations from previous studies [34-36] have been addressed effectively, as the hierarchical assembly of the oscillator nanostructure allows us to transmit a mechanical signal over several hundreds of nanometers in a directional manner.

While previous studies have demonstrated the actuation of dynamic nanostructures in a time-scale of minutes [29, 49], we expect that binding of the ssDNA overhangs promoted by an increase in salt will allow for the actuation of the oscillator in a time-scale of seconds. Current work focuses on evaluating the actuation time of oscillator polymers via single molecule FRET on a total internal reflection fluorescence (TIRF) microscope. To achieve this, a flow system has been devised to introduce ssDNA inputs and salt in real time and observe the actuation of oscillator polymers binding to a glass coverslip. Figure 27 shows the coverslip functionalized with polyethylene glycol (PEG), biotin (red), and streptavidin (yellow) that binds to the biotin. Finally, casein (grey) is added to prevent non-specific binding of the oscillator polymers, which have biotinylated overhangs on the platforms that bind to the streptavidin.

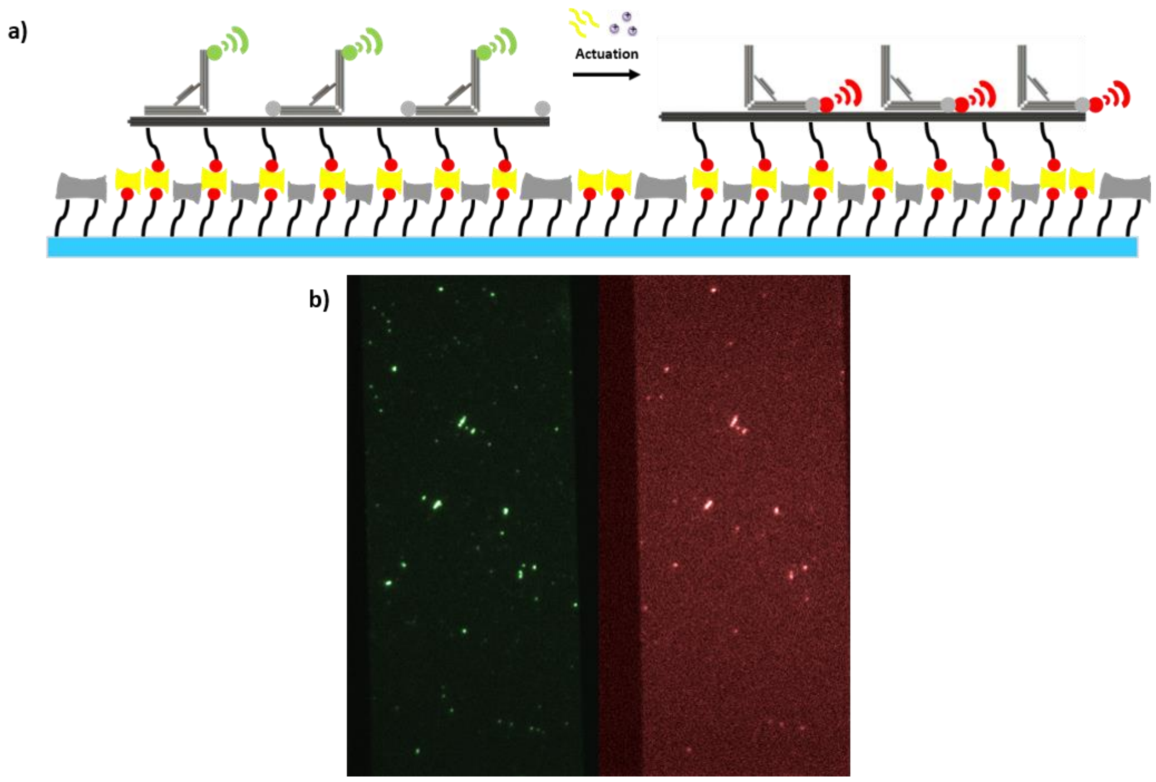


Figure 27: Single Molecule FRET setup. a) Schematic showing FRET after actuation of oscillator polymers attached to a biotin-PEG functionalized coverslip by biotin-streptavidin biding. b) Example of TIRF imaging display.

The flow system is placed on the TIRF microscope and a 561-nanometer laser is used to excite the Cy3B donor fluorophore for the emission of green light. As discussed in Section 3.2, after actuation its energy is transferred to the Cy5 acceptor fluorophore that emits red light. The acceptor fluorophore is also directly excited with a 640 nm laser to verify its presence. This single molecule FRET procedure will be repeated with MgCl2 but no ssDNA inputs to confirm that the actuation of the oscillator polymers only occurs after DNA strand displacement. The emission and excitation fluorescence data recorded by the TIRF will be processed with a MATLAB code to calculate a normalized FRET efficiency over time.

Short-term goals for this project will focus on optimizing the signal transmission mechanism by evaluating design parameters such as the number and location of the ssDNA latches and overhangs, or redesigning the complementary geometry between oscillators. We expect that improvements on the design of the nanostructure will allow us to observe an increase in FRET efficiency and the propagation of motion in a time-scale of seconds. Previous studies have shown the versatility of DNA Origami in engineered biological systems [5], and we expect that in the long-term this oscillator mechanism could be implemented to regulate and measure long-range biological processes.

Chapter 4: Conclusions

Recent research with DNA origami has focused on the development of dynamic systems that can mimic the transfer of information that occurs in biological processes. This thesis presents an innovative oscillator nanostructure that can be polymerized to transmit a mechanical signal across microscale distances in a directional manner. We expect that our actuation procedure, which uses DNA strand displacement reactions and the binding of weakly complementary ssDNA overhangs, will allow our oscillator polymers to propagate motion on a time-scale of seconds.

The results presented here are promising for engineering complex molecular transport systems, but DNA origami still has limitations to overcome. Biological applications would require the production of large amounts of nanostructures, and in addition, physiological barriers compromise their structural stability. Thankfully, ongoing research has focused on addressing some of these issues, giving the knowledge needed to eventually engineer our oscillator nanostructure into biological systems. Our contribution to DNA origami with this dynamic system will hopefully also accelerate the design and implementation of controllable mechanisms that regulate the transmission of signals of biological processes.

Bibliography

1. R. Boyle, "7 Amazing Ways Nanotechnology Is Changing The World", November 14, 2012. from https://www.popsci.com/science/article/2012-11/7-amazing-ways-nanotechnology-changing-world.
2. S. Parveen, R. Misra, and S.K. Sahoo, "Nanoparticles: a boon to drug delivery, therapeutics, diagnostics and imaging", *Nanomedicine: Nanotechnology, Biology, and Medicine*, pp 147-166, 2012.
3. V. V. Didenko, "DNA probes using Fluorescence Resonance Energy Transfer (FRET): designs and applications", *Biotechniques*, pp. 1106-1121, 2001.
4. Q. Zhang, Q. Jiang, N. Li, L. Dai, Q. Liu, L. Song, J. Wang, Y. Li, J. Tian, B. Ding, and Y. Du, "DNA origami as an in vivo drug delivery vehicle for cancer therapy", *ACS Nano*, pp 6633-6643, 2014.
5. E. Akbari, M. Y. Mollica, C. R. Lucas, S. M. Bushman, R. A. Patton, M. Shahhosseini, J. W. Song, and C. E. Castro, "Engineering Cell Surface Function with DNA Origami", *Advanced Materials*, 2017.
6. M. Iwaki, S.F. Wickham, K. Ikezaki, T. Yanagida, and W.M. Shih, "A programmable DNA origami nanospring that reveals force-induced adjacent binding of myosin VI heads", *Nature Communications*, 2016.
7. S. Ranallo, C. Prevost-Tremblay, A. Idili, Al. Vallee-Belisle, and F. Ricci, "Antibody-powered nucleic acid release using a DNA-based nanomachine", *Nature Communications*, 2017.
8. J.D. Watson and F.H. Crick, "Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid", *Nature*, pp 737-738, 1953.
9. L. A. Pray, "Discovery of DNA structure and function: Watson and Crick", *Nature Education*, 2008.
10. Seeman, N. C. (2015). *Structural DNA Nanotechnology*. Cambridge: Cambridge University Press.
11. N. C. Seeman, "Nucleic acid junctions and lattices", *Journal of Theoretical Biology*, pp. 237-247, 1982.
12. N. C. Seeman, "Nucleic acid nanostructures and topology", *Angewandte Chemie-International Edition,* pp. 3220-3238, 1998.
13. P. W. Rothemund, "Folding DNA to create nanoscale shapes and patterns", *Nature*, pp. 297-302, 2006.
14. C. E. Castro, F. Kilchherr, D-N. Kim, E. L. Shiao, T. Wauer, P. Wortmann, M. Bathe, and H. Dietz, "A primer to scaffolded DNA origami", *Nature Methods*, pp. 221-229, 2011.
15. S. M. Douglas, H. Dietz, T. Liedl, B. Hogberg, F. Graf, and W. M. Shih, "Self-assembly of DNA into nanoscale three-dimensional shapes", *Nature,* pp. 414-418, 2009.

16. H. Dietz, S. M. Douglas, and W. M. Shih, “Folding DNA into twisted and curved nanoscale shapes”, *Nature*, pp. 725-730, 2009.

17. E. S. Anderson, M. Dong, M. M. Nielsen, K. Jahn, R. Subramani, W. Mandouh, *et al.*, “Self-assembly of a nanoscale DNA box with a controllable lid”, *Nature*, pp. 73-76, 2009.

18. F. Zhang, J. Nangreave, Y. Liu, and H. Yan, “Reconfigurable DNA origami to generate quasifractal patterns”, *Nano Letters*, pp. 3290-3295, 2012.

19. T. Tomaru, Y. Suzuki, I. Kawamata, S. M. Nomura, and S. Murata, “Stepping operation of rotary DNA origami device”, *Chem. Commun.*, 2017.

20. E. Pfitzner, C. Wachauf, F. Kilchherr, B. Pelz, W. M. Shih, M., and H. Dietz, “Rigid DNA beams for high-resolution single-molecule mechanics”, *Angewandte Chemie-International Edition*, pp. 7766-7771, 2013.

21. Y. Ke, T. Meyer, W. M. Shih, and G. Bellot, “Regulation at a distance of biomolecular interactions using a DNA origami nanoactuator”, *Nature Communications*, p. 10935, 2016.

22. S. M. Douglas, I. Bachelet, and G. M. Church, “A logic-gated nanorobot for targeted transport of molecular payloads’, *Science*, pp. 831-834, 2012.

23. Q. Jiang, C. Song, J. Nangreave, X. Liu, L. Lin, D. Qiu, Z.-G. Wang, G. Zou, X. Liang, H. Yan, and B. Ding, “DNA origami as a carrier for circumvention of drug resistance”, *Journal of the American Chemical Society*, pp. 13396-13403, 2012.

24. P. D. Halley, C. R. Lucas, E. M. McWilliams, M. J. Webber, R. A. Patton, C. Kural, *et al.*, “Daunorubicin-Loaded DNA Origami Nanostructures Circumvent Drug-Resistant Mechanisms in a Leukemia Model,” *Small*, 2015.

25. D. Y. Zhang and G. Seelig, “Dynamic DNA nanotechnology using strand-displacement reactions”, Nature Chemistry, pp. 103-113, 2011.

26. B. Yurke, A. J. Turberfield, A. P. Mills Jr, F. C. Simmel, and J. L. Neumann, “A DNA-fuelled molecular machine made of DNA”, Nature, pp. 605-608, 2000.

27. L. Zhou, A. E. Marras, H. J. Su, and C. E. Castro, “DNA Origami Compliant Nanostructures with Tunable Mechanical Properties”, ACS Nano, pp. 27-34, 2014.

28. C. E. Castro, H. J Su, A. E. Marras, L. Zhou, and J. Johnson, “Mechanical Design of DNA Nanostructures”, Nanoscale, pp. 5913-5921, 2015.

29. A. E. Marras, L. Zhou, H. J. Su, and C. E. Castro, “Programmable Motion of DNA Origami Mechanisms”, Proc Natl Acad Sci U S A, pp. 713-718, 2015.

30. L. Zhou, A. E. Marras, H. J. Su, and C. E. Castro, “Direct Design of an Energy Landscape with Bistable DNA Origami Mechanisms”, Nano Letters, pp. 1815-1821, 2015.

31. K. Lund, A. J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, *et al.*, “Molecular robots guided by prescriptive landscapes”, *Nature*, pp. 206-210, 2010.

32. H. Gu, J. Chao, S. J. Xiao, and N. C. Seeman, “A proximity-based programmable DNA nanoscale assembly line”, *Nature*, pp. 202-205, 2010.

33. T. E. Tomov, R. Tsukanov, Y. Glick, Y. Berger, M. Liber, D. Avrahami, *et al.*, "DNA bipedal motor achieves a large number of steps due to operation using microfluidics-based interface", *ACS Nano*, pp. 4002-4008, 2017.

34. J. Song, Z. Li, P. Wang, T. Meyer, C. Mao, Y. Ke, "Reconfiguration of DNA molecular arrays driven by information relay", *Science*, pp, 2017.

35. S. Helmig, and K. V. Gothelf, "AFM imaging of hybridization chain reaction mediated signal transmission between two DNA origami structures", *Angewandte Chemie-International Edition, pp. 13633-13636, 2017.*

36. J. List, E. Falgenhauer, E. Kopperger, G. Pardatscher, and F. C. Simmel, "Long-range movement of large mechanically interlocked DNA nanostructures", *Nature Communications,* 2016.

37. Z. Lium M. Liu, L. Wang, J. Nangreave, H. Yan, and Y. Liu, "Molecular behavior of DNA origami in higher-order self-assembly", *Journal of the American Chemical Society*, pp. 13545-13552, 2010.

38. S. M. Douglas, A. H. Marblestone, S. Teerapittayanon, A. Vazquez, G. M. Church, and W. M. Shih, "Rapid prototyping of 3D DNA-origami shapes with caDNAno", Nucleic Acids Research, pp. 5001-5006, 2009.

39. Y. Ke, S. M. Douglas, M. Liu, J. Sharma, A. Cheng, A. Leung, Y. Liu, W. M. Shih, and H. Yan, "Multilayer DNA origami packed on a square lattice", *Journal of the American Chemical Society*, pp. 15903-15908, 2009.

40. J. M. Arbona, J. P. Aime, and J. Elezgaray, "Cooperativity in the annealing of DNA origamis", *The Journal of Chemical Physics*, p. 0.15105, 2013.

41. J. P. Sobczak, T. G. Martin, T. Gerling, and H. Dietz, "Rapid folding of DNA into nanoscale shapes at constant temperature" *Science*, pp. 1458-1461, 2012.

42. A. E. Marras, L. Zhou, V. Kolliopoulos, H. J. Su, and C. E. Castro, "Directing Folding Pathways for Multi-Component DNA Origami Nanostructures with Complex Topology", *New Journal of Physics*, p. 055005, 2016.

43. E. Stahl, T. G. Martin, F. Praetorius, and H. Dietz, "Facile and scalable preparation of pure and dense DNA origami solutions", *Angewandte Chemie-International Edition*, 2014.

44. S. Surana, A. R. Shenoy, Y. Krishman, "Designing DNA nanodevices for compatibility with the immune system of higher organisms", *Nature Nanotechnology*, 2015.

45. S. Ramakrishnan, G. Krainer, G. Grundmeier, M. Schlierf, and A. Keller, "Structural stability of DNA origami nanostructures in the presence of chaotropic agents", *Nanoscale*, pp. 10398-10405, 2016.

46. S. D. Perrault, W. M. Shih, "Virus-inspired membrane encapsulation of DNA nanostructures to achieve in vivo stability", *ACS Nano*, pp. 5132-5140, 2014.

47. N. Ponnuswamy, M. M.C. Bastings, B. Nathwani, J. H. Ryu, L. Y.T. Chou, M. Vinther, et al., "Olygosine-based coating protects DNA nanostructures from low-salt denaturation and nuclease degradation", *Nature Communications*, 2017.

48. R. Roy, S. Hohng, and T. Ha, "A practical Guide to single molecule FRET", *Nature Methods*, pp. 507-516, 2008.

49. T. Gerling, K. F. Wagenbauer, A. M. Neuner, and H. Dietz, "Dynamic DNA devices and assemblies formed by shape-complementary non-base pairing 3D components", *Science*, pp. 1446-1452, 2015.

Appendix A: Additional caDNAno Schematics

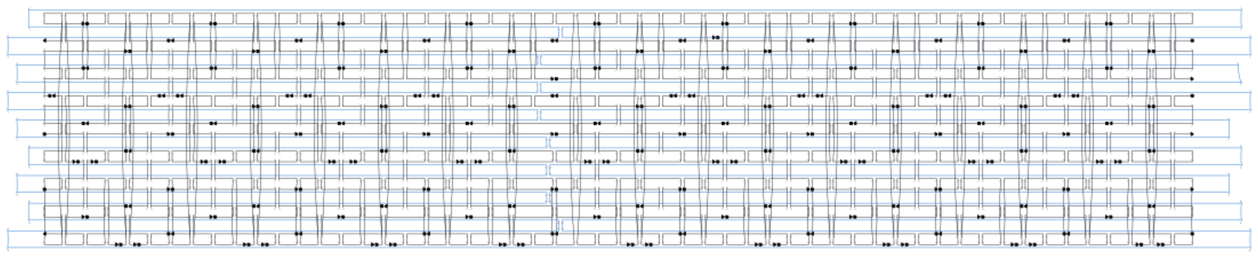


Figure 28: caDNAno schematic for 18hb Symmetrical Rod. The light blue lines represent the 7249-base long scaffold and the black lines represent the staples.

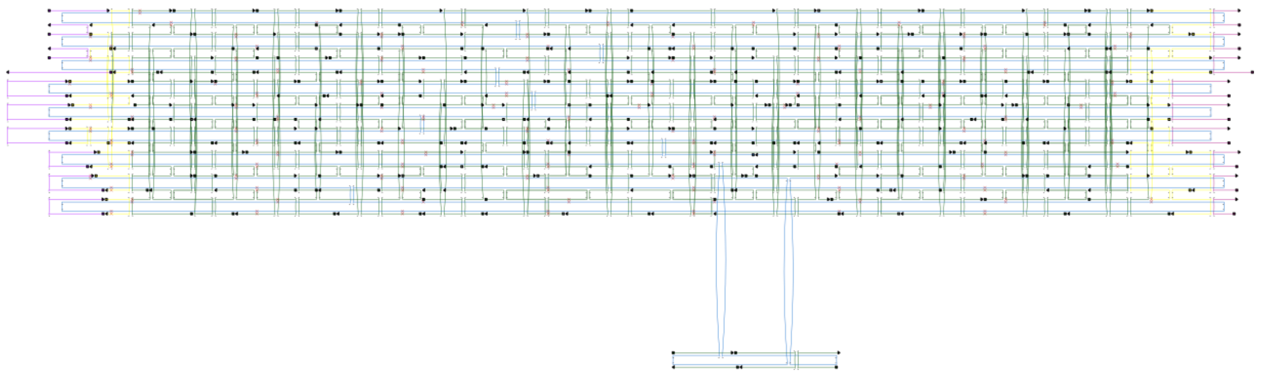


Figure 29: caDNAno schematic for 18hb Rectangular Rod. The light blue llines represent the 8064-base long scaffold and the colored lines represent the staples.

Appendix B: Additional TEM Images

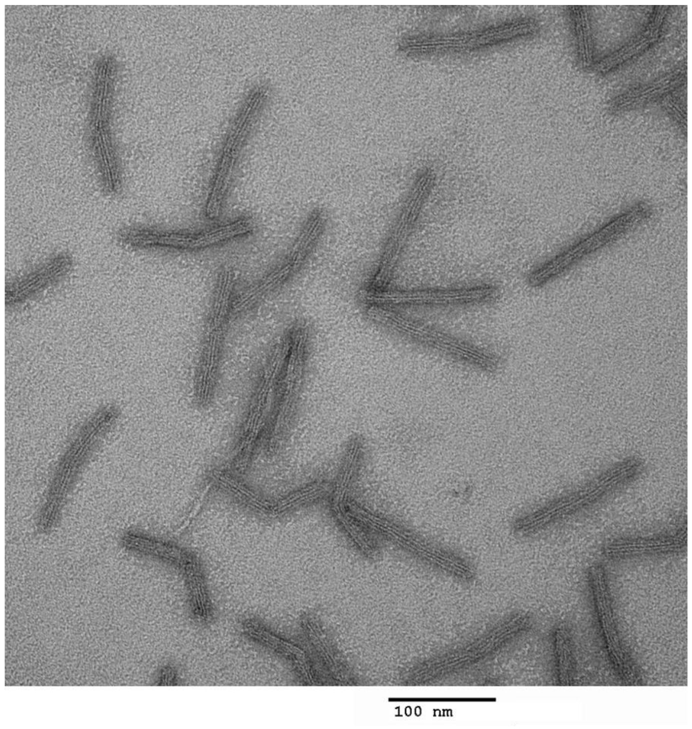


Figure 30: TEM Image of 18hb Symmetrical Rods at 20 mM MgCl2.

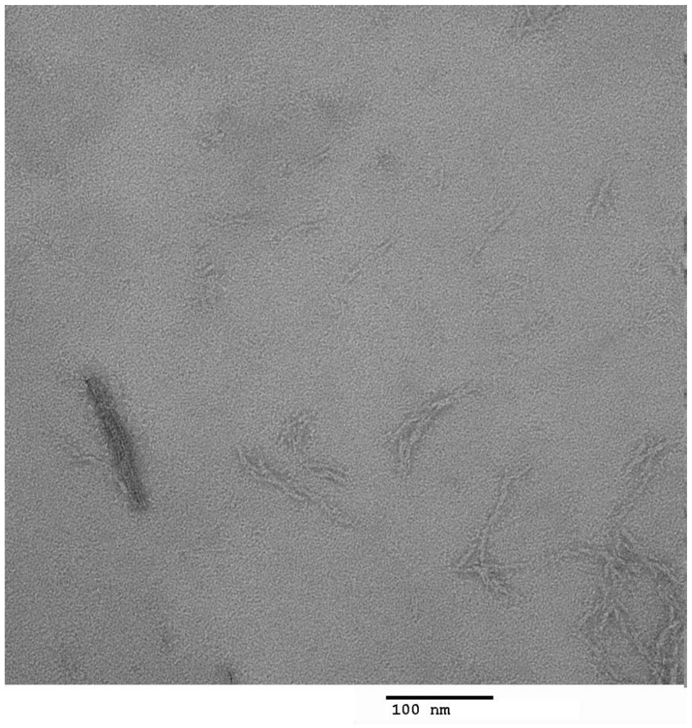


Figure 31: TEM Image of 18hb Symmetrical Rod at 3 mM MgCl2.

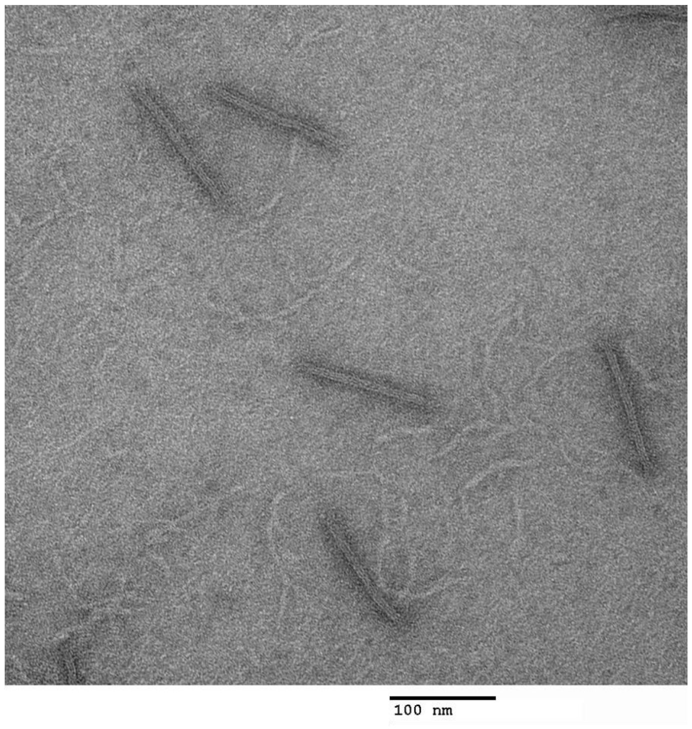


Figure 32: TEM Image of 18hb Rectangular Rods at 20 mM MgCl2.

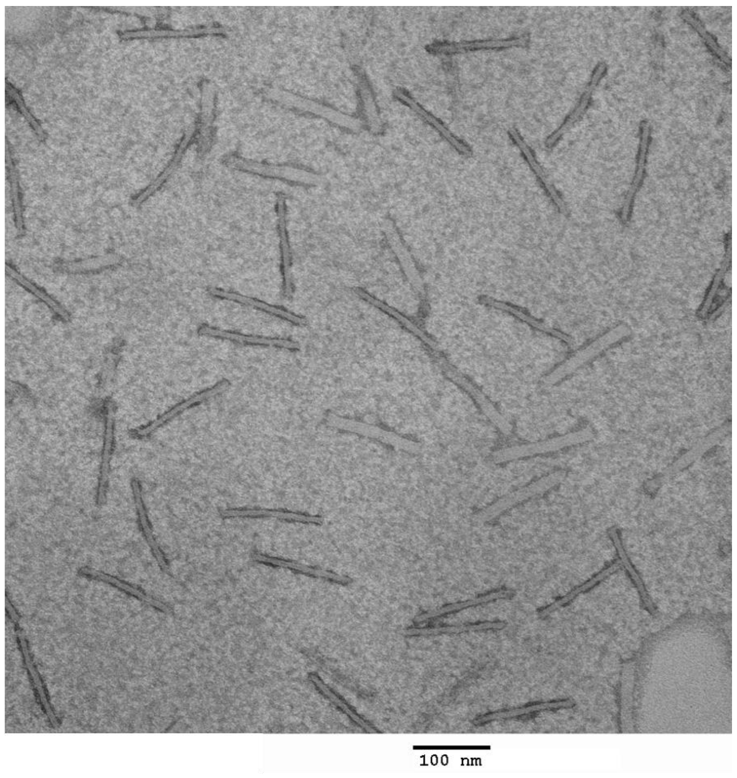


Figure 33: TEM Image of 18hb Rectangular Rods at 3 mM MgCl2.

Appendix C: DNA Sequences

Oscillator (8064)

	Staple Sequences	
1	**TTTTTTTT**ATTGGGCG	Left Edge Blocker
2	**TTTTTTTT**ATCGCACTCCAGCCAGATCGGCCTCAGGAAG**TTTTTTTT**	Left Edge Blocker
3	**TTTTTTTT**GAAGAAAGGAGTCCACTATTA**TTTTTTTT**	Left Edge Blocker
4	**TTTTTTTT**TTTAACCAATGCCGGAGAGG**TTTTTTTT**	Left Edge Blocker
5	GTTTGCGT**TTTTTTTT**	Left Edge Blocker
6	**TTTTTTTT**GTAGCTA	Left Edge Blocker
7	**TTTTTTTT**GTAATCA	Left Edge Blocker
8	AAGGAAGG**TTTTTTTT**	Left Edge Blocker
9	GCTCATTT**TTTTTTTT**	Left Edge Blocker
10	TCGAATTC**TTTTTTTT**	Left Edge Blocker
11	**TTTTTTTT**AAGAACGTGGAC	Left Edge Blocker
12	**TTTTTTTT**ATAAAGCTAAATCGGTAATAAAGCCTCAGAGC**TTTTTTTT**	Left Edge Blocker
13	ATCAATAG**TTTTTTTT**	Right Edge Blocker
14	**TTTTTTTT**AGCGCCAT	Right Edge Blocker
15	TAGAGCCG**TTTTTTTT**	Right Edge Blocker
16	AAACGTAC**TTTTTTTT**	Right Edge Blocker
17	**TTTTTTTT**TGCAACTAAAGTACGGTGTCTGGCTTTAAACAGTTCAGAAAA**TTTTTTTT**	Right Edge Blocker
18	**TTTTTTTT**CACTGGTGTGTTCAGCTCATAAACATCCCTTA**TTTTTTTT**	Right Edge Blocker
19	GGCCACCG**TTTTTTTT**	Right Edge Blocker
20	TTTAAATA**TTTTTTTT**	Right Edge Blocker
21	**TTTTTTTT**CCTCCGGCCAGAGCACGTCTCGTCGCTGGCAG**TTTTTTTT**	Right Edge Blocker
22	**TTTTTTTT**AAAATTCATATGGTTTTTGTCACA**TCGAC**	Right Edge Blocker – salt on
23	**TTTTTTTT**TCAATAGATAATACATTAATAGAT**TCGAC**	Right Edge Blocker – salt on
24	**TTTTTTTT**AGTAAAAGAGTCTGTCATCAGTGA**TCGAC**	Right Edge Blocker – salt on
25	**TTTTTTTT**CGAGAATGACCATAAATCACTAACAACTTGAGGAT**TCGAC**	Right Edge Blocker – salt on
26	**TTTTTTTT**AAAATTCATATGGTTTTTGTCACA	Right Edge Blocker – salt off

27	**TTTTTTTT**TCAATAGATAATACATTAATAGAT	Right Edge Blocker – salt off
28	**TTTTTTTT**AGTAAAAGAGTCTGTCATCAGTGA	Right Edge Blocker – salt off
29	**TTTTTTTT**CGAGAATGACCATAAATCACTAACAACTTGAGGAT	Right Edge Blocker – salt off
30	ATCAATAGATCGCACTCCAGCCAGATCGGCCT	Polymerization
31	AAGTACGGTGTCTGGCTTTAAACAGTTCAGAAAAGTAGCTA	Polymerization
32	TAGAGCCGTTTAACCAATGCC	Polymerization
33	AAACGTACAAGAACGTGGAC	Polymerization
34	GTTTGCGTCCTCCGGCCAGAGCACGTCTCGTCGCTGGCAGATTGGGCG	Polymerization
35	TCGAATTCCACTGGTGTGTTCAGCTCATAAACATCCCTTAGTAATCA	Polymerization
36	GGCCACCGGAAGAAAGGAGTCCACTATTAAGCGCCAT	Polymerization
37	TTTAAATAATAAAGCTAAATCGGTAATAAAGCCTCAGAGCTGCAACTA	Polymerization
38	CAGGAAGAAAATTCATATGGTTTTTGTCACA**TCGAC**	Polymerization – salt on
39	GCTCATTTTCAATAGATAATACATTAATAGAT**TCGAC**	Polymerization – salt on
40	AAGGAAGGAGTAAAAGAGTCTGTCATCAGTGA**TCGAC**	Polymerization – salt on
41	GGAGAGGCGAGAATGACCATAAATCACTAACAACTTGAGGAT**TCGAC**	Polymerization – salt on
42	CAGGAAGAAAATTCATATGGTTTTTGTCACA	Polymerization – salt off
43	GCTCATTTTCAATAGATAATACATTAATAGAT	Polymerization – salt off
44	AAGGAAGGAGTAAAAGAGTCTGTCATCAGTGA	Polymerization – salt off
45	GGAGAGGCGAGAATGACCATAAATCACTAACAACTTGAGGAT	Polymerization – salt off
46	ACGAGCTTGAGATGGTTTAATATACCAGAGTTGAGA	Core
47	GATACTTCATGAGGATGTATCATTAATCTTG	Core
48	ATGGTCAAGTCAGGATGACTTCAAATATCGCGCCTCAAATTACCAGAC	Core
49	AGACCGGATGCGGCCAGAATGCGGGAGGTG	Core
50	TTCGGTCGGGCGGATAAGTGCCGTAGAAGGATTAGGATTAATTT	Core
51	GAATCCTGAGAAGTGTCCCGGAATTCACCGGAAACG	Core
52	CATAAAGGACATCACTAAGCTTTCGGATAGCTAGGGTAA	Core
53	TCGAGGTGACGCGACCTG	Core
54	TGCTGTCACTGTTGCCCTGCTGCAGCCAGCGGTGCTTTTGCG	Core
55	AGGGAGGGAAGGAATTCAGAAGCACGGAATCGTTCTGCGA	Core
56	CGCACAGGCGGCCTTTAACCGCAAAACGACGG	Core
57	GCGGGAGCCGCAGAACGTTGTAAGAATGCCAATCCGCCG	Core

58	GAGAATCGCCATATTTATAATTACCCAGTAAG	Core
59	GGGTTACCGGCTGGTAATGGGTAGTGCCGGACTTGTAGATTGC	Core
60	ACGGGTAGAGGCTTTGAGGAGGTTTAGT	Core
61	GAGACTCCTCAAGCGAGAGGGTTGAT	Core
62	TCGAATCCTGAATCTTACCAGCCAGTTCCATCTTT	Core
63	CGCCCACGCATAATATCAGCTTGCTT	Core
64	ATAAGGCGTTAAACTCAGTACCACTGAGGCT	Core
65	AATAGTGAAATCTCCATACCTTTTTAATGGATAAAACAGAGAT GATG	Core
66	GAGACTACGTAAATGCAATATATGTGAGTGAATAACC	Core
67	TGTTAAGGCTTATCCGGTATTAGACGGGTAAGCCC	Core
68	ATATTTTAGCGAGCTGATGTACCCCGGCGAGA	Core
69	GTAACATTATCATTTT	Core
70	GACGAGCACGTATAAGCTCATTTTTGGGTAAATCCCACGCAGCG GGG	Core
71	TACCTGAGATTCGCCGGCATAGTAAGAGCAACACTATCA	Core
72	CACCGTCACCGACTTG	Core
73	CGAACCTCGAGAGAAAATAGCAA	Core
74	CGCGTAACCACCACACAATCCCTTCATCACCCTCCA	Core
75	ACAGACCAGGCGCATAGGCTGGCTAATTACGATGATTGCT	Core
76	TAGGGCGCTGGCAAGATAGGGTTGTCTATCATTAATGAAAATTG TTA	Core
77	ATAAGTATAACAACAACC	Core
78	TCCAGCATCAACCAGCTTACGGCTCCAGGGCGGTTG	Core
79	GTCGTACGAGCCGGAAGCATCTGCCAGCACGCGTGTTATTTCA	Core
80	GAACAAGGTTTCTTTGCTCGAAATCGTTAACGGCATATATAATG CTGTAGCTCAACATGT	Core
81	AAGACACCCGTTGTAGGATAACCTTGTGAGACCGGCAAACGCGG TCC	Core
82	AGGTAAGTAATTCTGTCCAGACGACGAAAGCGGAA	Core
83	CACCAGAACGAGTAGTATTCAGTGAATAAGGC	Core
84	TAAAGGGATTTTAGATGGTGAAGAGAGGTGGGTGG	Core
85	GGTATTTGAAGCCTTAAATCTAACGTCATTAAGAAA	Core
86	TCATTGCAGTGCACTCTGTGGTGCAGCAAACTCCAACAGTAAC	Core
87	ACCGAACTGACCAACACCAGAAGACGTCAG	Core
88	AAAACACTCATCTTTGACCCCCAGCGATTATCACT	Core
89	AGTTAAACGATGCTGAACGTCAGCAGCCGCC	Core
90	AGGTTGAGGAGCCGCCGAGCCACC	Core
91	CGACAATGGCCCGGAATAGGTGTATCTGAAACATGAAAGTAAAA	Core
92	GCTATTTTGCATCCCAATCCAAATAA	Core
93	AGTATCATACAGTAGGGCCTTGATATTCACAAACAA	Core
94	CTAATATCAG	Core

95	ACCGCCACCTTTCCTTATCATTCCCAATAAACGTTTTAG	Core
96	CGTATCAAACTTAAATTTCTCGTGCTTTCAGAAGGA	Core
97	ACAAGAACTCAACTAATGCGATT	Core
98	CACCGCTTGGAAAGCCGAAAAACCGAGTGTTGTTTCACC	Core
99	TTAGAAGTATTAGACGAATCCCCGTCTTTACATTGCTGACAGAT GCC	Core
100	AAATTGTGTCGAAATCATTTCTTAATTTCAAT	Core
101	TATTTAACGGGTATTAAACAATGATAACATAA	Core
102	CGCGCGGGGTTTTTCTTCCAGTTTGGAACAACGAAAGGATGGCG AGA	Core
103	TCGCTATTAATTAATTTTCCATCCTGATACCATATC	Core
104	AGTACCGACATTAAGAGG	Core
105	TCATCGGTGCCCCCTGCATCTTGCGGTATGAGCCGGGTCTGGT	Core
106	CAAATATTTAAATTGTACAAGAGAGGAGACA	Core
107	TTTTCCCAAAAGTTACCCTCGTTAGAATCAGA	Core
108	CTGTGTTTTGCCAGAGGGGGCCCTCGTTATCAAACC	Core
109	TTTTTGAATG	Core
110	AGAAGCCTCCTGTTCTTCGCGTCCCAATTCCA	Core
111	GGAACAACAT	Core
112	ACCCGCGCAGAGGCGATCAAGAGCGCCTGAT	Core
113	ATCCATTCACGTTGAAATTTATCACATAGCG	Core
114	AAAAGCCCCAAAAACATAGCATGTGTAGGTAATTTT	Core
115	GGAGAAACAATAACGGCAAAAGAA	Core
116	GGCGCGGAGACGATCCAGCGCAACCTTTAA	Core
117	CATTTTGAGGATCCCCGGGTACCGAGC	Core
118	TGGTCATAGCTGTTTCCTGTGTGATCGGCCAA	Core
119	ATAAACACCGGAATCAACAACGCTGACAGG	Core
120	CAAAGACATAGGAGCAAAAATCAGCTCAAATGAAGTTTCA	Core
121	CGTAACACTCAGAACCGCCACCAGCATGTAGAAACCAACGCC	Core
122	CCTATTATTCACCGTACTCAGGACTAAAGACTTTCGATAGTTGC GC	Core
123	AGTACAACGGAGATTAGTTTCCATGTGAATT	Core
124	AGCGAAATCAACGTAACAAATTCATCTCAGGACG	Core
125	CGCCATTCTAAAGGGACGTGAACATAAATCACCCTTCACCGCCT GGCCCT	Core
126	CGGATTGACGGATTCTAGTAATGTCAATCATAAAAGGTGGCATC AAT	Core
127	AAAAAAAGGTTAAAGGCCGCCGATGTGATAA	Core
128	TAGAGTCATACCGGGGGTTTAAAGTGTAAAGCCTGGTGCGTTG	Core
129	CTGGTAATATCCAGAA	Core
130	TGATGAAACAAGTACCTTTTACATCG	Core
131	TTTGAAATACCGACCGATAATAATTTTATCGCAAGACAAA	Core

132	TTAAAATTCGCATTAATATCAGGTTCAACCGTAAAA	Core
133	TCCGCTCAGTGAGCCTCCTCACAGATGACCCTGTAATACTCAGG	Core
134	CTCAATCAAGGTGAATTGCAACAGCCTTATTA	Core
135	AAAAGAACAAACGCAACGATTAAGGCCGCCAGCAGTTGCA	Core
136	AATAGCAAGCAAATCTTATTTTCATCGTAGG	Core
137	ATTGAAATTAATTACATTTACAGGTTTAGAGCGGAA	Core
138	AGTAAGCAGATAGCCG	Core
139	ATAGCTTATGATTATCAAATAAAG	Core
140	GAGAGCCCGCTTAAATCAAGTTTTTTGGGGTCGTGGTGATGGTG GCAAGCGG	Core
141	GTACTGGTCCATTGCAACAAAATA	Core
142	CCAGTTTGCGCGTCTGTATGATATCATTGCCTGCAAAGAA	Core
143	CCATGTTACTTAGCCGGAACGAGGCGCAGTTTA	Core
144	GCGCGCCTGGCGCTTTCGCACTCAACGGCAGCACCGTCGGATCC	Core
145	TGAGAGCCTGCTGAATTTTAATTCAAA	Core
146	ACGAGTAGATTTAGTTTTTGATAATGCATCAA	Core
147	TTCCATATAACAGTTGAGAGCTTACCTGACT	Core
148	AACCGAGGTGGCATGATTAAGACTTGCCACGC	Core
149	CGCAGTATTATCGGCCGTCACGAACAGCGGAAAAAAAGC	Core
150	TTCATTAAATATCTGGAGAGGAAAAAATGTTTAGATACATTTCG CAA	Core
151	CGTCATACCGGAACCAGCCAGCAT	Core
152	AGAATTAGCTATATTTTCATTTGGGGCAATGAACC	Core
153	TCCACGCTGGTTTGCCGCTAACTCGGATGTGC	Core
154	ATTAATTTTAAAAGTTTAATAGTGCCCGAAATAGAGAGTGTGTC ACT	Core
155	GAACGCGAACCTAAATTTAATGGGCCTGTTT	Core
156	CCGCCACCCTGAGTTTCGTCACCAGTAAAATACGTAATGCACCA	Core
157	GCCAGAATGGAAAGCGCAGTCTCAGAGCAGCCGCC	Core
158	CATCCAATAAATCATATTTGCGGGGTCAAATC	Core
159	TGTACATCGACATAAAAAATGGTGCCCGCCAGGG	Core
160	AGTGAGACGGGCAACAAGCTGCAGGGCGAT	Core
161	TGCAGGGAGCTCCAAAAGGAGCCACGGTCAATCATATGAGAAG AGTC	Core
162	TTAGCAAAATTAAGCTGTACCATCTAGCTG	Core
163	TGTAGATGCAACATTAGAAAGGCCATCGATGAAGCATTAA	Core
164	AATTTAGGCAGAGGCGCGGGGTTCAGGAGT	Core
165	TAATGGTGCCTAATGAGTGAGCGGGCCGTTTTCACGACCCTCAT	Core
166	CACAACAGGAAACCTGTCGTGCCGCTGA	Core
167	CAACTCGTATTAAATCCAATACTGAAGCGGATGAGG	Core
168	GCAGAACGTCAATAACATCGAGAACAAGCAAATTGAGTAGAATT AA	Core

169	CGCCGCTACAGGGCGCTCCTGTTCGAAAGGGACAT	Core
170	GAAACGATTTTTGTTAAGATTAG	Core
171	TGAGTAACAGTGCCCGTATAAACAGCCCTTTAAAATGAA	Core
172	CAATCGTCTGAAATGGATTATTTAGGTGAGGCGGTCAGTATTAACACC	Core
173	ATAAATTAATAGGAACTTAAATCA	Core
174	GTTTTTTCATCCTCATAACAATCGGCG	Core
175	CCAGGGTGGAGAGGCGTCCAACGT	Core
176	**GTCGT**AACGCTCACCGCCAGAATAAGTTTTAACGGGGTCAGTGCCT	Right OH's – Salt on
177	**GTCGT**AGCCATTTACAGTAACAACATCAAAGGACAGATGAACGGTGT	Right OH's – Salt on
178	ACTCAAACGTTAGCAAACGTAGAAGAAATTA**TCGAC**	Right OH's – Salt on
179	CAGTTGAAAGGTAAATATTGACGAATACATA**TCGAC**	Right OH's – Salt on
180	AATATCTTAAAGGGCGACATTCAGAAACGCA**TCGAC**	Right OH's – Salt on
181	TAGTAATATGGCAACATATAAAAACCGATTG**TCGAC**	Right OH's – Salt on
182	**GTCGT**CATAGCCAAAATCACATGGCTTTGAGCGTCTCAACATGT	Right OH's – Salt on
183	**GTCGT**CCAGTAGCTAGATTTTACAATTTCCGCAGGGA	Right OH's – Salt on
184	AAATTAACACGGAATAAGTTTATTACCAGCGC**TCGAC**	Right OH's - Salt
185	**GTCGT**AACGTCATTTGCACGAACAGTACCTTGAAAAAAATCATAGGTCTGA	Right OH's – Salt on
186	**GTCGT**GACTGTAGCCGCCTCCCTCTGAATGACGATTGGCTTAATT	Right OH's – Salt on
187	**GTCGT**CAATATTATGGAAATACCTACATTTTGACGCT	Right OH's – Salt on
188	TTGCTCCTTGACCATTAGACTGGATAGCGTCCTTTGCCCCAAATCAA**TCGAC**	Right OH's – Salt on
189	CAGCAGCAGTGATGACTCACGGAAAAAGAGATAAACAGGTAGAAGA**TCGAC**	Right OH's – Salt on
190	**GTCGT**ATGAATATGGGAATTAGAGCCAGCACAGGAAA	Right OH's – Salt on
191	**GTCGT**AAATTGCGACCATTACCATTAGCATTTTCGGT	Right OH's – Salt on
192	**GTCGT**TCATAATCCCCTTATTAGCGTTTGAAAATCA	Right OH's – Salt on
193	GATGGCTTATTCCCAATCATAAATATTCATTTTTACAAATTATCTAA**TCGAC**	Right OH's – Salt on
194	CGTTGATAGACTTTCTCCGCAGGAACGTCTTTGAT**TCGAC**	Right OH's – Salt on
195	**GTCGT**AAAATTACCAATGAAACCATCGATAGCGTCA	Right OH's – Salt on
196	**GTCGT**ACCGGAACGCGTTTTCATCGGCAAGGCCGGA	Right OH's – Salt on

197	GTTTACCAGTTTTTATACATCACGC**TCGAC**	Right OH's – Salt on
198	**GTCGT**AACAGCCATATTATTTACCCAGCTACAATTTTGCCAGTA ATAAGAGAATATAA	Right OH's – Salt on
199	CCAGTGCCTGCCTGAGAGGCCGAT**TCGAC**	Right OH's – Salt on
200	AAAGATTATCAGTTGGGAACGTT**TCGAC**	Right OH's – Salt on
201	**GTCGT**GCGGAACAAAGAAACCTTTGAAAGGAAAACATATCGGT TCCGATATA	Right OH's – Salt on
202	ACGGGAACGCAATACTGTACGCCA**TCGAC**	Right OH's – Salt on
203	**GTCGT**TTATCATCATATTCCGATTAAGAATTTGAAT	Right OH's – Salt on
204	**GTCGT**TTCCAGAGCCTAATTTACGCTAACTGATGATATTGTAAG A	Right OH's – Salt on
205	ATTATAGTGAGGAAGGCAATTCGA**TCGAC**	Right OH's – Salt on
206	**GTCGT**GCAATTCATCAATATACTTAGAATCATAAATCTGATGCA A	Right OH's – Salt on
207	**GTCGT**ACCAGAACCACCACCAGCAGGTCATTACCGTTTAGAAAA A	Right OH's – Salt on
208	AACGCTCACCGCCAGAATAAGTTTTAACGGGGTCAGTGCCT	Right OH's – Salt off
209	AGCCATTTACAGTAACAACATCAAAGGACAGATGAACGGTGT	Right OH's – Salt off
210	ACTCAAACGTTAGCAAACGTAGAAGAAATTA	Right OH's – Salt off
211	CAGTTGAAAGGTAAATATTGACGAATACATA	Right OH's – Salt off
212	AATATCTTAAAGGGCGACATTCAGAAACGCA	Right OH's – Salt off
213	TAGTAATATGGCAACATATAAAAACCGATTG	Right OH's – Salt off
214	CATAGCCAAAATCACATGGCTTTGAGCGTCTCAACATGT	Right OH's – Salt off
215	CCAGTAGCTAGATTTTACAATTTCCGCAGGGA	Right OH's – Salt off
216	AAATTAACACGGAATAAGTTTATTACCAGCGC	Right OH's – Salt off
217	AACGTCATTTGCACGAACAGTACCTTGAAAAAAATCATAGGTCT GA	Right OH's – Salt off
218	GACTGTAGCCGCCTCCCTCTGAATGACGATTGGCTTAATT	Right OH's – Salt off
219	CAATATTATGGAAATACCTACATTTTGACGCT	Right OH's – Salt off
220	TTGCTCCTTGACCATTAGACTGGATAGCGTCCTTTGCCCCAAATC AA	Right OH's – Salt off
221	CAGCAGCAGTGATGACTCACGGAAAAAGAGATAAACAGGTAGA AGA	Right OH's – Salt off

222	ATGAATATGGGAATTAGAGCCAGCACAGGAAA	Right OH's – Salt off
223	AAATTGCGACCATTACCATTAGCATTTTCGGT	Right OH's – Salt off
224	TCATAATCCCCTTATTAGCGTTTGAAAATCA	Right OH's – Salt off
225	GATGGCTTATTCCCAATCATAAATATTCATTTTTACAAATTATCT AA	Right OH's – Salt off
226	CGTTGATAGACTTTCTCCGCAGGAACGTCTTTGAT	Right OH's – Salt off
227	AAAATTACCAATGAAACCATCGATAGCGTCA	Right OH's – Salt off
228	ACCGGAACGCGTTTTCATCGGCAAGGCCGGA	Right OH's – Salt off
229	GTTTACCAGTTTTTATACATCACGC	Right OH's – Salt off
230	AACAGCCATATTATTTACCCAGCTACAATTTTGCCAGTAATAAGA GAATATAA	Right OH's – Salt off
231	CCAGTGCCTGCCTGAGAGGCCGAT	Right OH's – Salt off
232	AAAGATTATCAGTTGGGAACGTT	Right OH's – Salt off
233	GCGGAACAAAGAAACCTTTGAAAGGAAAACATATCGGTTCCGAT ATA	Right OH's – Salt off
234	ACGGGAACGCAATACTGTACGCCA	Right OH's – Salt off
235	TTATCATCATATTCCGATTAAGAATTTGAAT	Right OH's – Salt off
236	TTCCAGAGCCTAATTTACGCTAACTGATGATATTGTAAGA	Right OH's – Salt off
237	ATTATAGTGAGGAAGGCAATTCGA	Right OH's – Salt off
238	GCAATTCATCAATATACTTAGAATCATAAATCTGATGCAA	Right OH's – Salt off
239	ACCAGAACCACCACCAGCAGGTCATTACCGTTTAGAAAAA	Right OH's – Salt off
240	CATCACCTAGCAGCAAATGAAAAGAATACCC	Left OH's - Low Affinity
241	TTACGCCGATCGGTGCGGGCCTACAAACGG	Left OH's - Low Affinity
242	ACCACCAGCAAGTTACAAAATTTATGCAGATA	Left OH's - Low Affinity
243	GATTCACCTTACCGAAGTTAATGCCGGG	Left OH's - Low Affinity
244	GGACATTCGAGCAAGAAACCAAGTCGCGAGGCAACATGTTCAGC TAAT	Left OH's - Low Affinity
245	CGGAACCCGCCATTCAGGCTGCGCCACGTTGG	Left OH's - Low Affinity
246	CAACCCGTCCGTAATGGGATAGGTAACTGTTG	Left OH's - Low Affinity

247	CTAAAACATGGCTCATTTTCAACTCATTACCCCGAAACAA	Left OH's - Low Affinity
248	TGACCTGACCCACAAGAGCCGTTTAGAT	Left OH's - Low Affinity
249	CTTGACGGCTGGTGCCGGAAACCTGCATCTG	Left OH's - Low Affinity
250	GCTTTCATGGCGCATCGTAACCGAGGCAAAG	Left OH's - Low Affinity
251	GACGATAAAAACCAAAATAGTTGATCTAAAG	Left OH's - Low Affinity
252	TTGAATACCAGAAGATAAAACAGACATTGGCA	Left OH's - Low Affinity
253	TAGCTATCAGTCACACGACCAGTCCGAACGA	Left OH's - Low Affinity
254	GGCGAAAAGTACTATGCTTCGCTA	Left OH's - Low Affinity
255	CGCTCACTGAGTTGCAGTTCCGAAATCGGCAACCGCCGCGCACT AAAT	Left OH's - Low Affinity
256	ACGCAAGGTAGTAGTACGGTAATCGTAAAACGGAAGATTCGAG TAA	Left OH's - Low Affinity
257	TTAAGAACTCGCCATTAAAAATAAATAAAAG	Left OH's - Low Affinity
258	AATAATAATGGCCAACAGAGATAGTGATAGCC	Left OH's - Low Affinity
259	TTGGGAAGTCTTTAATGCGCGAACAACCCTTC	Left OH's - Low Affinity
260	CAAGGAGAGTCTGGAGCAAAAACGTTATGTAGCCA	Left OH's - Low Affinity
261	TTGAAAGAATAGCCCGAGTGTAGCGGATTTAGAG	Left OH's - Low Affinity
262	TGCAAGGTAATAACGGTTGCTTT	Left OH's - Low Affinity
263	GGCTTTTGCGAGCTTCAAAGCGCAATGCCTGCCGTGGGAATAAT CAG	Left OH's - Low Affinity
264	AATAGCAGCCTTTACACCGACTTGCCCCTGCC	Left OH's - Low Affinity
265	CATAACGCCAAAAGGGACCTTCAATTATTCAACAGCTT	Left OH's - Low Affinity
266	AAACAGGGAAGCGCATTCTAAGAAACCGCACTTCGGCTGTCCTC AGAA	Left OH's - Low Affinity
267	TCTACTAAATAAAAAAGATTCAAAAGGGTGAAATGTGAGGTATA AG	Left OH's - Low Affinity
268	TTTAGGAATACCACATCGGATATTTTAATCATTTAA	Left OH's - Low Affinity
269	GGAAGGGCAGCAGGTGCCGTAAAGCTTAATG	Left OH's - Low Affinity
270	ACCATCAAGCCTTCCATATTTTG	Left OH's - Low Affinity
271	TATTACAGGTAGAAAGAGCTGCTCAAATTGGAGGCACCACCATG TAC	Left OH's - Low Affinity

272	GGCCCACTAGCCCCCGTCACGCTG	Left OH's - Low Affinity
273	AGCGGAGTGAGAAGTTAGCGTATGGGATTAGCGAAAG	Strut
274	AAACTACAACGCCTGTGGTAGCAACGGCTACA	Strut
275	TAGTAAATGAATTTTCTGTAACGATCTACAGACAG	Strut
276	CCCTCATATAGAAAGGCCCTCAGCTTGCTAAACAACT	Strut
277	ACAGCATCGGAACGAGAGCATTCCAAAGTTTT	Strut
278	TTTTGCGGGATCGTCAAACAACTAAAGGAATT	Strut
279	CTGAACACCCTGAACAAAGTCAGAGGGTAAT**TGCTAAGATGGTGGA**	Top Left Latch1 - On
280	AGAGATAAAAGCGTAAGAATACGTGGCACAG**TGCAGCGATGCGTTT**	Top Left Latch1 - On
281	GCTATTAGAAAAATCTACGTTAATAAAACGA**AAGCGGAACCTACAT**	Top Left Latch1 - On
282	ATAGTATCAACAATAGATAAGTCCTGA	Top Left Latch - Off
283	CCCATCCTAATTTACGCTCAGAGCCACCACCC	Top Left Latch - Off
284	CAAGCCCAATAGGAACACCTAAAACGAAAGA	Top Left Latch - Off
285	CTGAACACCCTGAACAAAGTCAG	Top Left Latch - Off
286	AGAGATAAAAGCGTAAGAATACG	Top Left Latch - Off
287	GCTATTAGAAAAATCTACGTTAA	Top Left Latch - Off
288	CCACCACCCTCAGCGCCACCCTCAGAACCGCCAC**ATTCTTGTGCTAAGATGGTGGA**	Top Right Latch - On
289	TCAAGTTTGCCTTTAGCAGCACCGTAATCAGTAG**TACGATATGCAGCGATGCGTTT**	Top Right Latch - On
290	AGGGTTAGAACCTTGTTTGGATTATACTTCTGA**CTAGCCAAAGCGGAACCTACAT**	Top Right Latch - On
291	AGCCAACGCTCAATGCGTTATACAA	Top Right Latch - Off
292	ATTTCATCTTCTGGAAAACTTTTTC	Top Right Latch - Off
293	ATATAACTATATCTTTTTAACCTC	Top Right Latch - Off
294	CCACCACCCTCAGCGCCACCCTCAGA	Top Right Latch - Off
295	TCAAGTTTGCCTTTAGCAGCACCGTA	Top Right Latch - Off
296	AGGGTTAGAACCTTGTTTGGATTAT	Top Right Latch - Off
297	CAAAGGGCGGCGAACGGCGGGCGC**TCCACCATCTTAGCACAAGAAT**	Bottom Left Latch - On
298	AAATAATTAGGGGACGACGACAGTCTTTCCGG**AAACGCATCGCTGCATATCGTA**	Bottom Left Latch - On

299	TTTTTGAGAGATCTACAAAGGCATTTTTGGCCATCAA**ATGTAGGTTCCGCTTTGGCTAG**	Bottom Left Latch - On
300	CAAAGGGCGGCGAACGGCGGGCGC**ATGGCGACCAGGACATTCGGGT**	Bottom Left Latch2 - On
301	AAATAATTAGGGGACGACGACAGTCTTTCCGG**GGGCTGGTAGACCGTGACACCT**	Bottom Left Latch2 - On
302	TTTTTGAGAGATCTACAAAGGCATTTTTGGCCATCAA**CTGGGACTTGAGAGAGATCCGA**	Bottom Left Latch2 - On
303	CAAAGGGCGGCGAACGGCGGGCGC	Bottom Left Latch - Off
304	AAATAATTAGGGGACGACGACAGTCTTTCCGG	Bottom Left Latch - Off
305	TTTTTGAGAGATCTACAAAGGCATTTTTGGCCATCAA	Bottom Left Latch - Off
306	CTGAACACCCTGAACAAAGTCAGAGGGTAAT**TGTCCTGGTCGCCAT**	Top Left Latch2 - On
307	AGAGATAAAAGCGTAAGAATACGTGGCACAG**ACGGTCTACCAGCCC**	Top Left Latch2 - On
308	GCTATTAGAAAAATCTACGTTAATAAAACGA**TCTCTCAAGTCCCAG**	Top Left Latch2 - On
309	ACCCGAATGTCCTGGTCGCCAT	Input1
310	AGGTGTCACGGTCTACCAGCCC	Input2
311	TCGGATCTCTCTCAAGTCCCAG	Input3
312	ATTCTTGTGCTAAGATGGTGGA	Latch1 Release
313	TACGATATGCAGCGATGCGTTT	Latch1 Release
314	CTAGCCAAAGCGGAACCTACAT	Latch1 Release

18hb Symmetrical Rod (7249)

	Staple Sequences
1	CAGACTGAACCCAAAAGCATATGGTTTACCAAACCATCCTCCCTC
2	GCCAAGCTGCCGGACTGAGAACCGTCGAACAAACTTTCTTAA
3	GCGGGAGTGCTGAATTCAAAGCGAACCATGCAGATAACA
4	CAGTATTTACCGCCTTTTTAAGAATAACATAAGAACTGTTTA
5	CCGCATTTTCACCGATATATTCGGTTTGCGGGAAGAACC
6	TCGCGTGTGATATACAAATTCTTACTTACGAGGTTTTTA
7	ACGATACCGCATCGGCTGTCTTTCCAAAAAGCTAAACAC
8	CTCAGAGCCTCATTCCCACAACTACAATTAAGAACGTTCAGC
9	GGAATACACCCTGAGTAAGCGTCAATCGGCCATTAGTCAGTT
10	CGCGTTCAACCAAAAACATTATGACTCAACATATATCGC
11	TGTGTGAGTGCATCTTGATTAAACATGAAGTAAATTTTTCAC
12	GCACCGTGAAAATTAACTGGCATGATTAAGACGGGTGAT
13	TCTTATCAGATTACCAGCCGATTGAGGGAGGTAGCACCCACCACC
14	TAGGCTGCTGACGAAGGTGTAAAAGGGAGCCGCGCAAAATCC
15	GATGAAATACACCACCACAAACCCTCAATCAAACAAAGTACCATA

16	AAACATTTGAACTTCTGTTATCATCATATTCAAGCATCAGGCGGT
17	TTACGTAACCAATTGTTGCGCTCACTGCCCGTCAGGGCTTAGAGC
18	GTGAGTGAGAATATTTCAACGCAAGGCGGATGAGCAAAC
19	TCAGATAGCTAACGATTAAGACTGAGCATAAAGAAATTTTAA
20	GTACATTCCACCAGCTTGCTTTCGATTTTCATCTGACCA
21	ATACGGCCTCAGCTCGAGCTGCATTAATGAAAAAGAACAAGGGAA
22	AGATTTTTTCATTTTTCACCAGAATGGATCAGAGCATTACCA
23	AACCAAGTTTTTTGTGTGAGTTGGAAACAATCCTGTCAGATG
24	CACTACGTTGCGTTATCCGCTCACAATTGTAGATGTGCCTGATTT
25	TTTAGAGCTAACAGGGCGGATGGTGGTTCCGATTGCCCTGGTAACG
26	CATTTATGCGGCAGACGGTAAAATACGTAATAAAGGAGTAACGAT
27	GAGCTTGAGAAAGAGGAGAGGACTAAAGACTGGTGAATACAACGC
28	GTAAGCCCCCTGTTTTCATCGGCATAGGGCAACATATAA
29	CAACTCGTAGGAGCAATATTTTTGAATGGACCAGTGCAGAGGAAT
30	AAACCAAATGCGTTAAATAAGGCGTTAACTTGCTTCAAA
31	TGTGGCTGCGTTTCCCACGGGCAACAGCTGAAATCGGCTTAATGC
32	ATCGTAAAACAGGAATAGCGTGCTGCTCTCTTGACATCGTCA
33	GTGCGAATTACAGGTTTTCCTTTGCCCGAACGAAGGTTGTCTTTA
34	ATATGCATTGTACCGTTCTAGCTGATAACGCCATCCTAT
35	GGTACCGAGGAAGAAGTCTGTTTAGGATGTTAGCGCCTTTAA
36	CTAAAGTCCTCAAGGAACTGGTAGTAAGACTTCAAGTTTTAA
37	TAGTTAAAGCTTAGAGCGTCTCAGAGGGAACGCAAAGACAAA
38	AGTAATTAATATAACTATATGTAAATGCCTACCTTTCAA
39	CTGTATGACCTATTCTACGTTAATAAAACATCGCCAAAACAC
40	AGCACTAAACTCAATAGGTCACGTTGGTCCACACACTAACTC
41	CACCCTCCACCCTCTAACAAACCAATACCCTCAAAGTTTAGC
42	TGGGAATTTATTCACTTTTTAAGAAAAGGTTAAGCACAA
43	TTACTTCTGGTTGCATGGCGTATTGGGCGCCAGGGTTGGTGTAGC
44	ACATTAATGAACCAACCCTAAAGGGAGCCATCACTGGCGCATAAT
45	ATCAATAAATCAGTCAGAGCCACCACCGTGGCTTTAGAATTATTAC
46	CTTATAAAGTGAGAGTCACGACGTTGTACCATTCGCAGGAACGTA
47	AACAAGACAGTAGGTTCTGACCTAAATTAATCCTTTTGC
48	AAATCCGCCACAGCAAAATCACCAGGAAGGTAATAGCCG
49	GGGAGAGTTGGAACCGGGCGCTAGGGCGTATAATCGGCACCGAAT
50	CCTACAATATAACACCGAAATGAAAAATCTACTGATTAATTGTTT
51	AAGAAACGTAGCGCTATTAGCGTTTGCCCGGGGTCACAGAGAGAA
52	TTATACAAAGATTTAGGCAGAGGCAACAACATGCGAGGC
53	GGGTAGCTCATTTTAGACGACCTTTAATAACCGAAGAGGAAG
54	TTGCCGAACCTGACGACGACAATAATTTTCGAAATCCAA
55	ACCAACCATCTCCATCTTTCCAGACGTTAAGTATTCAGGACGGCC
56	GGTCACGCAGAATCAACCAGGATTCGCAGGCTATCAGGCAAA
57	CTGTTTACATAACTGATTGAAAGGAATTGAGGTTATTAATTGCGT
58	CCCCAAAAACTAGCTGAAAAGGTGGCATATAACCTTGCTTTA

59	TGAGAGCTTAAAGCCTGGGGTGCCTAATTTTTTTGGGGTCGC
60	AACAAAGGAGATAAAAAGCCAGTCACACGCTATTAATCTAAA
61	CCTTTTATGCTTTGAACAGAGTGGCCTTGAGCCGCAGCCATT
62	TCATCTTACAACTAGCTAAACAACTTTCATGCCCCACGGAACGAG
63	GAACCCTTTTTGATATTAGAGAGTACCTAAGATTCGGGAACA
64	TTGTGTCGAAAAATATTCTGAGTAATAACCCCGATGATGGCC
65	GTTTTAGTATTTTGATAGGTCAGTTACAATACAGTAATTCGA
66	TAAGAATAACGTTAGCACACCACGGAATAAGGTTTGCCCATAATC
67	ATGCGCGTGGCAGACAATTACCGCTGAGTTTCAAATTTAACA
68	CTGGCCCCCTGTTTCGTACTATGGTTGCTAGAATCGCCTCTT
69	TAGCACACTGAATAGTTGCGCCGACCGAGGGTAGGCGCA
70	GGAGCCGGAACAGCGACAGAATCAATTTATTTTTATTAC
71	ATATCTTTATTAAAAACGTCAGATGAATAAATCGCAATAAAAAAT
72	TCAAAATATCAAGATTGACGCTCATACAGAACCGCGATAGCA
73	TCAAAAAGTTTACCTTAACCACCAGCCACTAGAGGACGCGCG
74	TCCAACAAGGAATAACAACCCGTCGGATTGTAGGTATTTTTA
75	ACAGCTTAGGCTTTCAGATGAACGGTGTAGTAGTAGATA
76	ACGAGGCATTTTAAAGAAGGACCATCACAGAAAGGGTGGACT
77	GCAGTATATAAAAAGTAATAAGCAACAGAGAGGTGACCTTGC
78	CGCTCAAAAAATAAATCATTACCGCGCCGCCTAATGAAA
79	CCTCGATTAAGTACGGTGTCTGGAACAGAGCACCGGAGA
80	TACGTGGTCTGGCCAATACCATGAGAGATGATGCAGCCAGTA
81	CCAGGGTCAACTGTGCCGATTTCACCGTAATAGGAACAACCA
82	AAACGTAGGATATCCCATCCTAATCAGTATATTGAAAT
83	GTATCATTCAATAACTCATCGAGAACAACCCAATCCTGT
84	ATCTTACAAGAAACAGACGATATAGAACTGAAAGCGCCGTCA
85	ACAAACAAACTATTTGCTTATCATTTTGCGGATATCTGAAAATAC
86	GTAGTAGGATACATCGAGAATGACCATAGGGGGTAAAAT
87	CTGTAGCCAGGCGGATTTCAAGATAAAACAAAGCGTCCATAT
88	CCCTCAGCGCATAAGGGATAGCAAGCCCACTCAGGGAATAAGAAT
89	CGAAACATTACAGGCGTATAAGCCTTGCACGAGGTGCCGTAA
90	CGGAGGATTTTAAGGAATTGCGAATATACACTTGATAAA
91	CGTTGAGTCTGATCCAATAAATCATGTAGATTAATCAGG
92	CAAAAGGGCGAGTACCACATTCAACTAAGACCGGAGCTTAGA
93	AACGGATATATGCAATGCCTGAGTAATGTCTCCGTATCA
94	TACCGTACCGGAATGAAACACTGCCAGAAATCAAATAGTTTG
95	AACAGTTCAAAATTGAGAGATCTACAAATTAAATTTAGC
96	GCCGCTAACAGGAGTGGGAAGATAAGCAATCGATGACTAATA
97	ACCGACCTATTAATGCCATATGCGCATTAGACTCCTGTCACA
98	ACGCCAAGAACGCGCTTATCCGGTATTCTTTATCCAATA
99	GAATTAGGATTCCCATTATAGTCAGAAGACCAAAATTTG
100	ATTTGGTCAGGAAGAGGTCATTTTTGATAAAAAAAGATT
101	TGACCGTAGCATAAAGTGTCATCAATATAGTACATATCCAGATGA

102	AAGTTTGCAACAGTAGCCCTAAAACATCTCTGAAAATGATGATAG
103	ACGGAAATAGAGCCCAGAACCACCACCAGATATTCCCAATAATAG
104	TCAGACTGCAAAGAAACGTAGAAAATACAGCCTTTAGTG
105	TTTCACCATCAAAATAACCACCACACCCTTTTAGACCATTCAAAA
106	GGCAAATAGTAACAACGTAAAACAGAAAAAAGAAGTGGATTAAAT
107	TTTCCATGGTTTATAGACAGCCCTCATATAGCGGGAATTACCAAC
108	CCAACGTCGTGCCAATTCGTAATCATGGGACGACGAACCGTTTGA
109	ACTTTGATGGTTTAATAAGTGGTGTTTTCTGGCAAAGTGTTG
110	GGCGTAGCAAGGAAAGCCGAAAAACCGTCTACTTTCCATGTTTCC
111	TTAGCAACATTCAAAAGGAAACCGAGGATAATTGAGTCT
112	ACAGGGACATCACAGACACTAACAACTAATATACAAACAACAGTA
113	AACAGTTAGACTGGAGATTGTGGCGATCTAAGTTGTCACCGC
114	GATTAAGAGCCGAAGCCTTAAAGGTGAATTACGACTTGCGCCAGC
115	TCTTTACTGCAAAATGTTAAACAAAGCGAAACGACTTTTTCT
116	AGAGCCGCGTTCCAACAAAGTTTCCAGACAATAGCAGTCCTG
117	GCTACAGGATACCGGTTTCGTCACCAGTGAGGGTTAATTGGGAGG
118	CAACTAGAAGGCCTGTTTATCAACACGCCATATATATATTT
119	GAAAGGATGACCCCACGGAGATTTGTATCGAACTACTGC
120	CTAGTAGAAGAATCGGATCACCCAAATCAAGGAGTGAGACATACG
121	GTTAACATTAAAGTACACAGCGATTATACCATTGGAGTGAGAATA
122	CGAAAATTGAGAGATGCTGCAAGGCGATGGTGCGGAGAGCGGGTA
123	TTACCCACCCTAACGTCACCAATGAGCGCCAATAATAAC
124	ACCATTACATTAACGAGCAAACAAGAGAAATATTTATAG
125	AGCCGGAAATGGGAACTATCGACAGTTAAACAGTTTCAGCTC
126	TCGCAAGCAAAATCCACCCAGGAATTGATAAGCAGAATATTG
127	CTTTCCTGACTAATTCTGCGAACGAACAGGCAAGGTCAT
128	GTTGAAATAAAACGGCGACCTGCTCCATTACCAGTAAGA
129	AAAATTCGAGCTATAATGCTGTAGCCCTGTAACCATCAA
130	TGCCCACCGAAAAGGAGAAGAGTCCACTATTTCGGCCAATCCCCG
131	CGATTTTCATCGCTTAATTGAGAATATAGATAAAGCAAA
132	CGAACGACTACATTAAACAAACCCTTAGTAATGGTAAGCCAA
133	GGTCATACAGTGCCTAAATGAAAATAGCATACATAAAGGTCG
134	TGCCTGAAATATTTGAAGTTTCAGAACGACAGACCAGCAACG
135	TTCCAGTGCGGTTTCCTGCAGGTCGACTGCTTTCCAGTGAGGTCA
136	TAAGCTTGCCGCTGACCAGACAGCATCGGAAAATGACAACCCATG
137	TTGCTCCCATATATTTTAAAATTACTAGTTATCATTCCAAAA
138	TCGCCCACAGCGAATTCATCAAGAGTAAATTCAGTAGG
139	GTTTCAGAAAATTCGCAAATGGTCACAATTCTAACGGTA
140	TTGACGGTACTTCTTGCCAGTAGCCAGCCAAATCATACTTTT
141	TTTTCATATAAACATAATTTTATTAATTTAGAACCAAACCAC
142	TATGATATCTGGCCCGAGGCACTCATTAGTTACTTCGAAGGC
143	TTGTATCTAAACGGGTCAATCATAAGGGCATTGTGGTTT
144	ACCGCCTGATCATCGGGGAAGTATTAGACTTGATTAGAGTAAGAA

145	GACTTTTGTCGAAAAAAAGGCTCCAGCCACTAAGCCGGA
146	AACTTGGGAAGAAATCCAAAGAGGCAAAAGAAATAATTGAATTTT
147	GTTTTAAGGAATTATTCCTGTTTGAGGGTCATAGCGTCGGGA
148	AAATCGGACTAAAGAGGAAGCCCGAAAGAGCAACAAAAA
149	TAAGGTACGCCTGCGCGGAATAGCCCGAGATAGGGTGGGGCCAGT
150	GCTTAATAAGCCTTAGGCCGGAGACAGTTTTCATCACAT
151	CGGAATCCAATATATTTAACGTCAAAGATTGAACGGGTATTA
152	CAGAAGGCAAATATGCAGAAGATAAAACGAAAAACACAATTTTCG
153	AAAATCATGTACTGCAGGGAATATTTATGCAAGCCCATGTAG
154	GGATTATATTACCTAGCCATTGTTTTAAATCTTTTTTTAGCG
155	CAGCTATTTTTAAGCAATAAAGCCTGTTTCATGATTGCA
156	AGGGCGAGGCCGGACAGAACCGCCACCCAAGCGCAGCGCTAATAC
157	AACCTGTCAAAGGGCGGCGAACGTGGCGGCAAATTACAGTATATT
158	TGAACCTAGCGGAAAATAATGGAAGGGTACATTTAGCTCATGACA
159	TAATGCACATGTAAACGCGAGAAAACTTAAGAGTCTGAA
160	ATAAGAGCTGTCCACCCGACTTGCGGGACCTTAAATTTA
161	GAAAGCGGTAAAAGTCGCACTATAGGAAATTAATGTAAAGCT
162	ATGGCAACAGCAGCCCTGCAACAGTGCCTGGTAATAACGGAT

18hb Rectangular Rod (8064)

	Staple Sequences
1	ATAGCAGCTAAAGCCAATCCTGAATCTTACCACATATTATAAAGGTG
2	CACTATTAAAGAACGTGGTAATAAGAGCAAG
3	TATTCTAATATAAAGTACCGACAATTGCTGAA
4	AACAGTTTCAGCGGAGATAGCCCGGAATAGGTGTATCACTTTTGCTC
5	AGCCACCAGAGGTTGAGGCAGGTCAGTACCAGATGTGAGT
6	CATCCTAATTTACGCAATACTTCTTTGATTA
7	GTTGAGTGGAACAACATATAGTCAGAAGCAAATGAATATAATGCTGTA
8	AATTTAGGTTGCGGGAGGTTTTGAGCAGCCTTAGAACTGG
9	????ACCTATTAGA?????
10	TGCGGGATCAAACTAAGTGTACTGGTAATAA
11	ACCAGCAGAAGAAAAAAGCGCAG
12	CACCCTCACTCCTCAAAAATC
13	TTGAGGAAGGTTATCATCAATATGCGGATAA
14	AGCCAGCTTTGACCGTAGATTCAAAAGGGTGA
15	GCAACATGCTGGCTGAACTTTGAAAGAGGAC
16	GCGGCCAGTAATTGCTTGCCTGAGTAATGAAAATCTCCAA
17	ACCATTAGATCACCGGAACCAGAGCAGAATGG
18	TTGCCCCAGCTGCATTTTGCTTTGACAATATT
19	GAAACCAGCGTCGGAGTCAAATCACCATCAA
20	CGTGCCGGACTTGTAGTGCAGCCAGCGGTGCCTGCCTAATCTGACTATTTATTACA
21	TCGCTATTGGATAGCTCGGAACAAGAATGGCTATT

22	TTGTTCCAAGTGTAAATACGCCAG
23	GGGGTTTCCCAGAACCACCGTAAAGTTAAACGAAAAAAG
24	CTCGTTAGCAAACTATCGGCCTTGCACAGACAAGAATGCCTTCTCCG
25	ACCGTGCATCTGGTTGATATAAGT
26	??????GAGCTGGCGC??????
27	TTTAAGAACCGAGCTCGCAAGCAAGTCCAGACGAC
28	TGCCCCTGAAAGTATTCCTGATTATCAGTCAGATGAAGTTGGG
29	AGCTAATGCCCTACATGGCACTATATGAGAATAAAC
30	TTCCGGCCATAAAAAAATCCCGGCGGTCC
31	TCATTTGGGAAAAGGTGGCATCAACCGGTTGATAATCAGA
32	CGCTAGGGCGTAGCGGTCACGCTGCCAATCGTCTGAAATGGTTGGCAGA
33	CCTATTAAGCCACCAGTAACGATCTAAAGTACGCATAAAATCAAA
34	CGACAACCAGTTGGGCCTCCGGC
35	???????????CCCAAAAAGC??????
36	ATCCAATCCTGAGAAGTCCAAAAAACCCAAGAGTC
37	AAACAGGAAGATTGTATAAGCAAA
38	GAACGTGCTTTCCAGTCGGGACTGGTGTTATAACAG
39	CACCATTCCATTAAACAACAACCATCGCCC
40	GGTAGAAAGATTCATCTCAAAAGCTAACTCAGGCCGATTGTAATAACATCACTTG
41	CAGCTTGATACCGATATTTCTGTATGGGATTTGGAGGTTTAGTACCGC
42	TATCATCACGTAAGAATACGTGGCTGGTAATCCC
43	AGAGGGTATCTGGCCTTCCTGTA
44	TTTATTTTAACGCAAA
45	TAGCAAACCAAGAACCGACGGTCAATCATAAGGAGGCTTT
46	CCCTCATTTTCAGGGATAGCA
47	TGCCAGCACGCGTGCCTGAGGATCACCAAGCGTGAGATGGTTTAATTTCAACTTTA
48	CGAGCCGGGCATCAAACTACGTTAATAAAACGAACTAACG
49	ACCGGAATACCAGTAACCGTAA
50	TAGCGCGTTTTCATCGGCATTTATACAGGCAACGCCTGAACCCA
51	TTACCAGAAGGAAACCGAGGAAA
52	AGCTTAGATCCAATCGAATTTACCCTTTTCAT
53	GAATAACATAAAAACATAGCGAACCATTAAAGTTGGGAATCGAAGGCA
54	TTTCCAGAGCC??????????TAATTTGCCAGTTA
55	TAAAACGATATTTAAAGTCAATCATATGTACCTTCTACTATCATTGCA
56	CAGGTCAGGATTAGAGGAGCTTCAGTCATAGCATAGCTAT
57	AATATCTGGTCAATTTCATCTTCT
58	AATCTTGAGTAGAAAACCAGCGCCACAATTTTACGCTCAA
59	TGCCGTTCCTGCAACATAGATAATACATTTGAACAAAATTCGCACTCC
60	CAGTTGTTCTTCGCGTCCGCACCGACTTGAGCCATGTGAATT
61	GAGAAGCCTTTATTTCTATGATATGCTCATTT
62	GATAGACTAACGGCAGCAATAAAGTTGATTC
63	TATATTTTGTTTTTTCTGCACTCTTCAACAAT
64	ATGCTGATGGGCCGTCATGTTCAAGGAATCATTACCGCGCTAATATCTTACCG

65	AATTAATTTTCCCTTTTTTTCAAATTAAAGCCCACCACC
66	TCGCCTGATCATTTTGCTCACGGATGGTCTGGTTGTACCAACGGTGTCTGGA
67	ATAAAAGAGTCACAACGAGCGTC
68	TTACGGCGCTGGTAAACAGGAAAGCCGCTACAGGGCGCGGGGGAAAGAGCGGTC C
69	?????GCAGCGAAAGAAGATGAACGGT?????
70	TGCCACTATAGAGCCACTCCCTCA
71	CGGTCATGTTAAAGGGGTAGCAACGGCTACAGGAACCG
72	ATTCCACCACTCATCAGATAAG
73	CTTATAAAAGTTGAGATAAATCAAAAATCAGGAATATGCAGGGTTACC
74	AGTCTCCTGAGTAACGGCATCGCGTTGCGATGACCATTTAGGAATACCACAT
75	TATAATCAGTGAGGCCACCGAGTAGTAGAGCATGTAGAAACCAATC
76	ACCCTCAGTTCTGAAAAGTCTCTGCAAGACAATACCGACCGTGTGATA
77	CTTTCCAGACCCTCAGAACCGCC
78	TTCTCCGTGGGAACAAACGGCGGA
79	TAAGAGAAGAACGCGAGGCGTTTGGGAAGCGCATTAGACGGGAGAATTAACTGA
80	AATGGGATAGGTCACGTTGGTGT
81	TCGGCCTCGGCGGCCTTTAGTGATGAAGGACCAGAGCCGCCGCCAGC
82	CTGAGAAGAGTCAATAGTGAATTTTTATATATTTTGATG
83	AAAATAGCCCTAAAACATCGCCATTAAAACG
84	ACATTTCGGCCCTGCGTGGAGGTGCATTCTGG
85	GAATAACCTTGCTTCCAACAGTTGGCCTTG
86	AATAATCGGCTGTCTTTCCTTATCAT
87	CCTTTAGC?????????????????????????????????GTCAGACTG
88	CATTAATTAGATGCCACTAAAGT
89	CATAAATACCGAATCCTTTGCCCGAACAGAGGCGACATTCAGG
90	TGCACGTAAAACAGAAGTTTGGATCCTCACCGCAGCGGGG
91	ACCCTCATTCTTAAACCAACCT
92	ATAGGTCTGAGAGACTA??????????CCTTTTTAACCTCCGGC
93	TCTTTACCGAGTGAGAATAGCCC
94	GGTTTAACGATGATGGCGCCATGTCAACCAGCACAGGCAAAGTTTGAC
95	CGCAATAAGCCCTGACCGCCTGATAAATTGTG
96	TTTGTCGTAAAGCGCCATGAAAGTGAAAACATAGCGAT
97	TCATTTGATTAGAGCCGTCAAGTGCCACGCTGAGAGATAAACAATTTCACGG
98	GCTAATGCAGAACGCGAACACCGCCGGCAAACTAAAAAAA
99	AAAAACCAAAATAGCGAGAGGCTTAGTTTTGCCAGAGGGGCTATATTT
100	GAAAGGCTACATCGAACCGCTTCGATGATGAAACAAACA
101	TCCAAGAACGGGTATTAAAGTTTT
102	AACGTCAGCGTGGTGCAAAAGAGA
103	GAGGACTATTCGGTCGCTGAGGCTAGCCCTCAAAGCGTCACCTGCCTA
104	?????TTACAATTAT?????
105	ATCAGCTTGCTTTCGAGGTGAATGAACCGCCACCCTCAGCCTCAAAT
106	AACGCAAGCAGAGCACGACGATCCTAAT

107	GACAATGACGGGTAAAATACGTAATCGAAATCATTGACGGTACCCAAATACAGAG A
108	GGGCCTCTAATTCGCGGCTATTTTTGAGAGATTAAATCGGTCAGCAGCAATCGTTA
109	TTAGGTTGGGATCAAAATC
110	TGAGAATAGAAAGGAATTAATTGTGCCGCCA
111	AGGGGGATATTTTTAGTCATTGCCTGAGAGAATTAAGCACCGTCG
112	CTGCGCATCATCAACTCTAGCTGATAAATT
113	GCTATCAGACCAATAGGAACGCCATCAAAAAT
114	GTTAATATTTTGTTAAGGGTTTTCACGTACAGCAATTCATCCAACAGAGATAGAAC
115	ATGGTTTATACATACATTTATCCCAATCCAAA
116	GACACCAGCAGCAAATGAAAGCACTAACAACTAATAGAATTACCGGGGACGA
117	AAATGCAACCTTTTGATAAGAGGTCCGAAAGACGCTCACA
118	AATTGAGCGCCCAATAGAATTCGTCATACCG
119	CATTTTTCGCGCCTGGTCTCGTCGCTGGCAGCGGTTGTGCGGAGACA
120	CTTAATTGGAAACGTCAAGGGCGAGAGGCGCAGGATATTCATTACCCA
121	ATCGGTTTACTAAAACACTCATCTTTGACCAAACTCCAA
122	GAGCCGCCATATTCACAAACAAATGAGAAGGAGTCGCTATT
123	TCAACTAAGTGGTTCCCCGCTTTC
124	AACCATCGTATTAGCGGTTAAATATAAATGCTGATGCAAATTAAGACGTTTCGGAA
125	TCAGTAGCGGAACGAGCCGCTTT
126	?????CGTCACCAGTACGTCACCCTCA?????
127	???????ATCCGCCGGGCGCGGTGGCGCTTTCGCACTCA???????
128	TTAATGCGCGAACTGGTTTGAGTAACATTATTGCTTTGATCGGTGC
129	CAGAGAGATAACCCACAAGAATTTTTTATTGTGTGAAA
130	CAATATAATCCTGATTATAAAGAAGACGTTG
131	GACCTAAACAACATGTACCAGTAGAGGTAAAT
132	CCATTGCATGGGTAAAGGCCAACG
133	GATAAAAACTTAGAGCTTAATTGCGCGGATTAAGCATAA
134	TGCGGTATTTGACGCTGCGTAACCACCACACCGGGAAGAATTCACCGC
135	AGTTAAACGAGACTCACTGCGAAATCGGTGCCGTAAAGCACTAAATCG
136	GAGCCGGGTGGGCGCCAAAATGTTACGACGAT
137	CTAGAAAAAGCCTGTTT??????????AGTATCATATGC
138	CTTTAGGAAATCTAAAGCATCACCAAGGTAAAGTAATTCTATCAGATA
139	TAAGAAACACCCAGCTAAAGACAAACCAATGA
140	TAGAAGGCACACCCTGAGATAGCC
141	CCGATATAAAGACTTTTTCATGAGCCGGAACCATTCAATTATTACGGTTT
142	TGTAATACTTTTGCGGGCTCAACATGTTTTA
143	CGCCGCGTTGCGTATTCACTGTTCAAATGGTAGCATTA
144	GTTTGGAAGTCTATCAGGGCGATGGCCCACTA
145	GTTTGATGTGCAGATACTTTAAACAGTTCAGATCATTCCAGTTCAGCAAACCGCA
146	AAAACGAACGGAGATTATCACCGTTGAGCCTCCTCA
147	CATAAACATATCCAGAACGAGCACGTATAACCCTAAAGGAAAATCCT
148	GTGCCGTCGAGAGGCCAGTTTGATTTTTAATGGAAACA

149	CAGTAGGGAATAAGGCTTTGCCATGTTCCAGTTAGTTAGC
150	TCCTGAACAAGATAAAACAGAGGGACTTTACAAACAATTAAAAGAATGGTGCCG
151	CGACAGTAAGATGGGCCGAATAATAATTTTTTCACGTTGTGTAGGTAA
152	GAATGTTGCGCCGGAACCGCGCAAAATC
153	CGGAATCGAGAGCAACACTATCAT
154	AAAACATTATGACCCAATGCCGGCGCAGAAAGAAGGGCGAATACCA
155	TTTTGTTAAATCAGCTCGTGCTGCATTGTGAGA
156	TGTATCATGAGAAACACCAGAACGAGTAGTAA
157	??????AAAGATTGCA??????
158	GTAGATTTGGCAAAGAATTAGCAATCTGGAGCCCCGGAAT
159	???????GAGACGGGCAACAGCTTTTTCTTTTCACCAGT???????
160	GTTTTAAC????????????????????????????????GGGGTCAG
161	ATATTTTTAGAAACCACCAGAAGGACATCGGGTGGCGAA
162	AGGGTGGTGATTGCCCAGCGAAAGGAGCGGG
163	???????TGGAGCCGCCACGGGAGCCAAGCTTTCAGAGG???????
164	AGGCGATTAATATACAGTAACAGTACCTTTTAGCGGAAT
165	ACATCCAAATCGTAAATCGGCGAACCAGTCACATTGCGTAGATTTTCA
166	AAATTCATAACTGACCACCTTCATCAAGAGT
167	GCCAGCTTACTGTTGGCAGCGGATTTTTA
168	GCAAAGCGGCCGCCAGTCGTATTAAACGAACC
169	TTTAATGGTTTGAAAAGAACGCGAGAAAACAGAATCCTTATTAAGA
170	ATTATACCAGTCAGGACGTTGGGAAGAAAAATAAGATTAAGAGGAAGC
171	GGCTGAGAGAACCGCCACGTTAGTAAAT
172	GCTCCAAAAGGAGCCTCAACTAAAGGAATTGGCATCGTA
173	TTATCCGG
174	CCTGTTTAGTGGTGCTTTGTTATCCTTCAAAT
175	AACGTCAAAAATGAAAATAAGCCTTAAGGGAGGGA
176	AAACAATGACTGGCTCATCGCGTTTTAATTCAGTACCTTAATGCGGC
177	TCAAGAAAGGATTTAGAAGTATTATGAGGCGGTCAGTATT
178	TGCCTTGAGTAACAGTGCCCGTA
179	TTTTAAGAAAATAGCATGTTTCCTTTCATCGT
180	ATCACCCAAATCAAGTTTTTTGGAAACAGGA
181	ACCGCCAGCCTTCTGAATCCCACGTTACCAGTAAACAAGATTAAAT
182	ACGCAAATTAACCGTTAAAGAGTCTGTCCATC?
183	?????GTACAGACCAGGCGCATAG
184	TAGCGGGGCGTACTCATGCTAAACAACTTTC
185	CAGCATCGACAGAATCAAGTTTG
186	AAAGGGATTTTAGACCGTGAACCGAGATAGG
187	GAGCCCCCGATTTAGAGCTTGACTACTATGGAATGAATCGGTTTCTT
188	TAAGGCTTTAACGGAAAAATTATTCTCCCGACCAGAGGCATTTTCGAG
189	GAACAAAGATTGGGCTCGAAACAAAGTACAAAGAGGCAAAAGAATAC
190	TAACGCCAAATTCGCAGAATCGATGAACGGTATAAATCAT
191	CGCGACCCTGCTCATTCAGTGAA

192	TAAACAGTCCAATAGGTAGCATTCCACAGACTGCAGGGAAGCCCCCT
193	CGGCCAGTACGGATAATATACTTTCACACGACCAGTAATCCTACATT
194	CAAAATAAACAGCACGCTAATCAATAGA
195	CATTAGATCAATACTGCGCGGGGATTGCAGCACCGGCGAACGTGGCGAGAAAGGAA
196	GTACATAATAAAATATATTGACAGCCCTCAGA
197	GAAGAACTAATCAGAGCGGGAGCTGGTCGAGGCAAAATCC
198	GCCGCACAAGGAAGATAATTACATTTAACAATT
199	ATAGTAGTCAATAACCTGTTTAGGTAATAGT
200	CCCGGGTAAAGTAAGCAACAAAGTCAGAGGGT
201	AATCAACGAGACTCCCCGATTGAATCAAGACCATATTTAACAACGC
202	AGTTACAAAATCGCGCGTTATTAACAAACTT
203	GACACCAC????????????????????????????????GGAATAAG
204	CCAATTCTGAATCCCCTCGTGCCAGCAGGCG
205	TGTACCGTAACACTGAGTTT?????????????????????
206	GAAAAATTTCTTCAACCGTATTAAATGTGAGCGAGTAACAACC
207	GTTATACAAATTCTTCATAATTA
208	TAATCATGAAGCGAACGAATTACCTTATGCGA
209	TTACAAACCTGCTCAAATGCATAACGCCAAAAGGAATTACGAG
210	CAAGGCCGAGAATCGTTAGTTGCTATTTTGCGATTTTTTCAGTATGT
211	CCATTCGCATTATTCATTTCAATTACCTGAGC
212	ACGCCAGCAGAAACAATAACGGAT
213	AGACGATTGAAAGGAAATCAAACCCTCAATCCCAGTAA
214	AACCCTCGTGAGAGAGGAGGCGGTCTTAATGC
215	AGGAACGGGCCTGGGGGGTGCCCCCTGCATCAATCCT
216	CTGGCCCTTTACCAGTAGACTGGATAGCGTC
217	CATGATTATAACAAAGTGCTCCATGTTACTTAGGAAGTTT
218	AACGCTCATGGAAATAAAAAGGGATCCAGCATGAAACAAACTAGCATTTGTAAAC
219	ACGCTGGTGCATAGTATCATAAATATTCATTGCGAACGATGCTCGTGTGG
220	AGTACCGACAACATATGTCACTGGCGGATGGTTTTTAGAACCCTCA
221	TTCACCAGCTGAATAATGGAAGGGCCATATCAAAATTATT
222	AAGCCCTTATCATTGTCAGACCGGAAGCCCCAGCGATTAT
223	CTCATATATTTTAGTTAGTTGGCAAATTGTAAATCTTAGGAT
224	CCTCAGAGCATAAAGCCTACAAAGTGGTGAAG
225	GAGAACAAGCAAGCCGGAGTTAAGCCCAAACTCCAACGTCAAAGGGCG
226	?????CAAGTGCTGG??????

Appendix D: MATLAB Codes

Gel Analysis Code

```
% Halley megafold paper
% plotting and quantifying gel images rapid fold and kinetics

clc, clear all, close all

% gel_rgb = imread('LPP_Temp_screen1.png');
% gel_gray = rgb2gray(gel_rgb);
gel_gray = imread('18Rec.tif');
gel_double = im2double(gel_gray);

rect0 = [135.5100    0.5100  488.9800  103.9800];
h1 = figure(1);
imshow(gel_double)
xlabel('crop image to structure
lanes','FontSize',20,'Fontweight','bold')
gel_crop_rect = imrect(gca, rect0);
pause
rect = gel_crop_rect.getPosition;
xpos = rect(1);
ypos = rect(2);
box_width = rect(3);
box_height = rect(4);

gel_crop_im =
gel_double(ypos:(ypos+box_height),xpos:(xpos+box_width));

%% background subtraction
figure(2)
contour(gel_crop_im,'Fill','on')
set(gcf,'Position',[25 25 400 400])
xlabel('select 30 points for
background','FontSize',20,'Fontweight','bold')

fit_length = 20;
x_fit = zeros(fit_length,1);
y_fit = zeros(fit_length,1);
z_fit = zeros(fit_length,1);

% uncomment to pick points from image
for i=1:30
    [x_pt, y_pt] = ginput(1);
    x_fit(i) = round(x_pt);
    y_fit(i) = round(y_pt);
    z_fit(i) = gel_crop_im(y_fit(i),x_fit(i));
    hold on
    plot(x_fit(i),y_fit(i),'kx','Linewidth',2,'MarkerSize',10)
```

```
end

% coodinates for horse screen 2
% x_fit = [3 74 158 255 361 425 487 8 128 246 365 485 65 126 251 372
480 5 125 249 372 482 485 249 367 65 10 7 320 427]';
% y_fit = [97 97 100 101 102 101 101 80 79 80 80 82 58 57 57 59 61 38
40 40 41 40 24 25 25 25 22 11 15 15]';
% hold on
% plot(x_fit,y_fit,'kx','Linewidth',2,'MarkerSize',10)
% z_fit = [0.1176 0.1373 0.1569 0.1686 0.1529 0.1255 0.1020 0.1216
0.1529 0.1647 0.1490 0.1137 0.1333 0.1451 0.1333 0.1216 0.1451 0.1059
0.1294 0.1176 0.1020 0.0980 0.0824 0.1098 0.0941 0.1176 0.1098 0.0980
0.0980 0.0863]';

poly3 = polyfitn([x_fit y_fit],z_fit,3);
[r, c] = size(gel_crop_im);

[x_grid, y_grid] = meshgrid(1:c,1:r);
x_bg1 = reshape(x_grid,r*c,1);
y_bg1 = reshape(y_grid,r*c,1);
z_bg1 = polyvaln(poly3,[x_bg1 y_bg1]);

x_bg = reshape(x_bg1,r,c);
y_bg = reshape(y_bg1,r,c);
z_bg = reshape(z_bg1,r,c);

figure(3)
surf(gel_crop_im,'EdgeColor','none','FaceLighting','gouraud','EdgeLigh
ting','gouraud')
view([0 1 0.5])
set(gcf,'Position',[450 25 400 400])
x_lim = get(gca,'Xlim');
y_lim = get(gca,'Ylim');
z_lim = get(gca,'Zlim');
xlabel('Pre background subtration','FontSize',20,'Fontweight','bold')
set(gca,'Clim',[min(min(gel_crop_im)) max(max(gel_crop_im))])

figure(4)
surf(z_bg,'EdgeColor','none','FaceLighting','gouraud','EdgeLighting','
gouraud')
view([0 1 0.5])
set(gcf,'Position',[875 25 400 400])
set(gca,'Xlim',x_lim,'Ylim',y_lim,'Zlim',z_lim)
xlabel('Fitted background','FontSize',20,'Fontweight','bold')
set(gca,'Clim',[min(min(gel_crop_im)) max(max(gel_crop_im))])

gel_norm = gel_crop_im - z_bg;

figure(5)
surf(gel_norm,'EdgeColor','none','FaceLighting','gouraud','EdgeLightin
g','gouraud')
view([0 1 0.5])
```

```
set(gcf,'Position',[1300 25 400 400])
xlabel('Post background subtration','FontSize',20,'Fontweight','bold')

%% Thesholding and rotating image

% threshold image
z_norm = reshape(gel_norm,1,r*c);
im_avg = mean(z_norm);
im_std = std(z_norm);
im_thrsh = zeros(size(gel_norm));
im_thrsh(gel_norm>(im_avg+1*im_std)) = 1;

figure(7)
imshow(im_thrsh)
% imwrite(im_thrsh,'im_thresh.tif','tif');
im_thrsh2 = medfilt2(im_thrsh,[4 4]);
xlabel('Thresholded image','FontSize',20,'Fontweight','bold')
figure(9)
imshow(im_thrsh2)
xlabel('Median Filtered threshold','FontSize',20,'Fontweight','bold')

% rotating image if necessary

theta_deg = -5:0.2:5;
col_num_zero = 1000*ones(size(theta_deg));
max_col_sum = zeros(size(theta_deg));

% uncomment below to do automatic rotation
% for n=1:length(theta_deg)
%     gel_rot_thrsh = imrotate(im_thrsh2,theta_deg(n),'bicubic');
% %     figure(8)
% %     imshow(gel_rot_thrsh)
%     col_sum = sum(gel_rot_thrsh,1);
%     max_col_sum(n) = max(col_sum);
% %     figure(9)
% %     plot(1:length(col_sum),col_sum)
%     col_nonzeros = nonzeros(col_sum);
%     col_num_zero(n) = length(col_sum)-length(col_nonzeros);
% %     keyboard
% end

% [max_col, max_n] = max(col_num_zero);
% theta_rot = theta_deg(max_n);

% uncomment below to do manual rotation
% manually rotate
figure(8)
imshow(gel_norm,[0 max(max(gel_norm))])
colormap jet
xlabel('Set line for rotation','FontSize',20,'Fontweight','bold')
rot_line = imline(gca);
pause
```

```
rot_pos = rot_line.getPosition;
rot_x1 = rot_pos(1,1);
rot_x2 = rot_pos(2,1);
rot_y1 = rot_pos(1,2);
rot_y2 = rot_pos(2,2);
theta_rot = 180/pi*atan((rot_y2-rot_y1)/(rot_x2-rot_x1));



gel_rot_thrsh = imrotate(im_thrsh2,theta_rot,'bicubic');
[r1 c1] = size(gel_rot_thrsh);


%% increasing display resolution by interpolation
gel_rot = imrotate(gel_norm,theta_rot,'bicubic');
gel_norm2 = imresize(gel_rot,4,'bicubic');

figure(6)
surf(gel_norm2,'EdgeColor','none','FaceLighting','gouraud','EdgeLighti
ng','gouraud')
grid off
% view([1 0 4])
view([0 1 20])
set(gcf,'Position',[1725 25 400 400])
set(gca,'XDir','reverse')
xlabel('Rotated image','FontSize',20,'Fontweight','bold')

%% Detect lane edges
% Make sure you define the number of bands
N = 7; % number of bands
band_left = zeros(1,N);
band_right = zeros(1,N);

k_left=1;
k_right=1;
k=1; % if k=1 looking for left edge (if k=2 looking for right edge)

for n=1:c1
    col_sum_n = sum(gel_rot_thrsh(:,n));
    if k==1 % looking for left edge
        if col_sum_n ~= 0
            band_left(k_left)=n-1;
            k_left=k_left+1;
            k=2;
        end
    else
        if col_sum_n==0
            band_right(k_right)=n;
            k_right=k_right+1;
            k=1;
        end
    end
```

```
end

h10 = figure(10);
h10_axes = axes;
imshow(gel_rot), hold on
colormap jet
set(gcf,'Position',[750 600 1000 300])



%% Removing negative intensity from wells

% Set a rectangle to define new background. Wells will be removed by
% eliminating any pixels with intensity less than then minimum of the
% background.
xlabel('Set rectangle for
background','FontSize',20,'Fontweight','bold')
gel_back_rect = imrect(gca); % selecting rectangle of background on
background subtracted image
pause
back_rect = gel_back_rect.getPosition;
xpos_bg = back_rect(1);
ypos_bg = back_rect(2);
box_width_bg = back_rect(3);
box_height_bg = back_rect(4);

im_back_rect =
gel_rot(ypos_bg:(ypos_bg+box_height_bg),xpos_bg:(xpos_bg+box_width_bg)
);

% Now plot lane edges after selectingbackground
for i=1:N
    plot(band_left(i)*ones(1,r1),1:r1,'r','linewidth',2)
    plot(band_right(i)*ones(1,r1),1:r1,'b','linewidth',2)
end

min_bg = min(min(im_back_rect));
gel_rot2 = gel_rot;
gel_rot2(gel_rot2<min_bg)=0;

figure(11)
surf(gel_rot2,'EdgeColor','none','FaceLighting','gouraud','EdgeLightin
g','gouraud')
grid off
% view([1 0 4])
view([0 1 20])
set(gcf,'Position',[2225 25 400 400])
set(gca,'XDir','reverse')
xlabel('Neg. intensity of wells
removed','FontSize',20,'Fontweight','bold')

%% Calculating amount in folded structure band
```

```
% summing total intensity for each lane
tot_band_int = zeros(1,N);
for i=1:N
    tot_band_int(i) = sum(sum(gel_rot(:,band_left(i):band_right(i))));
end

% Defining section for folded structure band
figure(10)
xlabel('Set top of folded band
region','FontSize',20,'Fontweight','bold')
h_line1 = imline(h10_axes);
pause
pos_line1 = h_line1.getPosition;
c1 = polyfit(pos_line1(:,1)',pos_line1(:,2)',1);

pos_line2 = pos_line1;
pos_line2(:,2) = pos_line2(:,2)+14;
xlabel('Set bottom of folded band
region','FontSize',20,'Fontweight','bold')
h_line2 = imline(h10_axes,pos_line2);
pause
pos_line2 = h_line2.getPosition;
c2 = polyfit(pos_line2(:,1)',pos_line2(:,2)',1);
%%
% Displaying folded region

[r_rot c_rot] = size(gel_rot2);
x_plot = 1:c_rot;
y_plot1 = c1(1)*x_plot + c1(2);
y_plot1b = round(y_plot1);
y_plot2 = c2(1)*x_plot + c2(2);
y_plot2b = round(y_plot2);
for i=1:length(x_plot)
    z_plot1(i) = gel_rot2(y_plot1b(i),x_plot(i));
    z_plot2(i) = gel_rot2(y_plot2b(i),x_plot(i));
end

figure(11),hold on
plot3(x_plot,y_plot1b,z_plot1,'linewidth',2,'Color',[1 1 1])
plot3(x_plot,y_plot2b,z_plot2,'linewidth',2,'Color',[1 1 1])

%%

% summing intensity in folded structure band for each lane
band_sum = ones(1,N);
for i=1:N
    band_im = gel_rot(:,(band_left(i)):(band_right(i)));
    band_width = band_right(i)-band_left(i);
    figure(12),clf
    subplot(1,2,1)
    imshow(band_im)
    set(gcf,'Position',[1800 600 100 300])
    x1_top = 1;
    y1_top = c1(1)*band_left(i)+c1(2);
    x2_top = band_right(i)-band_left(i)+1;
    y2_top = c1(1)*band_right(i)+c1(2);
    x1_bot = 1;
    y1_bot = c2(1)*band_left(i)+c2(2);
    x2_bot = band_right(i)-band_left(i)+1;
    y2_bot = c2(1)*band_right(i)+c2(2);
%     hold on
%     plot([x1_top x2_top x2_bot x1_bot x1_top],[y1_top y2_top y2_bot
y1_bot y1_top],'r--')
%     pause
    poly_band = impoly(gca,[x1_top, y1_top; x2_top, y2_top; x2_bot,
y2_bot; x1_bot, y1_bot],'Closed',1);
    band_mask = poly_band.createMask;
    band_sum_im = band_im.*band_mask;
    band_sum(i) = sum(sum(band_sum_im));
end

%%
lane = 1:N;

% Change the independent variable to the appropriate time or
temperature series
% Temp = fliplr([40 41.7 44.4 47.8 52.5 56 58.4 60]);
% Temp = fliplr([48 48.4 48.9 49.6 50.5 51.2 51.7 52]); % LPP fine
screen
% Temp = fliplr([50.0 50.4 50.9 51.6 52.5  53.2 53.7 54.0]); % 18hb
fine screen
% Temp = fliplr([58 58.4 58.9 59.6 60.5 61.2 61.7 62]); % Horse fine
screen
% Time = [1 2 3 4 5 10 15];
Time = [1 2 3];
band_sum_norm = band_sum./tot_band_int;

figure(13),hold on, box on
set(gcf,'Color',[1 1 1])
% set(gca,'FontSize',30,'Xlim',[min(Temp)
max(Temp)],'Xdir','reverse','Ylim',[-10 320])
set(gca,'FontSize',30,'Xlim',[min(Time) max(Time)],'Ylim',[-10 320])
% plot(Temp,band_sum,'k','linewidth',2)
plot(Time,band_sum,'k','linewidth',2)
% xlabel('Annealing Temperature (^oC)','FontSize',30)
xlabel('Annealing Time (min)','FontSize',30)
ylabel('Intensity (a.u.)','FontSize',30)



figure(14),hold on, box on
set(gcf,'Color',[1 1 1])
```

```
% set(gca,'FontSize',30,'Xlim',[min(Temp)
max(Temp)],'Xdir','reverse','Ylim',[-0.05 1])
set(gca,'FontSize',30,'Xlim',[min(Time) max(Time)],'Ylim',[-0.05 1])
% plot(Temp,band_sum_norm,'k','linewidth',2)
plot(Time,band_sum_norm,'k','linewidth',2)
% xlabel('Annealing Temperature (^oC)','FontSize',30)
xlabel('Annealing Time (min)','FontSize',30)
ylabel('Lane Normalized Intensity','FontSize',30)

filename = 'gel_analysis_data.txt';
if exist(filename,'file')
    file_over = questdlg('File exists, do you want to
overwrite?','Check filename',...
        'Yes','No','No');
    if strcmp(file_over,'Yes')
    else
        filename1 = inputdlg({'Enter new file name:'},'Check
File',1,{'.txt'});
        filename = filename1{1};
    end
end
fileID = fopen(filename,'w+');
for i=1:N
fprintf(fileID,'%4.2f\t%4.2f\t%4.3f\n',Time(i),band_sum(i),band_sum_no
rm(i));
%
fprintf(fileID,'%4.2f\t%4.2f\t%4.3f\n',Temp(i),band_sum(i),band_sum_no
rm(i));
end
fclose(fileID);
```

Polyfit Function (for Gel Analysis Code)

```
function polymodel = polyfitn(indepvar,depvar,modelterms)
% polyfitn: fits a general polynomial regression model in n dimensions
% usage: polymodel = polyfitn(indepvar,depvar,modelterms)
%
% Polyfitn fits a polynomial regression model of one or more
% independent variables, of the general form:
%
%   z = f(x,y,...) + error
%
% arguments: (input)
%  indepvar - (n x p) array of independent variables as columns
%        n is the number of data points
%        p is the dimension of the independent variable space
%
%        IF n == 1, then I will assume there is only a
%        single independent variable.
%
%  depvar   - (n x 1 or 1 x n) vector - dependent variable
%        length(depvar) must be n.
%
%        Only 1 dependent variable is allowed, since I also
%        return statistics on the model.
%
%  modelterms - defines the terms used in the model itself
%
%        IF modelterms is a scalar integer, then it designates
%           the overall order of the model. All possible terms
%           up to that order will be employed. Thus, if order
%           is 2 and p == 2 (i.e., there are two variables) then
%           the terms selected will be:
%
%              {constant, x, x^2, y, x*y, y^2}
%
%           Beware the consequences of high order polynomial
%           models.
%
%        IF modelterms is a (k x p) numeric array, then each
%           row of this array designates the exponents of one
%           term in the model. Thus to designate a model with
%           the above list of terms, we would define modelterms as
%
%           modelterms = [0 0;1 0;2 0;0 1;1 1;0 2]
%
%        If modelterms is a character string, then it will be
%           parsed as a list of terms in the regression model.
%           The terms will be assume to be separated by a comma
%           or by blanks. The variable names used must be legal
%           matlab variable names. Exponents in the model may
%           may be any real number, positive or negative.
```

```
%
%            For example, 'constant, x, y, x*y, x^2, x*y*y'
%            will be parsed as a model specification as if you
%            had supplied:
%            modelterms = [0 0;1 0;0 1;1 1;2 0;1 2]
%
%            The word 'constant' is a keyword, and will denote a
%            constant terms in the model. Variable names will be
%            sorted in alphabetical order as defined by sort.
%            This order will assign them to columns of the
%            independent array. Note that 'xy' will be parsed as
%            a single variable name, not as the product of x and y.
%
%         If modelterms is a cell array, then it will be taken
%            to be a list of character terms. Similarly,
%
%            {'constant', 'x', 'y', 'x*y', 'x^2', 'x*y^-1'}
%
%            will be parsed as a model specification as if you
%            had supplied:
%
%            modelterms = [0 0;1 0;0 1;1 1;2 0;1 -1]
%
% Arguments: (output)
%  polymodel - A structure containing the regression model
%         polymodel.ModelTerms = list of terms in the model
%         polymodel.Coefficients = regression coefficients
%         polymodel.ParameterVar = variances of model coefficients
%         polymodel.ParameterStd = standard deviation of model
coefficients
%         polymodel.R2 = R^2 for the regression model
%         polymodel.AdjustedR2 = Adjusted R^2 for the regression model
%         polymodel.RMSE = Root mean squared error
%         polymodel.VarNames = Cell array of variable names
%            as parsed from a char based model specification.
%
%         Note 1: Because the terms in a general polynomial
%         model can be arbitrarily chosen by the user, I must
%         package the erms and coefficients together into a
%         structure. This also forces use of a special evaluation
%         tool: polyvaln.
%
%         Note 2: A polymodel can be evaluated for any set
%         of values with the function polyvaln. However, if
%         you wish to manipulate the result symbolically using
%         my own sympoly tools, this structure can be converted
%         to a sympoly using the function polyn2sympoly. There
%         is also a polyn2sym tool, for those who prefer the
%         symbolic TB.
%
%         Note 3: When no constant term is included in the model,
%         the traditional R^2 can be negative. This case is
%         identified, and then a more appropriate computation
%         for R^2 is then used.
%
%         Note 4: Adjusted R^2 accounts for changing degrees of
%         freedom in the model. It CAN be negative, and will always
%         be less than the traditional R^2 values.
%
% Find my sympoly toolbox here:
%
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?object
Id=9577&objectType=FILE
%
% See also: polyvaln, polyfit, polyval, polyn2sympoly, sympoly
%
% Author: John D'Errico
% Release: 2.0
% Release date: 2/19/06

if nargin<1
  help polyfitn
  return
end

% get sizes, test for consistency
[n,p] = size(indepvar);
if n == 1
  indepvar = indepvar';
  [n,p] = size(indepvar);
end
[m,q] = size(depvar);
if m == 1
  depvar = depvar';
  [m,q] = size(depvar);
end
% only 1 dependent variable allowed at a time
if q~=1
  error 'Only 1 dependent variable allowed at a time.'
end

if n~=m
  error 'indepvar and depvar are of inconsistent sizes.'
end

% check for and remove nans in data
nandata = isnan(depvar) | any(isnan(indepvar),2);
if any(nandata)
  depvar(nandata,:) = [];
  indepvar(nandata,:) = [];
  n = size(indepvar,1);
end

% Automatically scale the independent variables to unit variance
```

```
stdind = sqrt(diag(cov(indepvar)));
if any(stdind==0)
  warning 'Constant terms in the model must be entered using
modelterms'
  stdind(stdind==0) = 1;
end
% scaled variables
indepvar_s = indepvar*diag(1./stdind);

% do we need to parse a supplied model?
if iscell(modelterms) || ischar(modelterms)
  [modelterms,varlist] = parsemodel(modelterms,p);
  if size(modelterms,2) < p
    modelterms = [modelterms, zeros(size(modelterms,1),p -
size(modelterms,2))];
  end
elseif length(modelterms) == 1
  % do we need to generate a set of modelterms?
  [modelterms,varlist] = buildcompletemodel(modelterms,p);
elseif size(modelterms,2) ~= p
  error 'ModelTerms must be a scalar or have the same # of columns as
indepvar'
else
  varlist = repmat({''},1,p);
end
nt = size(modelterms,1);

% check for replicate terms
if nt>1
  mtu = unique(modelterms,'rows');
  if size(mtu,1)<nt
    warning 'Replicate terms identified in the model.'
  end
end

% build the design matrix
M = ones(n,nt);
scalefact = ones(1,nt);
for i = 1:nt
  for j = 1:p
    M(:,i) = M(:,i).*indepvar_s(:,j).^modelterms(i,j);
    scalefact(i) = scalefact(i)/(stdind(j)^modelterms(i,j));
  end
end

% estimate the model using QR. do it this way to provide a
% covariance matrix when all done. Use a pivoted QR for
% maximum stability.
[Q,R,E] = qr(M,0);

polymodel.ModelTerms = modelterms;
polymodel.Coefficients(E) = R\(Q'*depvar);
```

```
yhat = M*polymodel.Coefficients(:);

% recover the scaling
polymodel.Coefficients=polymodel.Coefficients.*scalefact;

% variance of the regression parameters
s = norm(depvar - yhat);
if n > nt
  Rinv = R\eye(nt);
  Var(E) = s^2*sum(Rinv.^2,2)/(n-nt);
  polymodel.ParameterVar = Var.*(scalefact.^2);
  polymodel.ParameterStd = sqrt(polymodel.ParameterVar);
else
  % we cannot form variance or standard error estimates
  % unless there are at least as many data points as
  % parameters to estimate.
  polymodel.ParameterVar = inf(1,nt);
  polymodel.ParameterStd = inf(1,nt);
end

% degrees of freedom
polymodel.DoF = n - nt;

% coefficient/sd ratio for a p-value
t = polymodel.Coefficients./polymodel.ParameterStd;

% twice the upper tail probability from the t distribution,
% as a transformation from an incomplete beta. This provides
% a two-sided test for the corresponding coefficient.
% I could have used tcdf, if I wanted to presume the
% stats toolbox was present. Of course, then regstats is
% an option. In that case, the comparable result would be
% found in:    STATS.tstat.pval
polymodel.p = betainc(polymodel.DoF./(t.^2 +
polymodel.DoF),polymodel.DoF/2,1/2);

% R^2
% is there a constant term in the model? If not, then
% we cannot use the standard R^2 computation, as it
% frequently yields negative values for R^2.
if any((M(1,:) ~= 0) & all(diff(M,1,1) == 0,1))
  % we have a constant term in the model, so the
  % traditional R^2 form is acceptable.
  polymodel.R2 = max(0,1 - (s/norm(depvar-mean(depvar)) )^2);
  % compute adjusted R^2, taking into account the number of
  % degrees of freedom
  polymodel.AdjustedR2 = 1 - (1 - polymodel.R2).*((n - 1)./(n - nt));
else
  % no constant term was found in the model
  polymodel.R2 = max(0,1 - (s/norm(depvar))^2);
  % compute adjusted R^2, taking into account the number of
  % degrees of freedom
  polymodel.AdjustedR2 = 1 - (1 - polymodel.R2).*(n./(n - nt));
end

% RMSE
polymodel.RMSE = sqrt(mean((depvar - yhat).^2));

% if a character 'model' was supplied, return the list
% of variables as parsed out
polymodel.VarNames = varlist;

% ==================================================
% =============== begin subfunctions ===============
% ==================================================
function [modelterms,varlist] = buildcompletemodel(order,p)
%
% arguments: (input)
%  order - scalar integer, defines the total (maximum) order
%
%  p     - scalar integer - defines the dimension of the
%          independent variable space
%
% arguments: (output)
%  modelterms - exponent array for the model
%
%  varlist - cell array of character variable names

% build the exponent array recursively
if p == 0
  % terminal case
  modelterms = [];
elseif (order == 0)
  % terminal case
  modelterms = zeros(1,p);
elseif (p==1)
  % terminal case
  modelterms = (order:-1:0)';
else
  % general recursive case
  modelterms = zeros(0,p);
  for k = order:-1:0
    t = buildcompletemodel(order-k,p-1);
    nt = size(t,1);
    modelterms = [modelterms;[repmat(k,nt,1),t]];
  end
end

% create a list of variable names for the variables on the fly
varlist = cell(1,p);
for i = 1:p
  varlist{i} = ['X',num2str(i)];
end
```

```
% ==================================================
function [modelterms,varlist] = parsemodel(model,p);
%
% arguments: (input)
%  model - character string or cell array of strings
%
%  p     - number of independent variables in the model
%
% arguments: (output)
%  modelterms - exponent array for the model

modelterms = zeros(0,p);
if ischar(model)
  model = deblank(model);
end

varlist = {};
while ~isempty(model)
  if iscellstr(model)
    term = model{1};
    model(1) = [];
  else
    [term,model] = strtok(model,' ,');
  end

  % We've stripped off a model term. Now parse it.

  % Is it the reserved keyword 'constant'?
  if strcmpi(term,'constant')
    modelterms(end+1,:) = 0;
  else
    % pick this term apart
    expon = zeros(1,p);
    while ~isempty(term)
      vn = strtok(term,'*/^. ,');
      k = find(strncmp(vn,varlist,length(vn)));
      if isempty(k)
        % its a variable name we have not yet seen

        % is it a legal name?
        nv = length(varlist);
        if ismember(vn(1),'1234567890_')
          error(['Variable is not a valid name: ''',vn,''''])
        elseif nv>=p
          error 'More variables in the model than columns of indepvar'
        end

        varlist{nv+1} = vn;

        k = nv+1;
      end
      % variable must now be in the list of vars.

      % drop that variable from term
      i = strfind(term,vn);
      term = term((i+length(vn)):end);

      % is there an exponent?
      eflag = false;
      if strncmp('^',term,1)
        term(1) = [];
        eflag = true;
      elseif strncmp('.^',term,2)
        term(1:2) = [];
        eflag = true;
      end

      % If there was one, get it
      ev = 1;
      if eflag
        ev = sscanf(term,'%f');
        if isempty(ev)
            error 'Problem with an exponent in parsing the model'
        end
      end
      expon(k) = expon(k) + ev;

      % next monomial subterm?
      k1 = strfind(term,'*');
      if isempty(k1)
        term = '';
      else
        term(k1(1)) = ' ';
      end

    end

    modelterms(end+1,:) = expon;

  end

end

% Once we have compiled the list of variables and
% exponents, we need to sort them in alphabetical order
[varlist,tags] = sort(varlist);
modelterms = modelterms(:,tags);
```

Polyval Function (for Gel Analysis Code)

```
function ypred = polyvaln(polymodel,indepvar)
% polyvaln: evaluates a polynomial model as a function of its
variables
% usage: ypred = polyvaln(polymodel,indepvar)
%
% arguments: (input)
%  indepvar - (n x p) array of independent variables as columns
%        n is the number of data points to evaluate
%        p is the dimension of the independent variable space
%
%        IF n == 1, then I will assume there is only a
%        single independent variable.
%
%  polymodel - A structure containing a regression model from polyfitn
%        polymodel.ModelTerms = list of terms in the model
%        polymodel.Coefficients = regression coefficients
%
%        Note: A polymodel can be evaluated for any set of
%        values with the function polyvaln. However, if you
%        wish to manipulate the result symbolically using my
%        own sympoly tools, this structure should be converted
%        to a sympoly using the function polyn2sympoly.
%
% Arguments: (output)
%  ypred - nx1 vector of predictions through the model.
%
%
% See also: polyfitn, polyfit, polyval, polyn2sympoly, sympoly
%
% Author: John D'Errico
% Release: 1.0
% Release date: 2/19/06

% get the size of indepvar
[n,p] = size(indepvar);
if (n == 1) && (size(polymodel.ModelTerms,2)==1)
  indepvar = indepvar';
  [n,p] = size(indepvar);
elseif (size(polymodel.ModelTerms,2)~=p)
  error 'Size of indepvar array and this model are inconsistent.'
end

% Evaluate the model
nt = size(polymodel.ModelTerms,1);
ypred = zeros(n,1);
for i = 1:nt
  t = ones(n,1);
  for j = 1:p
    t = t.*indepvar(:,j).^polymodel.ModelTerms(i,j);
  end
  ypred = ypred + t*polymodel.Coefficients(i);
end
```

Normalized FRET Efficiency function (for bulk FRET)

```
function E=RatioAc2(filename)

[Date,Sample,Titrant,Conc,DonExWav,DonExSig,DonExBnk,AccExWav,AccExSig
,AccExBnk,DonCorSig,DonCorBnk,DonCorR1,DonCorR2,DonAbsCo,AccAbsCo,RaIn
tR1,RaIntR2]=textread(filename,'%d %s %s %s %d %s %s %d %s %s %s %s %d
%d %s %s %d %d','headerlines',2);


%  Date - date that the measurement was taken  %d
%  Sample - name of the sample %s
%  Titrant - additional chemical added to systme  %s
%  Conc - concentration of titrant %s

%  DonExWav - Donor Excitation Wavelength in nm %d
%  DonExSig - Text File where the donor excitation data is stored %s
%  DonExBnk - Text File where the Blank excited at DonExWav is stored
%s

%  AccExWav - Accptor Excitation Wavelength in nm %d
%  AccExSig - Text File where the Acceptor excitation data is stored
%s
%  AccExBnk - Text File where the Blank excited at AccExWav is stored
%s

%  DonCorSig - Text File where the donor only correction data is
stored %s
%  DonCorBnk - Text File where the donor only blank excited at
DonExWav is stored %s
%  DonCorR1 - beginning of wavelength range in nm over whilch
DonCorSig
%              will be fit to correct the Donor Excitiation data.
%  DonCorR2 - end of wavelength range in nm over whilch DonCorSig
%              will be fit to correct the Donor Excitiation data.

%  DonAbsCo - Text file where the Donor molar extiction coefficients
are stored %s
%  AccAbsCo - Text file where the Accoptor molar extinction
coefficients are stored %s

%  RaIntR1 - beginning of integration range in nm for Ratio A method
%d
%  RaIntR2 - end of integration rang ein nm for Ratio A method %d

PIFE=zeros(length(size(Date,1)));

for i = 1:size(Date,1)
    i
%-----------------------DONOR Excitation Data---------------------------
----
```

```
    % Open DonExSig File
    fid = fopen(DonExSig{i});
    if fid ~= -1;
        fgets(fid);
        fgets(fid);
        fseek(fid, 0,'cof');
        DS = fscanf(fid, '%f %f %f %f %f %f %f %f', [8 inf])';
        fclose(fid);
    else
        'No DonExSig file'
        fseek(fid ,11, 'bof');
    end


    % Open DonExBnk File
    fid = fopen(DonExBnk{i});
    if fid ~= -1;
        fgets(fid);
        fgets(fid);
        fseek(fid, 0,'cof');
        DB = fscanf(fid, '%f %f %f %f %f %f %f %f', [8 inf])';
        fclose(fid);
    else
        DB = [DS(:,1),ones(length(DS),1),zeros(length(DS),1)];
        'No DonExBnk file'
    end


%--------------------Acceptor Excitation Data-------------------------
----

    % Open AccExSig File
    fid = fopen(AccExSig{i});
    if fid ~= -1;
        fgets(fid);
        fgets(fid);
        fseek(fid, 0,'cof');
        AS = fscanf(fid, '%f %f %f %f %f %f %f %f', [8 inf])';
        fclose(fid);
    else
        'No AccExSig file'
        fseek(fid, 11, 'bof');
    end

    % Open AccExBnk File
    fid = fopen(AccExBnk{i});
    if fid ~= -1;
        fgets(fid);
        fgets(fid);
        fseek(fid,0,'cof');
        AB = fscanf(fid, '%f %f %f %f %f %f %f %f', [8 inf])';
        fclose(fid);
    else
        AB = [AS(:,1),ones(length(AS),1),zeros(length(AS),1)];
        'No AccExBnk file'
    end


%-----------------Donor Correction Signal Data------------------------
----


    % Open DonCorSig File
    fid = fopen(DonCorSig{i});
    if fid ~= -1;
        fseek(fid, 11,'bof');
        DC = fscanf(fid, '%f %f %f %f %f', [5 inf])';
        fclose(fid);
    else
        DC =
[DS(:,1),ones(length(DS),1),zeros(length(DS),1),zeros(length(DS),1),ze
ros(length(DS),1)];
        'No DocCorSig file'
    end


    % Open DonCorBnk File
    fid = fopen(DonCorBnk{i});
    if fid ~= -1;
        fseek(fid, 11,'bof');
        DD = fscanf(fid, '%f %f %f %f %f', [5 inf])';
        fclose(fid);
    else
        DD =
[DC(:,1),ones(length(DC),1),zeros(length(DC),1),zeros(length(DS),1),ze
ros(length(DS),1)];
        'No DonCorBnk file'
    end



    % Open Donor Molar Extenction Coefficient File
    fid = fopen(DonAbsCo{i});
    DE = fscanf(fid, '%f %f', [2 inf])';
    fclose(fid);


    % Open Donor Molar Extenction Coefficient File
    fid = fopen(AccAbsCo{i});
    AE = fscanf(fid, '%f %f', [2 inf])';
    fclose(fid);
```

```
%--------------Signal Correction Calculations-------------------------
----

    % step size between wavelengths
    ST=DS(2,1)-DS(1,1);

    % Corrected Donor Signal - Corrected Blank (Sc/Rc)
    P1=DS(:,4)-DB(:,4);      %1D array of blanked donor sig (BDS) MG
    D1 = horzcat(DS(:,1),P1);  %2D array of wavelengths with BDS in
second column MG

    if i==1;
        DonNorm = max(D1(:,2)); %normalization factor, does not appear
to be used MG
    end


    % Donor Correction Signal - Blank  (with Ref Signal Correction)
    P2=DC(:,4)-DD(:,4);     %pure cy3 minus it's background MG
    D2 = horzcat(DC(:,1),P2); %2D array of wavelength and blanked pure
cy3 signal MG


    % Acceptor Direct Excite Signal - Blank  (with Ref Signal
Correction)
    P3=AS(:,4)-AB(:,4);
    D3 = horzcat(AS(:,1),P3);

    if i==1;
        AccNorm = max(D3(:,2)); %norm factor, not used MG
    end

    % Scaled Donor Only Signal
    if max(D2(:,2))==0;
        'No Donor Only Signal'
        S1 = [DS(:,1),zeros(length(D2),1)];
    else

PIFE(i)=sum(D1([find(D1(:,1)==DonCorR1(i)):find(D1(:,1)==DonCorR2(i))]
,2))/sum(D2([find(D2(:,1)==DonCorR1(i)):find(D2(:,1)==DonCorR2(i))],2)
);
        S1=horzcat(D1(:,1),PIFE(i)*D2(:,2));
    end
    % Correced Acceptor FRET Signal
    PI = max([D1(1,1),S1(1,1)]);
    PE = min([D1(end,1),S1(end,1)]);

    S2 =
horzcat(D1(find(D1(:,1)==PI):find(D1(:,1)==PE),1),D1(find(D1(:,1)==PI)
:find(D1(:,1)==PE),2)-S1(find(S1(:,1)==PI):find(S1(:,1)==PE),2));
```

```
    % Integrate over Acceptor due to direct excitation(AccInt) and
    % Integrate over Acceptor due to FRET from donor (DonInt)
    if i==1;
        AccIntI =
sum((D3([find(D3(:,1)==RaIntR1(i)):ST:find(D3(:,1)==RaIntR2(i))],2)));

    end

    AccInt =
sum((D3([find(D3(:,1)==RaIntR1(i)):find(D3(:,1)==RaIntR2(i))],2)));

    DonInt =
sum((S2([find(S2(:,1)==RaIntR1(i)):find(S2(:,1)==RaIntR2(i))],2)));


    % Ratio A
    RA = DonInt/AccInt;



    % Fret Efficiency

    E(i) = (RA*AE(find(AE(:,1)==AccExWav(i)),2)-
AE(find(AE(:,1)==DonExWav(i)),2))/DE(find(DE(:,1)==DonExWav(i)),2);




    h=figure('name',[num2str(Date(i)),'  ',Sample{i},'  ',Titrant{i},'
',Conc{i}]);
    %Donor Excite plot
    subplot(3,1,1), plot(D1(:,1),D1(:,2),'k-'), title(['Direct Donor
Excitation (',num2str(DonExWav(i)),' nm)']),...
       xlabel('wavelength (nm)'), ylabel('cps'),...
       text(0,1.4,[num2str(Date(i)),'  ',Sample{i},'  ',Titrant{i},'
',Conc{i},'  E=',num2str(E(i))],'units','normalized')

    % Donor only fit to Donor Excitation plot
    subplot(3,1,2), plot(D1(:,1),D1(:,2),'k-'), hold on,
plot(S1(:,1),S1(:,2),'r-'),...
        title(['Donor only fit to Direct Donor Excitation
(',num2str(DonExWav(i)),' nm)']), xlabel('wavelength (nm)'),...
        ylabel('cps'), legend('D+A','D')

    % Extracted acceptor via fret and acceptor direct excite

    subplot(3,1,3), plot(D3(:,1),D3(:,2),'k-'), hold on,
plot(S2(:,1),S2(:,2),'b-'),...
        title(['Acceptor Excitation via FRET and Direct Excitation
(',num2str(AccExWav(i)),' nm)']), xlabel('wavelength (nm)'),...
        ylabel('cps'), legend('Dir Ex','FRET')
    hgsave(h,[filename(1:end-4),num2str(i)]);
    pause(1);
    close(h);

    outfile = [filename(1:end-4),'Sample',num2str(i),'_out.txt'];
    fid = fopen(outfile,'w');
    fprintf(fid,'%-15s %-60s %-15s %-15s %-15s
\n','date','sample','titrant','conc','E');
    fprintf(fid,'%-15d %-60s %-15s %-15s %-15.3d
\n',Date(i),Sample{i},Titrant{i},Conc{i},E(i));
    fprintf(fid,'\n');
    fprintf(fid,'Acceptor Excite Norm Factor from 1st sample = %-
10d\n\n',AccIntI);
    fprintf(fid,'Acceptor Excite Norm Factor from this sample = %-
10d\n\n',AccInt);
    fprintf(fid,'All traces are normalized by
Raw_Data*AccIntI/AccInt\n');
    fprintf(fid,'1st set = Norm D+A Donor Excite, \n2nd set = Norm D+A
Acceptor Excite, \n3rd set = Norm D-only subtracted D+A Dornor Excite,
\n4th set = Norm Scaled D-only Donor Excite\n\n');

    FO = zeros(max([size(D1,1),size(D3,1),size(S2,1),size(S1,1)]),8);
    FO(1:size(D1,1),1:2)=[D1(:,1),D1(:,2)];
    FO(1:size(D3,1),3:4)=[D3(:,1),D3(:,2)];
    FO(1:size(S2,1),5:6)=[S2(:,1),S2(:,2)];
    FO(1:size(S1,1),7:8)=[S1(:,1),S1(:,2)];
    fprintf(fid,'%-5d %-10.9f %-5d %-10.9f %-5d %-10.9f %-5d %-
10.9f\n',FO');
    fclose(fid);


end
E=E'

fid = fopen('output.txt','w');
fprintf(fid,'%-15s %-60s %-15s %-15s %-15s %-15s
\n','date','sample','titrant','conc','FRETeff','PIFEmultipl');
for i = 1:size(Date,1)
    fprintf(fid,'%-15d %-60s %-15s %-15s %-15.3d %-15.3d
\n',Date(i),Sample{i},Titrant{i},Conc{i},E(i),PIFE(i));
end
    fclose(fid);
```

Actuation Bars Code

```
clc; close all; clear all;
set(0, 'DefaultLineLineWidth',3)
set(0, 'DefaultTextFontSize',20)
set(0, 'DefaultAxesFontSize',24)
set(0, 'DefaultFigureColor','White')

% Monomer Actuation:
% state 1:
latched_1_mean = 0.966165414;
latched_1_std = 0.026582962;
trans_1_mean = 0.014285714;
trans_1_std = 0.009569866;
flip_1_mean = 0.019548872;
flip_1_std = 0.017013095;

% state 2:
latched_2_mean = 0.125661354;
latched_2_std = 0.02124581;
trans_2_mean = 0.096464687;
trans_2_std = 0.001990722;
flip_2_mean = 0.777873958;
flip_2_std = 0.019255088;

% controls:
latched_3_mean = 0.965036465;
latched_3_std = 0.026998071;
trans_3_mean = 0;
trans_3_std = 0;
flip_3_mean = 0.034963535;
flip_3_std = 0.026998071;

y=100*[latched_1_mean flip_1_mean trans_1_mean; latched_2_mean
flip_2_mean trans_2_mean; latched_3_mean flip_3_mean trans_3_mean];
std=100*[latched_1_std flip_1_std trans_1_std; latched_2_std
flip_2_std trans_2_std; latched_3_std flip_3_std trans_3_std];


figure( 'Name', 'Monomer Actuation' );
set(gcf,'Position',[50 50 755 400])
h = bar(y,'grouped')
h(1).FaceColor = [0 .5 .8];
h(2).FaceColor = [.2 .8 0];
h(3).FaceColor = [1 .5 0];
set(h,'BarWidth',1);    % The bars will now touch each other
% set(gca,'YGrid','on')
set(gca,'GridLineStyle','-')
set(gca,'XTicklabel','')
yt = get(gca, 'ytick');
ytl = strcat(strtrim(cellstr(num2str(yt'))), '%');
```

```
set(gca, 'yticklabel', ytl);
set(get(gca,'YLabel'),'String','')
lh = legend('Latched','Actuated','Transition');
set(lh,'Location','EastOutside','Orientation','vertical')
hold on;
numgroups = size(y, 1);
numbars = size(std, 2);
groupwidth = min(0.8, numbars/(numbars+1.5));
for i = 1:numbars
      % Based on barweb.m by Bolu Ajiboye from MATLAB File Exchange
      x = (1:numgroups) - groupwidth/2 + (2*i-1) * groupwidth /
(2*numbars);  % Aligning error bar with individual bar
      errorbar(x, y(:,i), std(:,i), 'k', 'linestyle',
'none','linewidth',2);
end

%% Dimer actuation
figure( 'Name', 'Dimer Actuation' );

latched_1_mean = 0.901481481;
latched_1_std = 0.086947945;
trans_1_mean = 0.098518519;
trans_1_std = 0.086947945;
flip_1_mean = 0;
flip_1_std = 0;

latched_2_mean = 0.036375661;
latched_2_std = 0.000935326;
trans_2_mean = 0.347222222;
trans_2_std = 0.137492985;
flip_2_mean = 0.616402116;
flip_2_std = 0.138428312;

y=100*[latched_1_mean flip_1_mean trans_1_mean; latched_2_mean
flip_2_mean trans_2_mean];
std=100*[latched_1_std flip_1_std trans_1_std; latched_2_std
flip_2_std trans_2_std];

set(gcf,'Position',[650 50 550 400])
h2 = bar(y,'grouped')
h2(1).FaceColor = [0 .5 .8];
h2(2).FaceColor = [.2 .8 0];
h2(3).FaceColor = [1 .5 0];
set(h2,'BarWidth',1);    % The bars will now touch each other
% set(gca,'YGrid','on')
set(gca,'GridLineStyle','-')
set(gca,'XTicklabel','')
yt = get(gca, 'ytick');
ytl = strcat(strtrim(cellstr(num2str(yt'))), '%');
set(gca, 'yticklabel', ytl);
set(get(gca,'YLabel'),'String','')
lh = legend('Latched','Actuated','Other');
```

```
set(lh,'Location','EastOutside','Orientation','vertical')
hold on;
numgroups = size(y, 1);
numbars = size(std, 2);
groupwidth = min(0.8, numbars/(numbars+1.5));
for i = 1:numbars
      % Based on barweb.m by Bolu Ajiboye from MATLAB File Exchange
      x = (1:numgroups) - groupwidth/2 + (2*i-1) * groupwidth /
(2*numbars);  % Aligning error bar with individual bar
      errorbar(x, y(:,i), std(:,i), 'k', 'linestyle',
'none','linewidth',2);
end
```