Kinesin's Force Generation Mechanism: Study and Practical Application

by
William R. Hesse
B.S., Drexel University (2009)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2011

Author . . . .
Department of Mechanical Engineering
January 20, 2011

Certified by . . . .
Matthew J. Lang
Associate Professor of Mechanical Engineering and Biological Engineering, *currently* Associate Professor of Chemical and Biomolecular Engineering, Vanderbilt University
Thesis Supervisor

Certified by . . . .
Roger D. Kamm
Singapore Professor of Biological and Mechanical Engineering
or

Accepted by . . . .
dt
Professor of Mechanical Engineering
Graduate Officer

Kinesin's Force Generation Mechanism: Study and Practical Application

by

William R. Hesse

Submitted to the Department of Mechanical Engineering
on January 20, 2011, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Chimeric kinesin 1 (KHC) / kinesin 5 (Eg5) constructs were used to study the force generation mechanism of the motor protein. The kinesin family of proteins walk along microtubules to carry cargo or pull microtubules along each other and do so by hydrolyzing a single ATP per 8nm step. The mechanism that is employed by kinesin to generate the force necessary for motion is not completely understood. One recent model, the cover neck bundle (CNB) model, utilizes two structural elements of the motor (the coverstrand, which is at the N-terminus of the motorhead) and $\beta 9$, which is part of the necklinker (C-terminus of the motorhead) to form a beta sheet. The CNB folds toward the motor head and causes the rear, unbound head to travel from behind to in front of the bound motorhead. Recent investigations have shown that the CNB produces enough force to explain single molecule experiments, and that when the CNB is not allowed to form by deletion of the coverstrand, the motor looses the ability to generate force. Recent experiments on dimeric forms of Eg5 (a member of the kinesin 5 family) have shown that the motor is capable of generating nearly analogous amounts of force as kinesin 1, but that it generally dissociates from the microtubule under load rather than coming to a true stall. To investigate the applicability of the CNB model of force generation and to see if a motor with high force capabilities could be generated, these chimeras employed the Eg5 CNB (and in some cases L13), and the rest of the kinesin 1 motorhead. It was found from experiments with a stationary optical trap to measure stall force and the force-velocity relationship of the motors as well as unloaded measurements, such as processivity and velocity, that motors with a matched CNB operated the best, however the use of the Eg5 CNB did not reproduce the force generation capabilities of dimeric Eg5 constructs. These results suggest that perhaps while the CNB mechanism is very important to force generation, it may not be the full explanation. A possible link between the CNB's effect on the rate of the mechanical step of the mechanochemical cycle and the stall force is discussed. This link between mechanical rate and stall force is a potential avenue for rational design of kinesin motors for a specific stall force. Secondly, these results show that the parts of kinesin are not directly interchangeable, and that careful consideration of the interactions between parts must be considered when engineering these motors. A second set of experiments were designed to explore whether the CNB mechanism could be exploited for a rational purpose. The literature consists of many reports of various methods to target kinesin and stop its motility for therapeutic reason. For example, some chemotherapies target kinesin to treat cancer. In this way, an antibody was designed to bind specifically with the coverstrand, thus hopefully disrupting the CNB formation and mechanism of force generation. These experiments demonstrated the the antibody was successful in targeting the coverstrand and in inhibiting kinesin's motion.

Thesis Supervisor: Matthew J. Lang
Title: Associate Professor of Mechanical Engineering and Biological Engineering, *currently* Associate Professor of Chemical and Biomolecular Engineering, Vanderbilt University

Thesis Supervisor: Roger D. Kamm
Title: Singapore Professor of Biological and Mechanical Engineering

Acknowledgments

The completion of this thesis required the help and support a great number of individuals. I would first like to extend my deepest gratitude to my advisers in this work Matt Lang and Roger Kamm for their support and without whom this research would not have been possible. Next, a close collaborator in this project, Wonmuk Hwang, has been instrumental in experimental design and analysis. I would like to thank Matthew Wohlever, who generated the plasmids for the chimeras and transfected the *E. coli* that were necessary to express the motors. Members of the Lang and Kamm labs, both past and present, have made spending time in the lab a great experience, and for that I am deeply thankful. I would like to particularly thank Miriam Steiner, who provided some of the data used for the wild type kinesin 1 motor, Ricardo Gonzalez Rubio who mentored me early on in the project, and who showed me the single molecule assays, Ahmad "Mo" Khalil and David Appleyard who provided troubleshooting support and clarification on past experiments and the optical trap's operation. Karolina Corin of the Zhang lab was especially helpful in learning the procedure for western blotting. Michael DeMott of the Dedon lab and Ifach Yacoby of the Zhang lab were helpful regarding general laboratory procedures. Youling Zou of Epitoimcs, Inc was a pleasure to work with in producing the specific antibodies. Finally I would like to thank all of my friends at MIT and my parents, which have been a source of never ending support and encouragement. This work was funded by by the DOD NDSEG and NSF GRF fellowships.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to kinesin

1.1 Molecular motors

Molecular motors include kinesin, dynein, myosin, the bacterial flagellar motor, polymerases, the ribosome, the Clp/Hsp family of proteins and the ATP sythases. Of these, kinesin, dynein, myosin, and the polymerases are related in that they use chemical energy from nucleotides (ATP for kinesin, myosin, dynein, and NTPs for polymerases) [1, 2, 3, 4] to generate force and move along a linear track, microtubules, actin, or DNA and RNA, respectively. Kinesin and dynein both translate along microtubules, however, there are distinct structural differences between these two motors, as dynein belongs to the AAA+ family of motors[3]. Kinesin and myosin have striking similarity with respect to the ATP active site, which is also very similar to that of the G-proteins (guanine nucleotide-binding proteins) [5].

Molecular motors carry out a great deal of the necessary processes for life. They allow for the transport of materials within cells such as elements of the cytoskeleton, and signaling molecules to name a few [6]. Motors are also responsible for cell division and contraction by allowing motors to pull on cytoskeletal filaments [7, 8]. The Clp/Hsp proteins are proteases and chaperones that use a motors mechanism to allow for the degradation or remodeling of protein [9, 10]. In the case of the bacterial flagella motor, the motor allows for the movement of bacterial cells [11]. Recently, molecular motors have been suggested for use in nanotechnology applications [12, 13].

1.2 *In-vivo* duties of kinesin

Kinesin is one of a wide variety of molecular motors that are used by cells for a wide variety of tasks including organelle movement [6] and mitotic spindle assembly and organization [7]. Defects in motor-dependent transport are associated with many diseases, including neurodegeneration [14], cancer [15], heart disease [16], developmental defects [17] and ciliopathies [18]. Directed transport by kinesin allows for the polarization of cells, which would not be possible with simple diffusion [6]. The kinesin families are localized in cells as shown in figure 1-1.

In-vivo, kinesin transports cargos in complex arrangements with multiple motors. Multiple copies of kinesin have been found to transport lipid-droplets toward the plus end of microtubules [19]. It was found that unlike *in-vitro* findings, *in-vivo*, cargos are transported the same distance regardless of the number of copies of kinesin present on the cargo, and thus a strict regulation of cargo travel must exist *in-vivo* that doesn't exist *in-vitro*. Recently, *in-vitro* cargos transported by multiples of kinesin have been found to demonstrate step wise movement with steps sizes that are even fractions of kinesin's 8nm step [20]. It has also been suggested from *in-vitro* measurements that cargos transported by two kinesin generally rely upon the action of only one of the attached kinesins at a time [21], except under the high force regime. Bidirectional transport is accomplished via combinations of kinesin and dynein [22]. Combinations of kinesin and myosin are also utilized for transport of cargos across microtubules and actin filaments [23, 24].

1.3 Structure of the kinesin family of proteins

A large number of proteins have been found to be part of the kinesin family via genetic and biochemical approaches [25, 26, 27, 6]. The human genome contains sequences for 45 kinesins [25]. These motors have a motor domain that is very similar among the different types of kinesin [28]. The motor domain contains an ATP binding and a microtubule binding domain, both of which are highly conserved [26]. The location

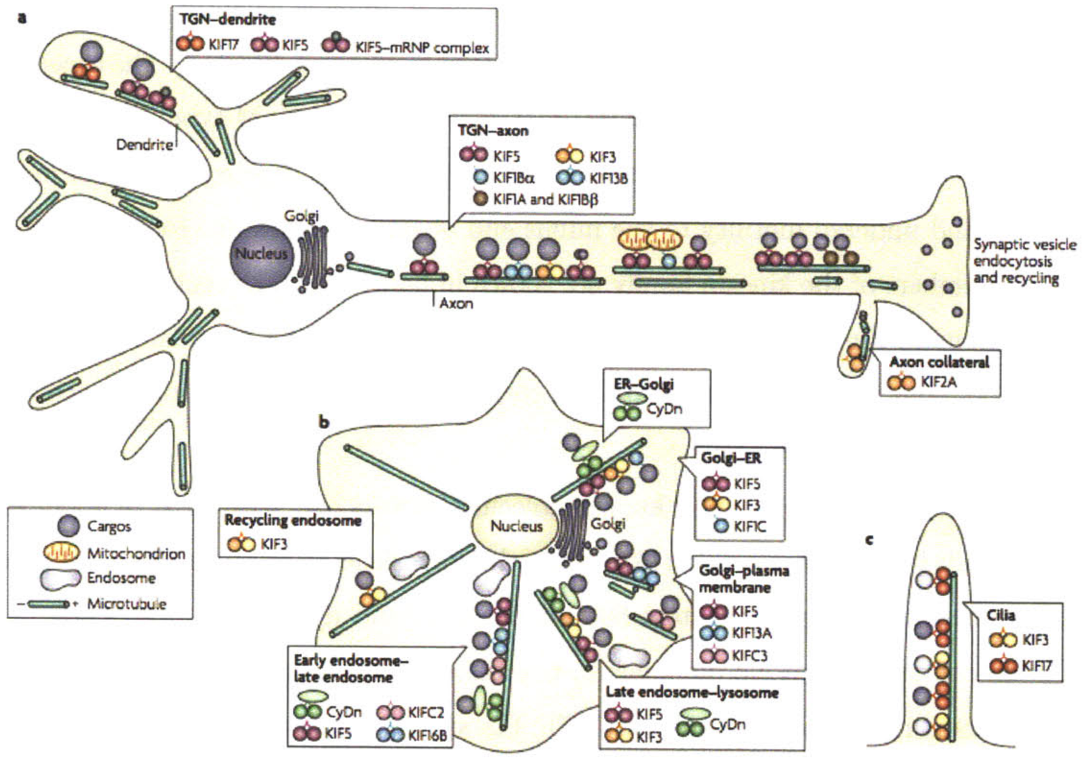


Figure 1-1: Localization of the various members of the kinesin family within cells. a) In neurons plus end directed kinesin transport cargo towards the periphery of the cell due to polarization of the microtubules to always have the plus end pointing towards the peripheral edge of the cell. Transport in the cell body is carried out by kinesin 1 (KIF5), kinesin 2 (KIF3), and kinesin 3 (KIF1A, KIF1Bα, KIF1Bβ, and KIF13B), which are all plus end directed motors. Within the dendrites of neurons, the polarity of the microtubules is mixed, and the transport uses kinesin 1and kinesin 2 (KIF17) motors. Specifically within dendrites, kinesin 1 is responsible for transport of mRNA-protein complexes. Kinesin 13 (KIF2A) is involved in microtubule depolymerization within dendrites. b) Within non-neuronal cells, microtubules are generally organized such they point to the cell periphery with their plus end, as in neuronal cells. Kinesin 1, kinesin 2, and kinesin 3 (KIF1C, KIF13A, and KIF16B) are used for plus end directed transport while dynein and kinesin 14B (KIFC2 and KIFC3) are used for minus end directed transport. Cargos typically contain both plus and minus end directed motors. c) Intraflegellar transport includes kinesin 2 (KIF3A, KIF3B, and KIF17 which carry cargos towards the plus end of microtubules, which are oriented such that the plus end is oriented towards the end of the flagellum. Endoplasmic reticulum and trans-Golgi network cargos are the major entities being transported within the flagellum. This figure has been reproduced with permission from [6], copyright 2009 Macmillan Publishers Limited.

of the motor domain with respect to the ends of the polypeptide chain determines the directionality of the motor. The majority of kinesin proteins have the motor domain at the amino terminus (N type), and are microtubule plus end directed. The exception to this is the Kinesin-13 family (central motor domain motors, M type) and Kinesin-14 family (carboxy-terminal motors, C type) which have an carboxyl-terminal motor domain and undergo motility to the minus end of microtubules [6]. Representations of the members of the kinesin family are shown in figure 1-2. The regions outside the motor domain are not conserved and are family and function specific. These non-motor domains include regions that form coiled coils for oligomerization, attachment to cargo, and regulation [27].

Kinesin is one of the smallest known molecular motors [29], with a size of about 350 amino acids in the motor domain [30]. The other cytoskeleton related motors are much larger, myosin has approximately 800 amino acids and dynein has over 4,000 [30]. Kinesin constructs may be monomeric, homodimeric, heterodimeric, heterotrimeric, homotetrameric and heterotetrameric [27]. The structure is broken down into seven α-helices, ten β-strands, and thirteen loops [31]. The layout of these structural elements is shown in figure 1-3.

1.4 Overall mechanism of motility

Kinesin takes the chemical energy of ATP and converts it to mechanical energy. It has been shown that kinesin is able to convert about 50 to 70 percent of the chemical energy into mechanical work [32, 33]. These efficiencies were able to be estimated after it was found that kinesin hydrolyzes a single ATP molecule per 8nm step [1]. The exact mechanism regarding the processes by which kinesin carries out its cycle is still not completely clear, although there has been very significant progress identifying many of the critical elements. The way in which kinesin takes its steps, how these steps are regulated (gating), and manner in which kinesin generates force are the fundamental processes which need to be understood in order to fully understand the mechanism of kinesin motion.

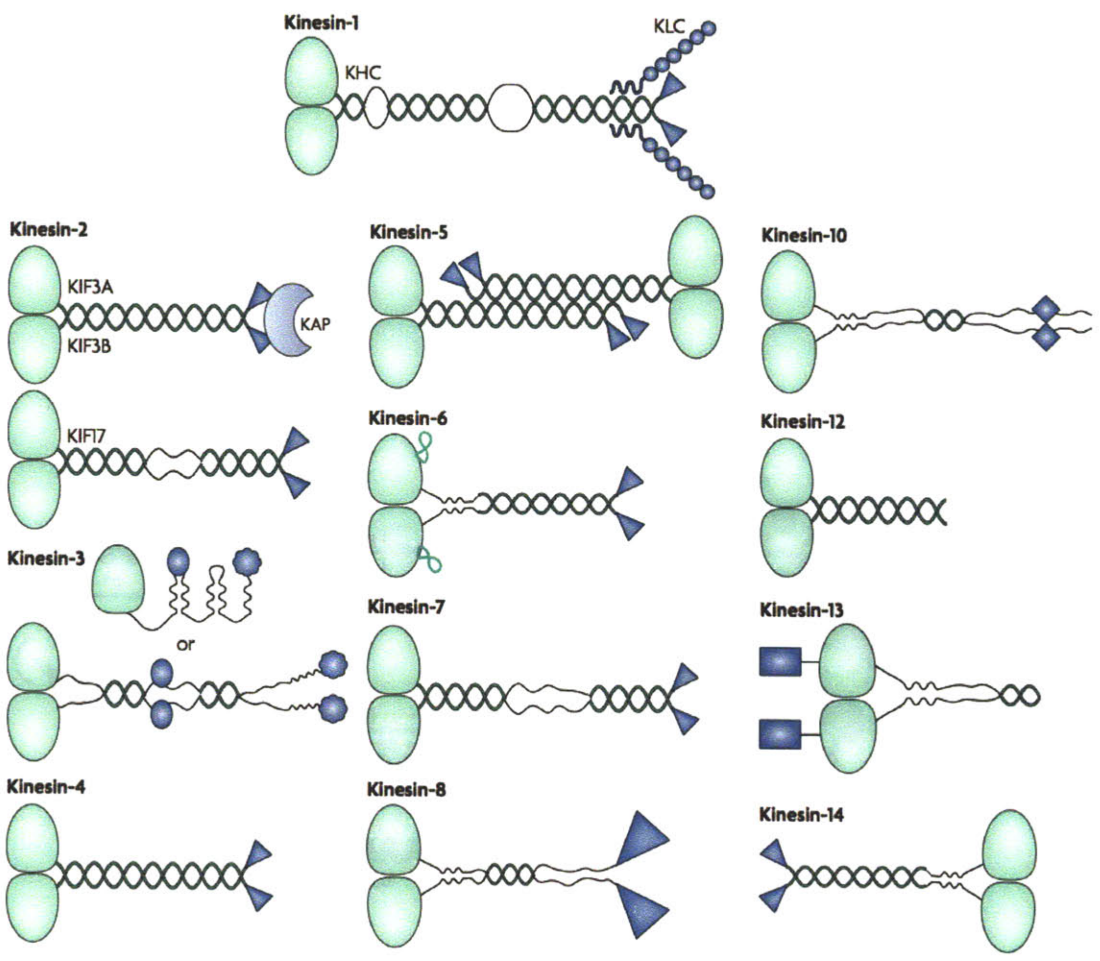


Figure 1-2: Organization of the domains of the members of the kinesin family of proteins. As can be seen, each member consists of a motorhead domain (large green, oblong shapes) and a coiled coil domain for oligomerization, except for the monomeric form of kinesin 3. Generally, the members of the kinesin family also include special domains that are specific to the motor's cargo binding requirements or regulation. Many of the motors exist as homodimers (some members of kinesin 2 and kinesin 3, kinesin 4, kinesin 6, kinesin 7, kinesin 8, kinesin 10, kinesin 12, kinesin 13, and kinesin 14). Kinesin 1 is a heterotetramer with two heavy chains (KHC, containing the motor domain) and two light chains (KLC). Kinesin 5 (referred to generally as Eg5 later) is a homotetramer, and acts to crosslink microtubules. The positioning of the motorhead within the polypeptide chain determines the direction of the motor. Most of the motors are plus end directed (the motor domain is at the N terminus (N type), on the left side of the molecule in the diagram), while kinesin 13 has the motor domain in the middle (M type) and do not have a directional preference, but act to destabilize microtubules, and kinesin 14 (C type) which are minus end directed. Some notable departures from the generalizations above include the ability of kinesin 2 to form heterotetramers (with 2 chains containing motor domains), kinesin 3 to form monomers, kinesin 9 and 14 can both act to destabilize microtubules, and kinesin 14 can also crosslink microtubules. This figure has been reproduced with permission from [27], copyright 2009 Macmillan Publishers Limited.

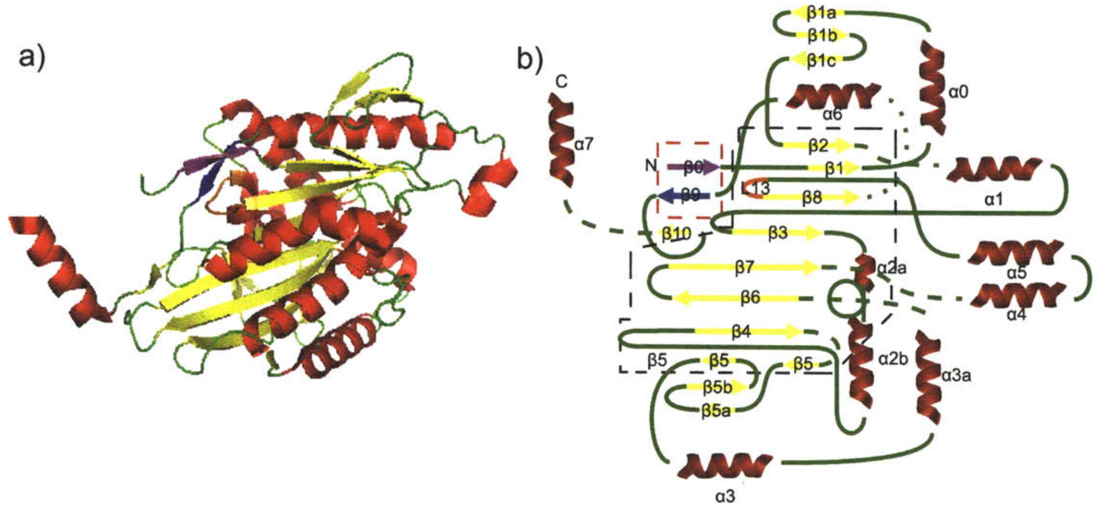


Figure 1-3: Structural layout of the kinesin 1 motorhead (also largely the same for the other members of the kinesin family). a) Three-dimensional view of the motorhead colored to show the segments that are alpha helices (red), beta sheets (yellow), and loops (green). The coverstrand ($\beta 0$) has been colored magenta, $\beta 9$ of the necklinker has been colored blue, and loop 13 (L13) has been colored orange. PDB 1MKJ was used to generate this figure b) Two-dimensional representation of the layout of the motor head. Dotted and dashed lines are included for clarity of which elements the loops connect. The black dashed line surrounds the central beta sheets of the motorhead. The cyan box surrounded with a red, dashed line highlights the location of the cover-neck bundle (CNB).

1.4.1 Models of stepping

The kinesin motor has been shown to step in an asymmetric "hand-over-hand" fashion [34]. This motion is identified by a walking motion where the motor heads trade places with respect to the center of kinesin stalk during the cycle, that is neither of the heads is always in front of the other, and both heads do not complete the same exact motion through the cycle. "Inch-worm" movement had also been suggested, which is where one of the motor heads is always in the front and the other is always in the back, due to the lack of stalk rotation [35], and the assumption that the cycle would be symmetric with both motor heads carrying out the same motion. Subsequently, it was found that the neck stalk (also referred to as the coiled coil) does indeed rotate by 180o in a small fraction of steps [36], although this finding was still able to be explained by the asymmetric hand-over-hand mechanism. The inch-worm mode of movement was found to be inconsistent with the finding of "limping" in homodimeric kinesin constructs, which is only consistent with asymmetric hand-over-hand motion [37]. Measurements taken with a singly labelled motor head demonstrated that the labelled motorhead took 16nm steps [34], which was evidence for either symmetric or asymmetric hand-over-hand, thus the only model that was congruent with all of the reported data was the asymmetric hand-over-hand type of motion.

The reason for kinesin's unidirectional motion has been the subject of great interest. Most members of the kinesin family travel towards the plus end of microtubules, except for kinesin 15, or NCD. One possible reason for the unidirectional motion of kinesin is the formation of the cover-neck bundle and the docking of the necklinker [29, 38, 39, 40, 41, 42] to the motor head. This process causes the necklinker to go from unstructured and generally pointing backwards to structured and pointed towards the front of the motor head, thus bringing the rear head forward and the coiled coil stalk in front of the bound motor head. It was noted that necklinker docking alone did not provide enough of an energy release to cause a directional preference in kinesin. Cover-neck bundle formation, which is the formation of a beta sheet between the coverstrand ($\beta 0$) and the first half of the necklinker ($\beta 9$) is enough to cause the

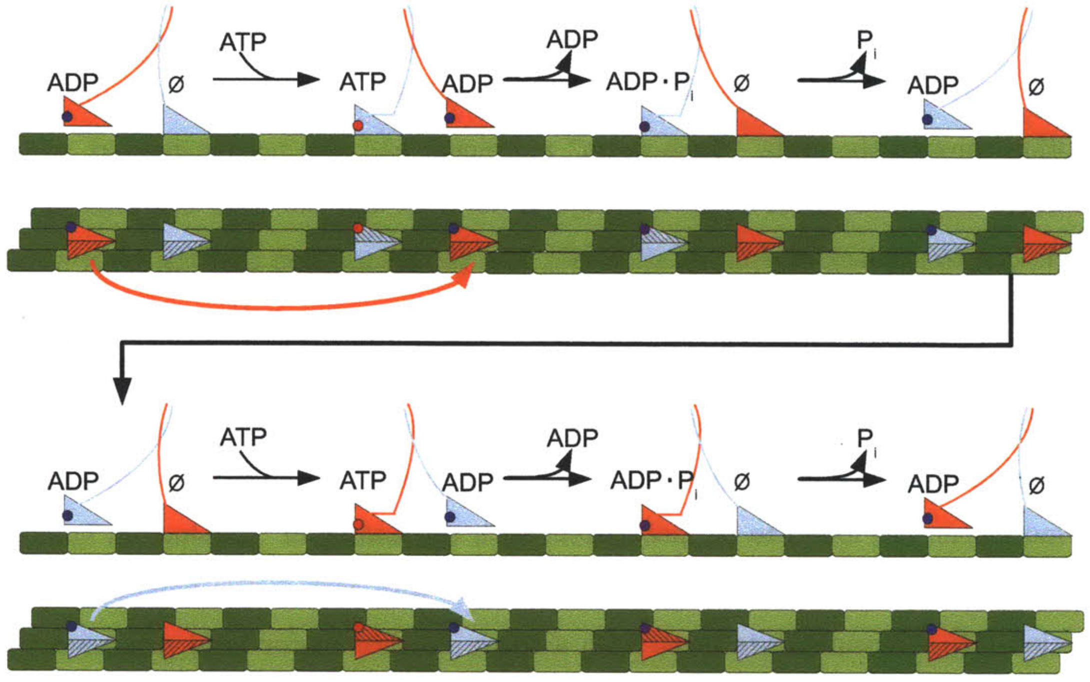


Figure 1-4: Schematic of kinesin's proposed mechanism for motion. The front motor-head attaches to the microtubule in the empty state. In this state, the motor head is tilted such that the left side is raised (the lower side is identified by black hatching). ATP causes the head to rotate and the cover-neck bundle to form thus propelling the trailing head in front of the bound head. The unbound head looses its ADP, which allows it to bind to the next microtubule binding site, while the other head hydrolyzes its ATP to form the ADP·P_i complex. The reverse head looses its organic phosphate leaving the head in the ADP state and thus releases from the microtubule. The process then repeats itself for the other head to complete the cycle. Note that one head passes the bound head on the right and the other passes on the left to form the asymmetric hand-over-hand walking motion.

forward bias of kinesin [29]. The importance of cover-neck bundle formation was also shown when the usually minus end directed motor, NCD, was transformed to be a plus end moving motor with the replacement of its neck domain with that of kinesin 1 [40]. An alternative explanation for the bias of kinesin is that entropy causes a bias towards forward stepping over backward stepping. This entropy was thought to arise from affinity changes between the microtubule binding sites and the motor heads [43]. It was noted that the entropy bias was six times greater than expected for the necklinker docking model, but this doesn't fully describe why NCD would be able to reverse direction by changing only the necklinker, as the motor heads are almost identical to that of kinesin 1. Further, the cover-neck bundle formation model accounts for the ability for directional bias that is not covered by the necklinker docking model.

1.4.2 Gating of the motorheads

Another great controversy regarding the mechanism of kinesin motion is that of how the two heads are controlled with respect to each other [37, 3]. Kinesin 1 is a highly processive motor that can take on the order of hundreds of steps before dissociation from the microtubule. This fact necessitates that the motors be controlled with regard to when one may detach from the microtubule and find the next binding site to take a step. It also remains to be shown why certain members of the kinesin family are very processive, such as kinesin 1, while others, such as kinesin 5, are much less so [44].

One of the most highly regarded theories of how the heads are coordinated is though strain built up in the structural elements that link the motor heads, namely the necklinker and the coiled coil [37]. It was found that the use of non-hydrolyzable ATP analogs induced kinesin to take backsteps before continuing to walk, which suggested that strain gated the front head [45]. This finding meant that without hydrolysis, the only way for the ATP to be released was for the molecule to step back to release the analog and continue forward stepping. The hypothesis of necklinker mediated gating was further supported by results obtained with an assay that used the microbead handle attached to one of the kinesin heads rather than the coiled coil stalk [46], as is typically done. Here, it was found that kinesin could only adopt a state with two bound heads when the rear head's necklinker was docked (ATP or ADP·P_i) and the front head was in a state with an undocked necklinker (no nucleotide or ADP). Single molecule FRET experiments suggested that the rear head waits behind the front head in the ADP state [47], which could be reconciled with the previous results by noting that the FRET data were likely an ensemble average of possible positions, and thus looked like the unbound head had to stay in a prescribed location [46].

1.4.3 Force Generation

The way in which kinesin generates force has been the subject of much debate. A model by which the necklinker docked to the motor to form a power stroke was hypothesized by [42], which was then investigated by thermodynamic means [48]. It was found that the amount of energy accounted for in the docking of the necklinker was a small percentage of that available from ATP hydrolysis, and does not account for the overall efficiency of kinesin, which is approximately 50 percent [37].

A second proposed mechanism for kinesin's ability to generate force is the brownian ratchet, which causes movement rectified by entropy [43]. In this mechanism, the motor walks by allowing the unbound head to randomly diffuse by brownian motion until it finds the next microtubule binding location. It was found in [43] that when the kinesin motility assay was carried out at different temperatures, the number of forward and backward steps that the motor would take changed. As a result it was concluded that entropy change associated with binding the next binding location on the microtubule made it favorable for a the freely diffusing, unbound head to find a binding spot that carried the molecule towards the plus end of the microtubule. As pointed out in [37], the mechanisms of a power stroke and the brownian ratchet are not necessarily exclusive, and that both conformational change and diffusive movement of the motor head likely account for the movement and force generation of kinesin.

Recently, the cover-neck bundle (CNB) formation model was proposed [29, 38]. In this model, the coverstrand ($\beta 0$) and the first half of the necklinker ($\beta 9$, referred to as simply the necklinker later) forms a beta-sheet and folds forward to generate the forward motion and force of the protein. Molecular dynamics simulations showed that the force generated by the CNB was sufficient to match with experimentally determined stall forces of kinesin [29]. Further, kinesin constructs lacking the CNB were unable to generate substantial amounts of force [38]. It is largely believed that ATP binding to the bound head causes the unbound head to move to the next microtubule binding location. It was shown with cryo-EM that upon ATP binding, the kinesin motor head undergoes conformation changes that tilt the head [49] and allows for CNB formation [39].

1.5 Specific Aims of this thesis

This thesis aims to investigate the theory of CNB formation as the force generation mechanism of kinesin and ways in which to harness this mechanism for potential therapeutic means. The CNB model has been tested experimentally with a kinesin construct that lacked the coverstrand, and thus lacked the ability to form the CNB as well as a construct that had two glycines mutated into the coverstrand [38]. The lack of the coverstrand abrogated the ability of the kinesin to generate significant force, while the construct with glycines allowed for a kinesin that was slightly faster than the wildtype, and had a stall force that was reduced by about two piconewtons. The proposed reason for the increase in speed was the increased flexibility of the coverstrand, and thus a faster formation of the CNB. As will be discussed later, the glycines may reduced the stability of the CNB, which made it less resistant to force, thus leading to a lower stall force.

To expand upon these experiments, this thesis demonstrates work done with chimeric kinesin constructs generated with the coverstrand, necklinker and loop 13 from the kinesin 5 protein, Eg5. The reason for the selection of loop 13 (L13) will be discussed in chapter 3. These chimeric proteins were examined using laser tweezers (described in chapter 2) to determine the loaded and unloaded characteristics of the proteins. This allowed for the determination of the stall force, the velocity as a function of applied force, kinetic parameters, and run length as a function of structural make up of the motors. These studies allowed for the elucidation of the relative effects of each of these structural elements as well as how these elements interact.

A second goal was to determine if the CNB model of kinesin motion could be harnessed to engineer methods for controlling kinesin motion. Specifically the thesis addresses wether antibodies targeted towards the coverstrand can stop kinesin from moving. The full details of the motivation behind this specific goal is covered in chapter 4.

Chapter 2

Optical Trapping and Instrumentation

2.1 Theory of Optical Trapping

Trapping particles with light was first described by [50] where latex particles 2.68μm could be trapped in the center of a 514.5nm wavelength laser. Later it was shown that particles ranging in size from 25nm to 10μm could be trapped, which confirmed that the trap could operate in the Rayleigh regime [51]. Subsequently it was shown that optical trapping could be extended to biologically relevant specimens, such as viruses and bacteria [52]. Optical traps have found use in an extremely wide array of experiments ranging from condensed matter physics to biophysics [53]. An extensive review of optical trapping theory, methods, instrument design, and experiments can be found in [54].

The method of treatment used to describe the physics behind the trapping of dielectric particles varies depending upon the relative size of the particle to the wavelength of the light used. When the particle is much larger than the wavelength of light used for illumination, the ray optics model (see figure 2-1) may be used to describe the action of optical traps. In the ray optics approach, the light refracts as it passes through the particle, which is generally spherical in the case of biophysical experiments. This refraction causes a change of momentum in the light, as the light exits the particle. Conservation of momentum dictates that this change in momentum be accounted for, and results in a force that acts in a direction opposite to the change in direction of the light due to refraction. The light has a higher intensity near the center of the laser beam (as is true in the case of a TEM_{00} mode laser, is generally correct in practice), thus when the particle is displaced from the center, light that is refracted from the center of the beam causes a restoring force that pulls the particle back to the center of the beam. When the particle is in the center of the beam, both beams are refracted equally and the intensities are symmetric along the face of the particle that is illuminated, thus the particle stays in the center of the beam. The force of the light to push the particle in the direction of light propagation is cancelled by scattering. In reality, the bead comes to equilibrium slightly downstream from the true waist of the trap, and must be accounted for when calibrating optical trapping instruments.

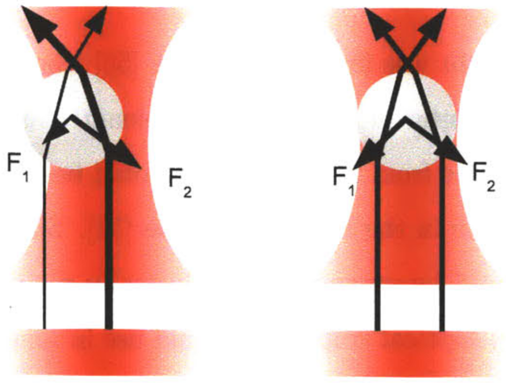


Figure 2-1: Explanation of the mechanism of optical trapping with ray optics. The gaussian beam (signified by the color bar below each trapped bead image) causes the light intensity to vary radially (perpendicular to the axis of propagation of the light) in the beam. When the particle is not in the center of the trap, high intensity light is refracted towards the outside edge of the trap and the change in moment creates a strong force that points towards the center of the trap. The lower intensity light that is refracted towards the center of the trap creates a small force that points towards the the outside of the trap. The balance of these forces creates a net force towards the center of the trap. When the particle is in the center of the trap, the light is symmetrically refracted by the particle, and there is no net force in the radial direction, thus causing the particle to remain in the center of the trap.

If the particle is much smaller than the wavelength of light, then electric dipole approximation may be used. In this model, the particle is treated as a point dipole in an inhomogeneous electric field. The following mathematical treatment for this case comes from [55]. A schematic of the relevant axes and quantities is shown in figure 2-2.

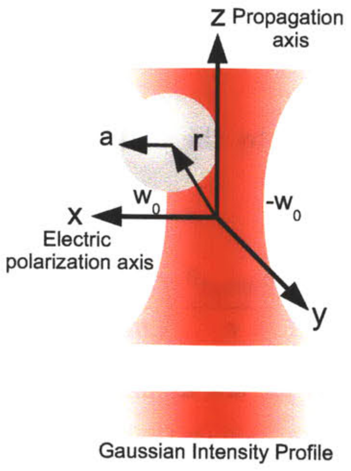


Figure 2-2: Coordinate system and variables of interest in the explanation of the mechanism of optical trapping in the Rayleigh regime. The z axis points in the direction of propagation of light, the x axis points in the direction of the electric polarization. The axis system has its origin at the center of the beam at the beam's waist. The variable w_0 is the radius of the beam at the waist, a is the radius of the particle being trapped, and r is the distance of the particle from the origin.

The scattering force is given by

$$F_{scat}(r) = \hat{z}\left(\frac{n_2}{c}\right) C_{pr} I(r) \tag{2.1}$$

where z is the distance along the axis of propagation of the laser, n_2 is the refractive index of the surrounding medium, C_{pr} is the cross section over which the radiation pressure acts, and $I(r)$ is the light intensity. The scattering cross section and light intensity can be substituted to give

$$F_{scat}(r) = \hat{z}\frac{n_2}{c}\frac{8}{3}\pi(ka)^4 a^2 \left(\frac{m^2-1}{m^2+1}\right)^2 \left(\frac{2P}{\pi w_0^2}\right) \frac{1}{1+(2\tilde{z})^2} exp\left[-\frac{2(\tilde{x}^2+\tilde{y}^2)}{1+(2\tilde{z})^2}\right] \tag{2.2}$$

where a is the radius of the particle, the normalized spatial coordinates are given by $(\tilde{x}, \tilde{y}, \tilde{z}) = (x/w_0, y/w_0, z/kw_0^2)$, k is the wave number in the medium given by $k = 2\pi/\lambda$, and w_0 is the radius of the beam at the focus.

The force that keeps the particle in the center of the beam, the gradient force is given by

$$F_{grad}(r,t) = [p(r,t) \cdot \nabla] E(r,t) = 4\pi n_2^2 \epsilon_0 a^3 \left(\frac{m^2-1}{m^2+2}\right) \frac{1}{2} \nabla E^2(r,t) \tag{2.3}$$

The steady state force experienced by the particle is the time average of the above equation which results in

$$F_{grad}(r) = \frac{2\pi n_2 a^3}{c} \left(\frac{m^2-1}{m^2+2}\right) \nabla I(r) \tag{2.4}$$

In practice, the particles trapped for use in single molecule experiments, and indeed those used in this work, are on the same order of magnitude in size as the wavelength of the trapping laser (1064nm in this case) and the physics become complicated. Both ray optics and effects from the Rayleigh regime play a role in explaining the mechanism for trapping, and the solution to these problems becomes complex to calculate [56].

2.2 Experimental Instrumentation

The instrument used in the experiments carried out for this thesis utilizes a single trapping laser and a second laser for position detection. A schematic of the experimental apparatus is shown in figure 2-3. A 1064nm wavelength Nd:YAG laser was used for trapping. The trapping beam first passed through a zeroth order half wave plate mounted on a rotational mount. This allows for adjustment of the polarization, which modulates the strength of the laser beam after the beam passes through a polarizing beam splitter cube, the next optic in the beam's path. The beam next travels through an acousto-optic deflector (AOD), which steers the beam in two dimensions via scatter from acoustic energy. A rigorous explanation of the operation of AOD's can be found in [57]. The beam is then expanded by a set of lenses and steered by another set of lenses before entering the microscope. The set of steering lenses is used to position the trap such that it appears on the monitors being used for viewing the microscope's field of view. The difference in beam steering between the AODs and lenses is that the AODs are used for moving the trapped particle during an experiment while the lenses are used for alignment purposes and are not adjusted during an experiment. A 975nm wavelength laser is used for particle detection. This laser is expanded and steered by a set of lenses then introduced into the same set of optics as the trapping laser with a dichroic mirror. This setup allows for independent x, y, and z positioning of the beams and combined movement of both beams along the z axis so that trapped particle can be placed within the focal plane of the microscope and the particle sits in the detection laser's waist, which as discussed above, is not the waist of the trapping laser as the particle comes to equilibrium at a location slightly down stream from the trapping laser's waist. After the beam passes through the optics of the microscope and sample (polarizing filter, Wallaston prism, objective lens, condenser lens, second Wallaston filter, and the second polarizing filter) the beam is steered by a dichroic mirror to the detection branch. Here the trapping laser is filtered out and the detection laser is focused onto a position sensitive device (PSD) for detection. For all of the experiments conducted here, the fluorescent branch of the instrument was not used, and illumination was provided by a mercury lamp. Differential interference contrast microscopy (DIC) was used for viewing the samples. DIC was necessary for viewing microtubules, which are approximately 20nm in diameter.

2.3 Experimental Assay

Optical traps have been pivotal in the endeavor to understand the mechanism behind kinesin's motion. The assay used for this work is the classical kinesin motility assay, shown in figure 2-4. In this assay, kinesin motors are attached to submicron sized beads via the biotin-avidin interaction. A diffusing bead, which is coated with kinesin, is captured by the optical trap which is used to place the bead near a microtubule. When a motor engages the microtubule, it pulls the bead out of the center of the

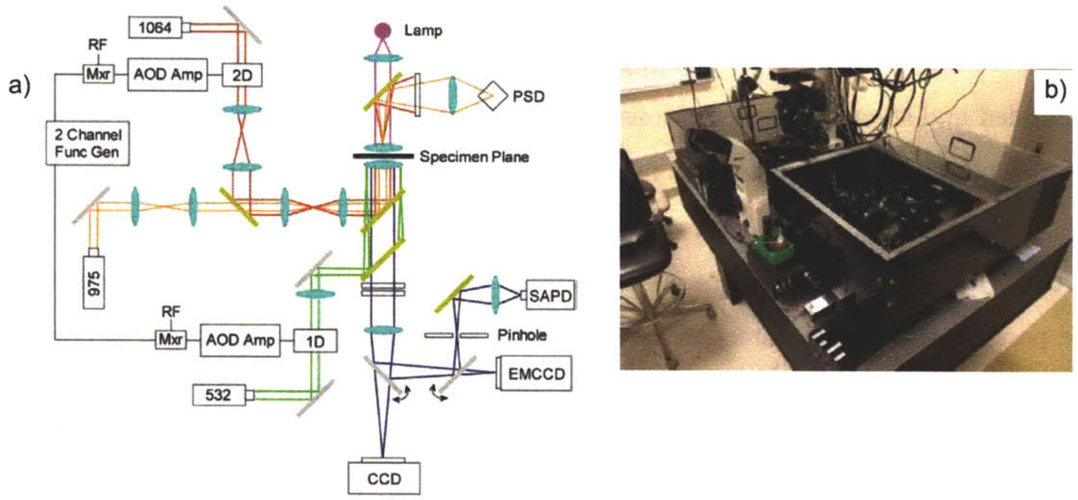


Figure 2-3: Instrumentation used in this study. a) Schematic of the instrument layout which shows the optical path of the trapping (1064) and detection lasers (975) as well as the brightfield illumination system (lamp). The trapping laser is steered with a two-dimensional acousto-optic deflector (AOD), which is controlled via an amplifier connected to a computer using LabVIEW for control. The beam is then expanded and steered with lenses before entering the microscope. Not shown is the zeroth order wave plate and beam splitting cube which act to attenuate the power of the beam before entering the AOD. The detection laser is expanded and steered with lenses before entering the microscope with the trapping laser. After the microscope optics, the trapping and detection laser enter the detection branch via a dichroic mirror. The trapping laser is filtered and the detection laser is focused onto a position sensitive device (PSD) for detection. The PSD's output is sent to a computer for acquisition via a LabVIEW data acquisition board. b) Photograph of the instrument. One of the plexiglass panels that cover the trapping and detection optics has been removed to show the optics. The dark box on the far side of the microscope contains the CCD cameras and single avalanche photodiode (SAPD) for image or photon collection. Figure a) has been reproduced with permission from [58], copyright 2006 Biophysical Society, and b) has been reproduced from the diploma thesis of Benjamin Pelz.

trap until it stalls and detaches from the microtubule. After detachment, the bead returns to the center of the trap.

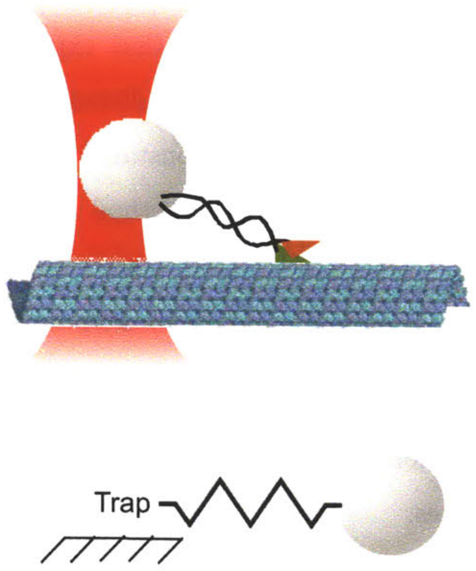


Figure 2-4: Schematic of the assay used in this work. The top figure shows the trapped bead with attached kinesin (not drawn to scale). The kinesin walks along the microtubule (blue and purple filament), which pulls the bead out of the stationary trap, thus causing a force to be exerted on the kinesin molecule. The force on the bead from the trap acts as a Hookean spring, as shown in the bottom figure.

The displacement of the bead is tracked with the use of a second laser that is used for position detection. The light from the detection laser is captured on a position sensitive device (PSD), which has a variable output voltage based on bead position. The voltage output is converted to nanometer space by scanning the bead and recording the voltage signals as a function of bead position. The output voltages from the PSD, V_1 and V_2 are used to obtain the coefficients, a and b for conversion from voltage to nanometers, by fitting with the following fifth order equations.

$$X(V_1, V_2) = \sum_{i,j=0}^{5} = a_{ij} V_1^i V_2^j \tag{2.5}$$

$$Y(V_1, V_2) = \sum_{i,j=0}^{5} = b_{ij} V_1^i V_2^j \tag{2.6}$$

The position sensing system is calibrated so that experiment is performed in the regime where force is linearly proportional to distance. This makes the trap act analogous to a Hookean spring. The "spring constant" that is used to convert distance to force is calculated using the equipartition method

$$\frac{1}{2} k_B T = \frac{1}{2} k_x \langle (x - \overline{x})^2 \rangle \tag{2.7}$$

In this method, the bead is trapped and its position is tracked for a short time. The variance in bead position is then used in the above equation to obtain the trap stiffness in the x and y axes.

After calibration, the bead is placed near a microtubule for a few seconds to determine whether the bead has a kinesin motor that will bind to the microtubule and pull the bead. This process is repeated a total of three times. Experiments in which approximately 50 percent or less of the beads run are used for analysis.

The unloaded measurements are completed by trapping a bead with kinesin and testing it for motility, as with the loaded measurements. Instead of leaving the laser on during the experiment as the bead is pulled by the motor, the trap is shuttered after the motor takes a couple steps. A CCD camera is used instead of the detection laser to record the movement of the bead. After the motor releases from the microtubule, the trapping laser is unshuttered to trap the bead again. The unloaded experiments were calibrated by replacing ATP with AMP-PNP, a non-hydrolyzable ATP analog. AMP-PNP causes the kinesin to bind to microtubules but not run. A bead is trapped, placed next to a microtubule to allow the motor to bind then shuttered. The bead remains tethered to the microtubule, and the piezo stage is used to move the bead while the camera captures the movement of the bead. The known displacements of the bead are used to convert pixels to nanometers. The tracking code is then used to track the beads and ensure that the correct stage velocity is recovered from the tracked bead.

Chapter 3

Study of chimeric kinesin constructs

3.1 Motivation for chimera creation

Chimeras of kinesin 1 and kinesin 5 have been used to elucidate the relative significance of the various structural elements in the kinesin motor head. The past studies done with kinesin I (KHC) / kinesin 5 (Eg5) constructs is shown in table 3.1. The necklinker (both $\beta 9$ and $\beta 10$), coiled coil, and the elements $\beta 8$ through $\alpha 6$ have been investigated previously [59, 60, 61, 62]. New in this study, is the investigation of the coverstrand ($\beta 0$ and loop 13 (L13)). In this study, only $\beta 9$ of the necklinker was mutated to investigate the importance of the cover neck bundle (CNB). A structural alignment done with PDB structures 1MKJ [63] for kinesin 1 and 2WBE [64] for kinesin 5 using the CE calculate two chains tool (`http://cl.sdsc.edu/ce/ce_align.html`). The two proteins align extremely well, which can be seen in 3-1. The alignment is extremely good in the regions of interest for this study, the CS, NL, and L13. Places of diversion include extended loop2 (L2) and loop4 (L5) and shortened $\beta 4$, $\beta 6$, and $\beta 7$. L5 has been the subject of recent investigation [65, 61] and is the target of the anticancer drug monastrol [66].

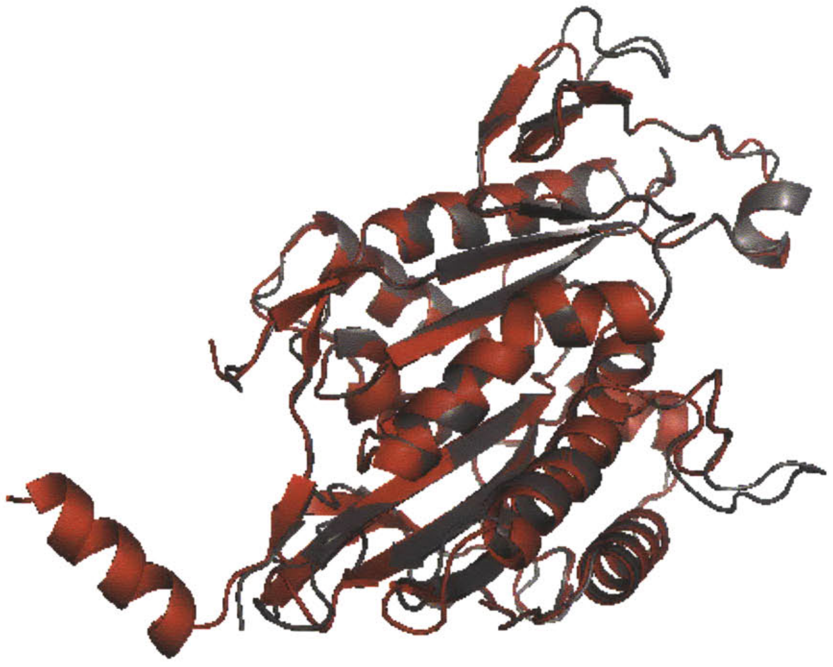


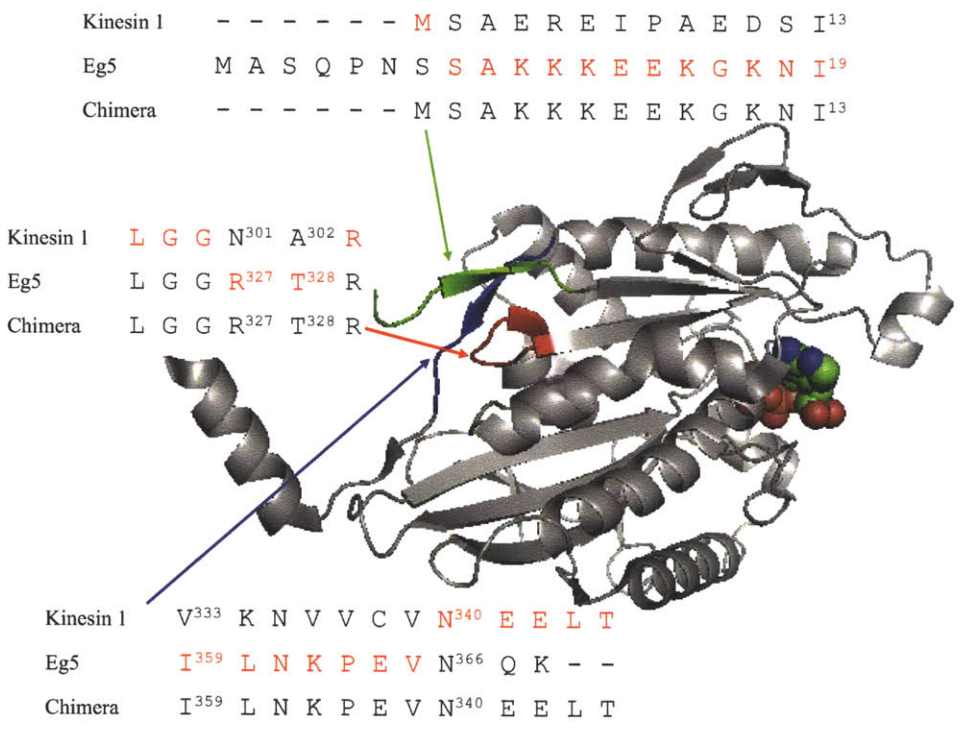
Figure 3-1: Structural alignment of kinesin 1 and kinesin 5 (Eg5). Kinesin 1 is shown in red and Eg5 in gray. It can be seen that the two structures align very well, especially in the segments that are of interest for this study, the coverstrand, necklinker, and L13. The view used here is approximately the same as in 1-3a. Major departures from alignment in the two proteins happen at loops 2 and 5 as well as in the beta sheets at the front tip of the motor ($\beta 4$, $\beta 6$, and $\beta 7$).

Table 3.1: Previous studies using kinesin 1 / kinesin 5 chimeras. The construct name, the origin of each the structural element, a description of the chimera, and the reference from which the chimera originated is provided. The present study is the first to investigate the effects of the coverstrand and loop 13 (L13). The construct designations, WT, CS, NL, L13, CS-NL, CS-NL-L13, which are the constructs used in this work will be used later to describe the constructs listed here.

	Coverstrand		Core		β9		β10		Coiled Coil			
Construct	K	E	K	E	K	E	K	E	K	E	Notes	Ref
K401 (WT)	X		X		X		X		X		*D. melanogaster* KHC up to aa401 (first hinge)	This study, [38]
Eg5		X		X		X		X		X	*X. laevis* Full length, tetrameric Eg5 / Up to aa513, dimer	[67] / [68]
CS		X	X		X		X		X		K401 with Eg5 CS	This study
NL	X		X			X	X		X		K401 with Eg5 β9	This study
L13	X		X	L13	X		X		X		K401 with Eg5 L13	This study
CS-NL		X	X			X	X		X		K401 with Eg5 CS and β9	This study
CS-NL-L13		X	X	L13		X	X		X		K401 with Eg5 CS, β9, and L13	This study
K-E(necklinker)	X		X			X		X		X	*H. sapiens* KHC with Eg5 β9, β9, coiled coil	[59, 61]
E-K(necklinker)		X		X	X		X		X		Eg5 motorhead with *H. sapiens* KHC β9, β9, coiled coil to aa560	[59]
K-E(neck)	X		X		X		X			X	*D. melanogaster* KHC with Eg5 coiled coil	[59]
E-K(neck)		X		X		X		X	X		Eg5 with *H. sapiens* KHC coiled coil	[59]
K(5aa linker)-E(necklinker)	X		X		TIKNT	X		X		X	*H. sapiens* KHC up to middle β9 with Eg5 remainder of β9, β9, coiled coil to aa513	[59]
K-E(β8α6-necklinker)	X		X	β8α6		X		X		X	*H. sapiens* KHC up to β8, Eg5 β8 onwards	[60]
E-K(β8α6-necklinker)		X	β8α6	X		X		X		X	Eg5 up to β8, *H. sapiens* KHC β8 onwards	[60]
DK4mer	X		X		X		X			X	*D. melanogaster* KHC with Eg5 full length coiled coil, tetrameric	[62]

3.2 Chimeras used in this study

The mutations made to the k401 [38, 69] construct are shown in figure 3-2. The sequences of *Drosophila melanogaster* kinesin heavy chain (KHC), *Homo sapiens* kinesin 5 (Eg5), and the resulting chimeras are shown in tables 3.2, 3.3, and 3.4 for the coverstrand, necklinker ($\beta 9$ only), and L13, respectively. Notable mutations that the Eg5 sequences bring to kinesin 1 are the mutation from valine to proline in the necklinker and the mutation from asparagine to arginine in L13. The asparagine in L13 is highly conserved among a wide range of organisms for kinesin 1.



Kinesin 1 - - - - - - M S A E R E I P A E D S I13
Eg5 M A S Q P N S S A K K K E E K G K N I19
Chimera - - - - - - M S A K K K E E K G K N I13
Kinesin 1 L G G N301 A302 R
Eg5 L G G R327 T328 R
Chimera L G G R327 T328 R
Kinesin 1 V333 K N V V C V N340 E E L T
Eg5 I359 L N K P E V N366 Q K - -
Chimera I359 L N K P E V N340 E E L T

Figure 3-2: Identification chimeric sequences used in this study. The fruit fly sequence of the wildtype protein, the human Eg5 sequence, and the resulting chimeric sequence for each of the locations of mutations (coverstrand, green; $\beta 9$, blue; and L13, red) are shown. The view is similar to that used in figures 1-3a and 3-1. ATP is shown in the sphere representation. PDB 1MKJ was used to generate this figure.

Table 3.2: Sequence comparison for the coverstrand. An alignment was made between human kinesin 1 (KHC), fruit fly KHC, and human kinesin 5 (Eg5). The isoleucine at the end of the coverstrand is conserved in all of the proteins listed here. It is interesting to note that the coverstrand is highly variable in length.

															*		*		*
H. sapiens KHC	-	-	-	-	-	-	-	-	-	-	M	A	D	L	A	E	C	N	I^{9}
D. melanogaster KHC	-	-	-	-	-	-	M	S	A	E	R	E	I	P	A	E	D	S	I^{13}
H. sapiens Eg5	M	A	S	Q	P	N	S	S	A	K	K	K	E	E	K	G	K	N	I^{19}

Table 3.3: Sequence comparison for the $\beta 9$ segment of the necklinker. The same sequence alignment was used as in table 3.2. $\beta 9$ corresponds to the segment between the far left isoleucine or valine to asparagine 332 for human KHC (340 for fruit fly KHC and 365 for human Eg5). Of particular note is the proline residue in Eg5 in place of the conserved valine in KHC. Proline is known to act as a beta sheet breaker, thus limiting the size of $\beta 9$.

								*				
H. sapiens KHC	I^{325}	K	N	T	V	C	V	N^{332}	V	E	L	T
D. melanogaster KHC	V^{333}	K	N	V	V	C	V	N^{340}	E	E	L	T
H. sapiens Eg5	I^{359}	L	N	K	P	E	V	N^{366}	Q	K	-	-

Table 3.4: Sequence comparison for loop 13 (L13). The same sequence alignment was used as in table 3.2. Of particular note to L13 is the arginine in Eg5 in place of the conserved asparagine in the KHC sequences. As can be seen, much of L13 is highly conserved.

H. sapiens KHC	L^{290}	G	G	N	C	R^{295}
D. melanogaster KHC	L^{298}	G	G	N	A	R^{303}
H. sapiens Eg5	L^{324}	G	G	R	T	R^{329}

3.3 Kinetic model fits

A number of kinetic mechanisms were considered for fitting including the Boltzmann (equation 3.1), Fisher two state (equation 3.2), and Three State models(equation 3.3). The force-velocity data was fit by the three-state model described in [68], equation 3.3. The Boltzmann model

$$v(F) = \frac{v_{max}\,(1 + A)}{1 + Aexp\left(\frac{F\delta}{k_B T}\right)} \tag{3.1}$$

has been used to model RNA polymerase [4] and kinesin [38] and uses the parameters v_{max}, A, and δ in the fit. In this model, A is the ratio of time of the mechanical component of the cycle to the biochemical component (τ_m/τ_b) and δ is the distance to the transition state. This distance is not necessarily, the step size of the protein, such as in the case of kinesin. The Fisher two state model [70, 71]

$$
\begin{aligned}
v(F) &= d\left(u_0 u_1 - \omega_0 \omega_1\right)/\sigma \\
\sigma &= u_0 + u_1 + \omega_0 + \omega_1 \\
u_0^0 &= k_0^0[ATP] \\
u(F) &= u_j^0 exp\left(-\theta_j^+ F d/k_B T\right) \\
\omega_0^0 &= \frac{k_0'[ATP]}{\left(1+[ATP]/c_0\right)^{1/2}} \\
\omega(F) &= \omega_j^0 exp\left(+\theta_j^- F d/k_B T\right) \\
d_j &= \left(\theta_j^+ + \theta_{j+1}^-\right) d
\end{aligned}
\tag{3.2}
$$

splits the kinesin's cycle into two states with forward and backward rates, which results in the cycle being split into four segments each with an associated rate. In this model, the u terms are forward reaction rates and the ω terms are the reverse rates. The θ terms are the characteristic fractions of the cycle that are occupied by each segment. The sum of all four θ values must equal one. In this model, each of the four rates are capable of being force dependent. The last model considered for fitting was the three state model

$$
\begin{aligned}
v(F) &= \frac{dk_1[ATP]k_2k_3}{k_1[ATP]\left(k_2+k_3\right)+k_3\left(k_2+k_{-1}\right)} \\
k_2 &= k_2^0 exp\left(-F\delta_2/k_B T\right)
\end{aligned}
\tag{3.3}
$$

used in [68]. The rates k_1, k_{-1}, k_2, and k_3 are for ATP binding, ATP dissociation, the mechanical step, and ATP hydrolysis, respectively. k_2^0 is the unloaded rate for the mechanical step and δ_2 is the characteristic distance to the transition state, as in the Boltzmann model.

Each of these models was used to fit the data to determine which would provide the greatest amount of information with simplest form. It was found that the Boltzmann model and the three state model fit the data with nearly identical curves. Upon investigation, it was found that the rates of the biochemical (ATP hydrolysis) and the mechanical step could be extracted from the A value that was fitted by the

Boltzmann model. The time for the biochemical step could be extracted by

$$\tau_b = \frac{8.2}{v_{max}(1 + A)} \tag{3.4}$$

the reciprocal of which is the rate of the biochemical step. The length of the mechanical step was found with

$$\tau_m = A\tau_b \tag{3.5}$$

As with the biochemical step, the reciprocal of this time is the rate of the mechanical step. When these rates were compared with those that were obtained by the three state model it was seen that the rates agreed very well. The additional information that the three state model provided was the rates of ATP binding and dissociation. These rates were globally fit for all of the chimeras and the wild type motors as it is assumed that the mutations do not affect the ATP binding domain of the proteins. This should be a reasonable assumption as the ATP binding domain is located far from any of the mutated regions. The global fit for the ATP binding and dissociation constants gave values that were very close to those that had been previously published [72]. There were great difficulties in using the Fisher two state model for fitting the data due to the large number of parameters to be fitted (nine parameters), and the requirement for all of the θ values to sum to one. For these reasons, the three state model was selected for fitting the data in this investigation.

3.4 Results

3.4.1 Stall force measurements

Representative runs for the various chimeric kinesin constructs are shown in figure 3-3. As can be clearly seen, all chimeras take well defined steps of 8nm, and each of the motors come to a well defined stall. The stall forces obtained by the optical trapping experiments are shown in figure 3-4. Each of the runs was broken into 15ms segments in which the average force was measured as well as average velocity, which was calculated by fitting a line to the position as a function of time data. These data were used to generate the force-velocity relationships seen in figure 3-5. The determination of the force-velocity information from stall force data is considered to be a lower bound [73, 48] because velocities are calculated by assuming that for each run the velocities above the force where the motor stalls is assumed to be zero. The force-velocity data was globally fit using the three state model described in the previous section. The parameters returned by these fits appear in table 3.5. The stall forces are the mean values plus or minus the standard error of the mean. To determine whether the chimeric motors indeed stalled or rather released before a true stall was encountered, the velocity distribution at microtubule release was calculated. The histograms of velocities at stall for each of the kinesin constructs used in this study is shown in figure 3-6. The velocities were normalized to the unloaded velocity of each motor for comparison.

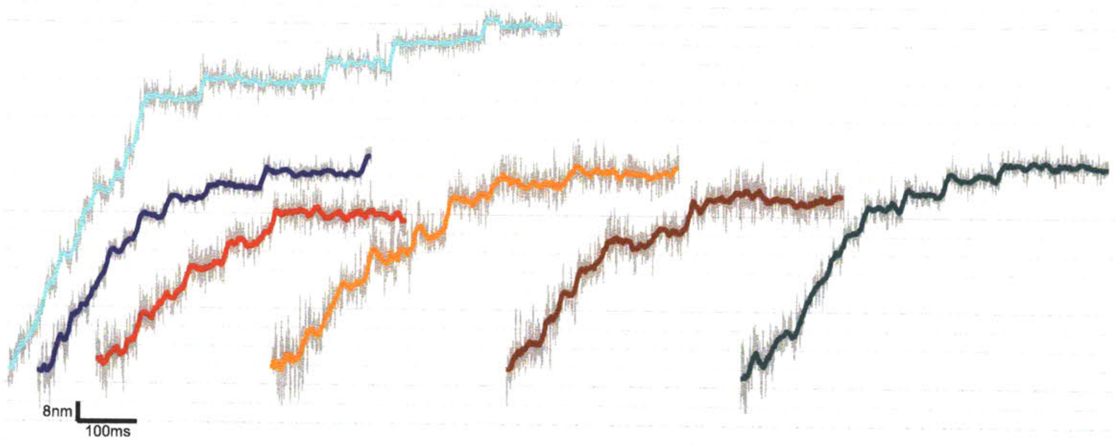


Figure 3-3: Representative traces of runs from each of the constructs used here. Wildtype is shown in cyan, CS in blue, NL in red, L13 in yellow, CS-NL in brown, and CS-NL-L13 in turquoise. In each case, the proteins take well defined 8nm steps have well defined stall plateaus. The scale bars represent 8nm in vertical direction and 100ms in the horizontal direction.

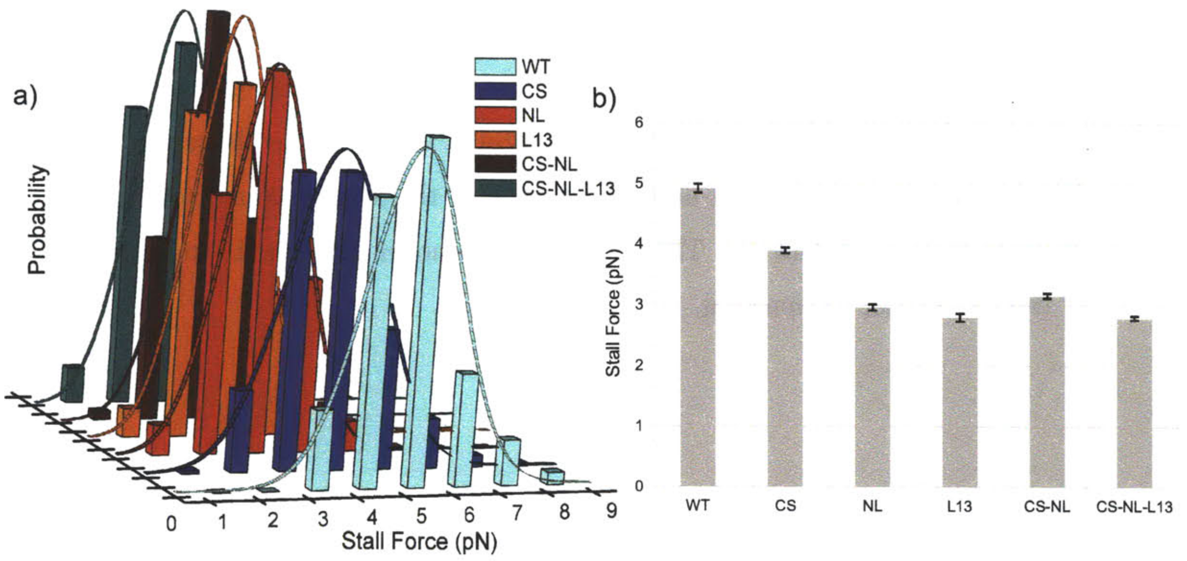


Figure 3-4: Stall forces of each of the kinesin constructs used in this study. The coloring scheme of a) is the same as in figure 3-3. a) Histograms of the stall forces obtained from stationary trap experiments. As can be seen, each of these distributions can be fit with a normal distribution (curve) very well. b) The average stall forces for each of the kinesin constructs. The error bars are plus minus the standard error of the mean. The wildtype motor produced the most force with a stall force of approximately 5pN, while all of the chimeric proteins produced less force. The numerical values of the stall force are shown in table 3.5.

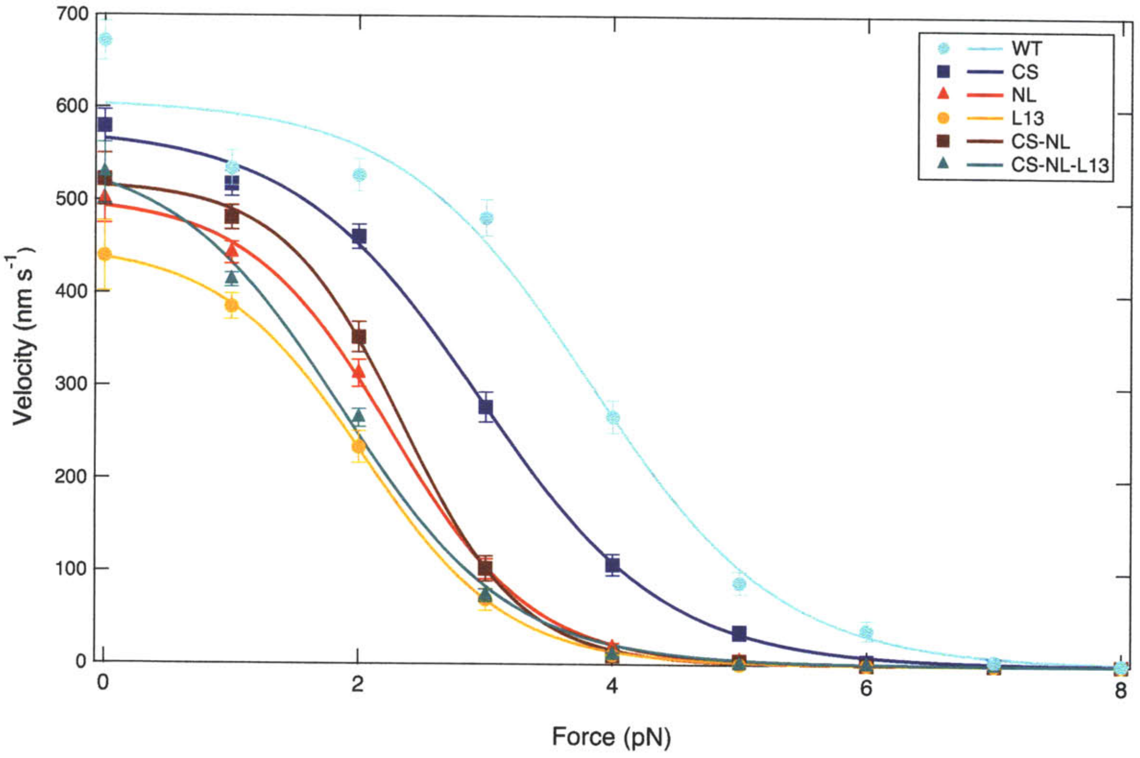


Figure 3-5: The force-velocity behavior of the kinesin constructs used in this study. The same coloring scheme used here is the same as in figures 3-3 and 3-4a. The symbols are are the data obtained by mathematical treatment of the stall force data, as described in the text, except for the velocities at zero force, which were obtained via the unloaded velocity measurements. The error bars are standard error of the mean of the data in each force bin. The data was fit with the three state model, equation 3.3. The unloaded velocities and the data obtained from the stall force measurements were used in fitting the data.

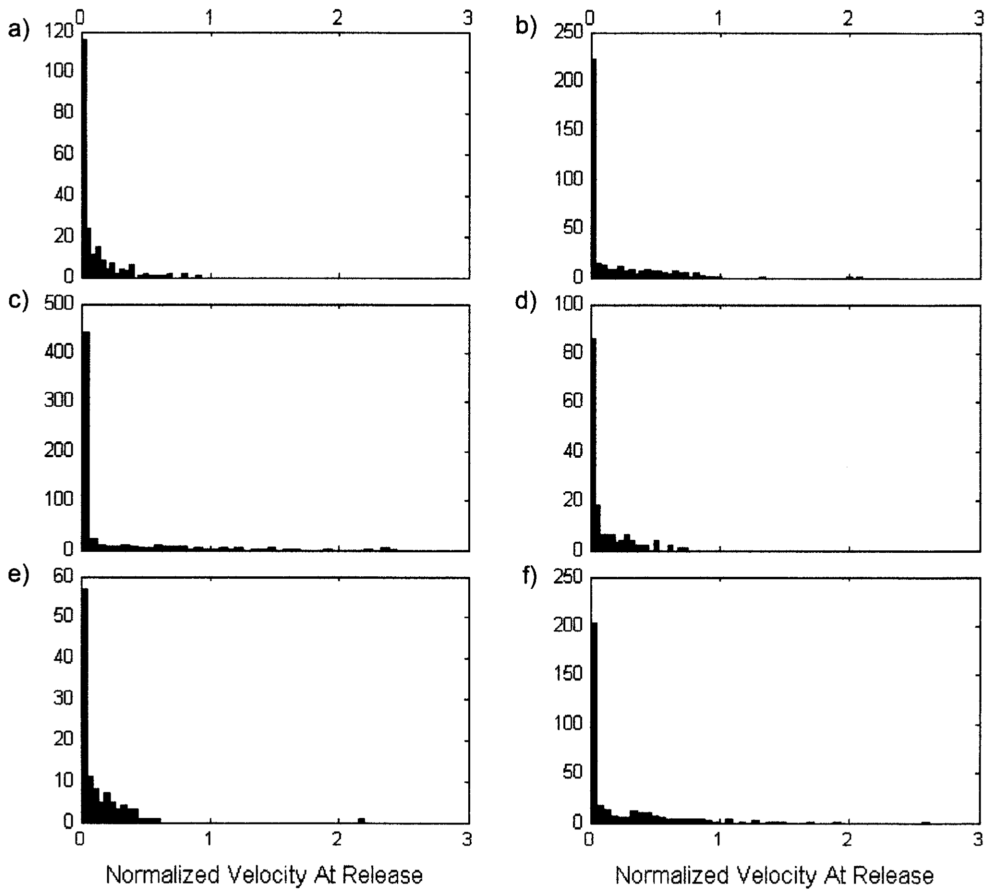


Figure 3-6: The distributions of velocity at stall from the stall force data. The distributions are as follows a) WT b) CS c) NL d) L13 e) CS-NL f) CS-NL-L13. These distributions were obtained by manually fitting a line to time-displacement data obtained from the stall force measurements to the last few moments before dissociation. If the slope of this line was negative (backwards motion), the velocity was assumed to be zero. Each of the distributions was normalized to the unloaded velocity of the respective motor. As can be seen, these motors all had a sharp peak in dissociation velocity at very low speeds and the majority of dissociations occurred below the unloaded velocity.

Table 3.5: Stall force and fitted parameters for force-velocity data for each of the constructs used in this study. The three state model (equation 3.3 was used to fit the force-velocity data shown in figure 3-5 to obtain the values shown here. The stall force is mean plus minus standard error of the mean. The ATP binding and dissociation rates (k_1 and and k_{-1} were globally fit to all of the motors).

	Stall Force	Three State Model						
		k_1	k_{-1}	k_2^0	k_3	δ_2	K_M^0	v_{max}
	(pN)	$(\mu M^{-1} \cdot s^{-1})$	(s^{-1})	(s^{-1})	(s^{-1})	(nm)	(μM)	$(nm \cdot s^{-1})$
WT	4.92 ± 0.08	1.25	619.77	18400	78.65	5.51	63.08	644.95
CS	3.89 ± 0.05	1.25	619.77	6030	74.37	5.65	59.65	609.84
NL	2.95 ± 0.05	1.25	619.77	5150	64.43	7.35	51.68	528.32
L13	2.79 ± 0.06	1.25	619.77	3000	57.41	7.27	46.04	470.75
CS-NL	3.15 ± 0.04	1.25	619.77	13400	66.67	8.69	53.47	546.66
CS-NL-L13	2.78 ± 0.03	1.25	619.77	1690	70.90	6.22	56.87	581.39

Table 3.6: Unloaded velocities and run lengths for each of the constructs used in this study. The data listed is are the average plus or minus the standard error of the mean. This data is visualized in figure 3-7.

	$v^0 (nm \cdot s^{-1})$	Run Length (nm)
WT	671.27 ± 21.15	1163.32 ± 172.08
CS	579.24 ± 17.70	1056.80 ± 252.70
NL	500.67 ± 25.84	259.67 ± 27.12
L13	439.70 ± 37.85	300.94 ± 56.12
CS-NL	521.98 ± 28.17	518.44 ± 79.19
CS-NL-L13	528.82 ± 32.86	573.59 ± 137.67

3.4.2 Unloaded measurements

The unloaded velocities and run lengths of the chimeric kinesin constructs and the wildtype motor are shown in figure 3-7, and the parameters are shown in table 3.6. In all cases, the values are averages plus or minus the standard error of the mean.

3.5 Discussion

Kinesin 1 has been been shown in many studies to have a stall force of approximately 5-7pN, an unloaded velocity in the mid hundreds of nanometers per second and a run length of a few microns [38, 37, 73, 74, 32]. Eg5's motile characteristics are not as clearly defined as kinesin 1. A study that measured the stall force of tetrameric

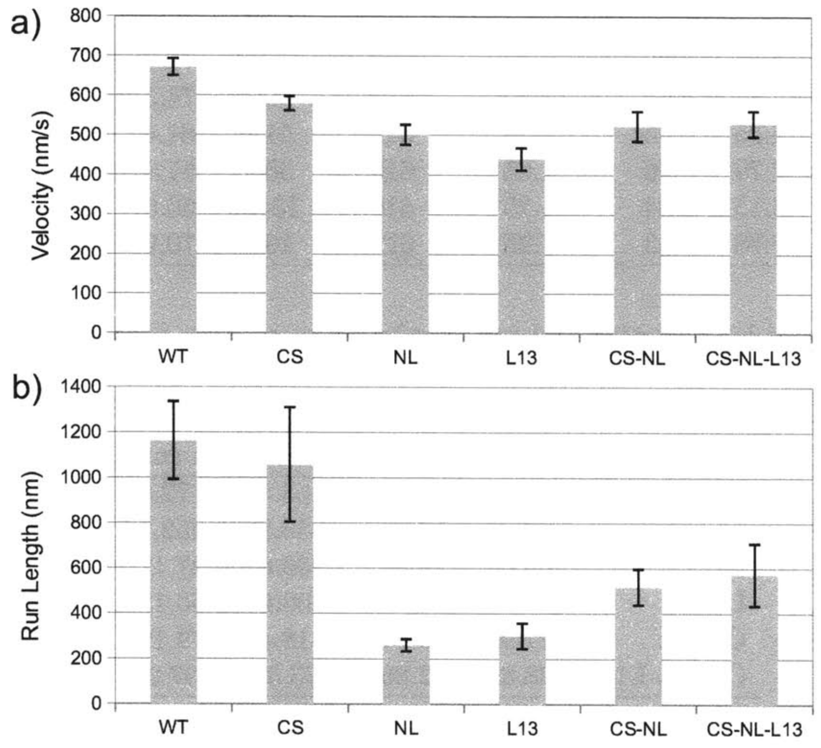


Figure 3-7: Unloaded velocities and run lengths for each of the constructs used in this study. The bars are the average plus or minus the standard error of the mean. As can be seen, the unloaded velocity was lower for all of the chimeras, but generally not drastically so. All of the velocities were well above those found for wildtype Eg5 (around 100nm s^{-1}). The run lengths displayed great variability. The run lengths of the NL and L13 constructs was approximately that of the construct with a deleted coverstrand [38]. The presence of the paired coverstrand (CS-NL) and (CS-NL-L13) recovers some run length, but only to a value about half of the wildtype motor. The numerical values for these bar plots are shown in table 3.6.

Eg5 in a stationary trap found a stall force of approximately 1.5pN, well below that of kinesin 1 [67]. In that study it was also found that the tetrameric Eg5 tended to dissociate from the microtubule before stall. Other studies using truncated constructs of Eg5 found that the motor could pull against significantly higher forces. A truncated form of Eg5 that formed dimers (Eg5-513) was able to walk against forces in excess of 4pN using a force clamp and that forces at dissociation up to 7pN could be obtained with the use of a fixed trap [68]. From the run traces shown in that study, it appeared as though the motor generally dissociated from the microtubule before coming to a stall. A very recent study has used a kinesin 1 / Eg5 chimera to study the effect of monasterol on the processive movement of Eg5 [61]. In that study, the motor was comprised of the Eg5 motorhead up to the coiled coil. After the Eg5 necklinker, the sequence for kinesin 1's coiled coil was used. It was found that this construct had a dissociation force of 4.6pN, and that again, the motor usually came off of the microtubule during the run before stalling. It was suggested from that work that the discrepancy in the force required to dissociate the motor from the microtubule between the dimeric and tetrameric constructs may lie in the C-terminal end of the Eg5, motor which contains the BimC box, which is highly conserved in Eg5. Regardless, the reason for the great difference in tetrameric and dimeric constructs with regards to the force that the motor can withstand is not understood.

It has been recently proposed that the cover neck bundle (CNB), which is the interaction of the coverstrand ($\beta 0$) and $\beta 9$ of the necklinker is the primary mechanism by which kinesin generates force [29, 38]. This contrasts with previous models in which force was generated by necklinker docking to the motorhead, which was found to be slightly thermodynamically favorable [42, 48], however it had been pointed out that it is unknown how this small thermodynamic favorability would allow the motor to step against high loads [37] and that the energy accounted for in necklinker docking is a small fraction of that available from ATP, and it is known that kinesin is at least 50% efficient. This study aimed to further investigate the plausibility of the CNB model of force generation, and to possibly develop a kinesin motor that was able to achieve very high force generating capability by combining kinesin 1's ability to stall with

Eg5's ability to generate high forces without stalling. This aim of engineering a more powerful kinesin motor came with the assumption that the individual parts of proteins would be interchangeable, and that that the motor's characteristics are a sum of the contributions from each of the individual components. To test the above hypothesis, chimeric kinesin 1 / Eg5 constructs were developed that employed all permutations of the Eg5 CNB as well as the Eg5 L13. L13 was also chosen for investigation, as this loop sits directly below the CNB, and has been shown in molecular dynamics simulations to possibly obstruct CNB mediated docking of the necklinker to the motorhead [29]. Further, it was shown in [75] that L13 forms specific contacts with $\beta 9$. Additionally, it was shown that mutation of residues in L13 caused severe defects in motility [76]. In that study, the highly conserved glycine residues in L13 were mutated to alanine (G291A / G292A). It was hypothesized that the reason for the reduced motility was from a reduction in flexibility in L13 [29]. The location of these mutations is shown in figures 3-1 and 3-2. A set of seven chimeras were designed, the plasmids generated (the design and synthesis of the plasmids as well as transformation of the plasmids into *E. coli* was done by Matthew Wohlever), expressed in *E. coli* and purified. Due to constraints on time, the most important of these were characterized using the kinesin motility assay described in chapter 2. These constructs included CS, NL, L13, CS-NL, and CS-NL-L13. The naming of these constructs is such that the signifiers in the name designate which structural elements the chimera posses from Eg5. The chimeric sequences for each of the individual components are found in tables 3.2, 3.3, and 3.4 for the coverstrand, $\beta 9$ (NL), and L13, respectively. Surprisingly, none of the chimeric proteins was able to withstand forces as high as the wildtype motor, and indeed they were below the dissociation forces of the dimeric Eg5 constructs of [68, 61]. The stall forces generated by these motors are shown in figure 3-4 and table 3.5 Each of the chimeric motors also had defects in the unloaded velocity and in unloaded run length (with the exclusion of the CS chimera in terms of unloaded run length). The unloaded characteristics of the motors is shown in figure 3-7 and table 3.6. The motors all retained the ability to take 8nm steps and reach full stall, as shown in figures 3-3 and 3-6.

3.5.1 The role of the coverstrand

The question then becomes why do the chimeras demonstrate reduced functional properties and what does this information mean for the current models of kinesin force generation? It appears from the data here and in [38] that the coverstrand has the ability to affect the velocity of the motor in addition to force generation. The fact that the Eg5 coverstrand did not affect the motor as greatly as the other mutations suggest that perhaps the coverstrand, while necessary may not need to be as specific as the other parts of the protein that were tested. It may be sufficient that the coverstrand be able to form a beta sheet with $\beta 9$. Molecular dynamics simulations [29] suggest that CNB has a conformational bias to move towards the motorhead and that in crystal structures that are in the ADP state and thus do not have a formed CNB, when the missing necklinker and coiled coil helix are added, the same conformational bias of the CNB towards the motor head was seen. The same sort of "autonomous" behavior was seen here. In this study, when the Eg5 coverstrand was used on an otherwise unmodified motor, the rate of the mechanical step was slightly reduced, but remained the same order of magnitude. Thus perhaps that as long as a coverstrand is present to form the CNB, it will fold forward to generate force. While the coverstrand may appear to be more general than some of the other parts of the motor, there is evidence that a matched CNB operates more efficiently. The motors where the coverstrand was matched to the correct $\beta 9$ (WT compared to CS and CS-NL compared to NL) had higher stall forces and longer runs than motors where the coverstrand did not match $\beta 9$. The unloaded velocity was not greatly affected by the matching of the coverstrand to $\beta 9$. This finding is likely due to the fact that the unloaded velocity of the motor is highly dependent upon the catalytic rate of the motor (usually referred to as k_{cat}, called k_3 in the model used here for fitting the force-velocity data), and this rate did not differ significantly between the motors.

3.5.2 The interaction of β9 and L13

As stated above, it has been shown in [75] that the asparagine in the the kinesin 1 L13 interacts with the valine of the necklinker. These two residues are greatly different in Eg5. The major mutation in the necklinker between kinesin 1 and Eg5 is the substitution of proline for valine. In the case of L13, the major mutation is arginine for asparagine. The interruption of this contact may explain why the NL (where the β9 comes from Eg5) and L13 (where L13 comes from Eg5) have the most drastically reduced performance. This suggestion is supported by a study [76] where neighboring residues (that are shown to interact in [75]) were mutated to alanine, which resulted deleterious effects on motor velocity. In that study, the residues that interact that were modified were N327 and T328 (numbering used for human KHC) in the necklinker and the two highly conserved glycine residues in L13, which were mentioned above. When either the NL had the mutations or L13 had the mutations, it resulted in a 10 and 100 fold, respectively, decrease in microtubule velocity in a gliding filament assay. Further it was found that the alanine mutations in L13 caused such severe loss of processivity that stall forces could not be measured [77]. Like in this study, [76] found that the catalytic activity of the mutant motors did not change significantly. Mutant kinesin 1 constructs where the "asparagine latch" (highly conserved asparagine between β9 and β10 where the necklinker docks to the motorhead) was mutated to alanine (V331A / N332A, again using human KHC numbering) showed disruption in the ability to generate force. The mutant was able to walk against a maximum of 3pN while the wildtype protein in the same study was able to walk against 4pN of force. The asparagine latch is conserved among both kinesin 1 and Eg5, and thusly was not mutated in any of the chimeras. Therefore, none of the reductions in force generating capabilities of the chimeras studied here can be attributed to defects in the latching action of the asparagine latch. Interestingly, the NL and L13 chimeras have similar run lengths to the kinesin construct that had the coverstrand deleted [38]. These chimeras had higher stall forces and unloaded velocities though. It is possible that these mutations destabilize the CNB or the docking of the necklinker to the motorhead which results in defects in the gating mechanism between the heads. In the case of the Eg5 necklinker, a proline is present, which is known as a beta sheet breaker. This proline would appear to limit the size of the CNB, and thus potentially its force generation capability.

3.5.3 L13 as a stabilizer

L13 may act to stabilize the powerstroke when the CNB matches the L13. However when the CNB does not match the L13, as in the case of when the the Eg5 L13 was mutated into kinesin 1or when the wildtype kinesin 1 L13 was used with the Eg5 CNB, the L13 may act to destabilize the folding of the CNB toward the motorhead. This is due to the reasons listed above regarding the contacts between $\beta 9$ and L13. In the case of the L13 chimera, the arginine residue in place of the asparagine residue appears to hurt force generation. This might be expected as the Eg5's L13 contains an arginine in place of an asparagine. The arginine is much larger than the asparagine and may interfere with the CNB's fold toward the motorhead and the necklinker's docking to the motorhead. The Eg5 L13 does not appear to significantly affect either unloaded velocity or run length when used with the Eg5 CNB, but it strikingly reduced both the unloaded velocity and run length when used with the kinesin 1 CNB. Interestingly, the characteristic distance for the mechanical step, δ_2 is lower for the chimera that contains the mutated coverstrand, $\beta 9$, and L13 (CS-NL-L13) than the construct containing the mutated coverstrand and $\beta 9$ (CS-NL), which would suggest a decrease in force sensitivity on the mechanical rate (as will be discussed later). In the three state model, as with the Boltzmann model, the characteristic distance is not necessarily the full size of the step that the motor takes. The characteristic distance is usually thought to be less than the step size for kinesin, but in some cases has been found to be greater than the step size [38].

3.5.4 The structural link to stall force

The characteristic distance can be thought of as a measure of force sensitivity, as it is used in the exponential term of the mechanical rate, as seen in equation 3.3. Upon inspection of the unloaded mechanical rate, it is seen that this rate is an order of magnitude faster for the CS-NL chimera than the CS-NL-L13 chimera. The unloaded mechanical rate, k_2^0 for the chimeras with a matched CNB are the fastest as stated above, likely due to fast formation and folding forward of the CNB, however in the case of the CS-NL chimera, the sensitivity to force is high because the Eg5 L13 is not present to stabilize the CNB when it folds forward. In the case of CS-NL-L13, the Eg5 L13 may cause CNB folding to be slower, but in the end stabilizes the CNB when it folds forward, thus producing a less force sensitive mechanical rate. The slower CNB folding (unloaded mechanical rate), may be the source of the lower stall force. The slower mechanical rates of these proteins would not significantly affect the velocity of the motors for most of kinesin's run, as the mechanical rate does not become limiting until the motor is nearly stalled. Generally, the rate of ATP hydrolysis, k_3 is much slower, and thus limits the velocity of the motors. It is not certain why this might be the case, but the motors have a mechanical rate of $24 \pm 4\text{s}^{-1}$ (average plus or minus the standard deviation) at the stall force. The force at which the mechanical rate becomes rate limiting (that is, when it becomes slower than the rate of ATP hydrolysis) is $80 \pm 3\%$ of the stall force. The consistency of the mechanical rate at stall and the fraction of the stall force where the mechanical rate becomes limiting among all of the motor constructs suggests that the mechanical rate and stall force are closely related. Furthermore, when the data for the constructs where two glycines were mutated into the coverstrand and where the coverstrand was deleted [38] were analyzed in this way, it was found that they also had mechanical rates at stall on the order of tens per second and that the force at which the mechanical rate became limiting was above 80%. Similar results were also found with the dimeric Eg5 data of [68, 61]. These results are shown in table 3.7. A hint at the link between the mechanical rate and the stall force may come from kinesin's dissociation rate from

Table 3.7: Rates for the mechanical step of kinesin's mechanochemical cycle. The stall force, unloaded mechanical rate and δ_2 come from table 3.5. The mechanical rate at stall (k_2^{stall} were calculated by using the stall force, k_2^0, and δ_2 for each motor. The force at which the mechanical step becomes limiting, $F_{mechlimiting}$ was calculated by rearranging the second equation in equation 3.3 to solve for force and plugging in k_3, the catalytic rate, in place of k_2.

	F_{stall} (pN)	k_2^0 (s^{-1})	δ_2	k_2^{stall} (s^{-1})	$F_{mechlimiting}$ (pN)	$F_{mechlimiting}$ / F_{stall} (%)
WT	4.92	18400	5.51	25.40	4.08	82.4
CS	3.89	6030	5.65	28.86	3.20	82.3
NL	2.95	5150	7.35	26.47	2.45	83.1
L13	2.79	3000	7.27	21.70	2.24	80.2
CS-NL	3.15	13400	8.69	17.23	2.51	79.7
CS-NL-L13	2.78	1690	6.22	25.21	2.10	75.4
2G [38]	3.02	4830	7.15	39.97	2.47	81.8
DEL [38]	1.37	1710	11.28	25.39	1.22	88.8
Eg5 [68]	4.6 [61]	86	1.9	10.28	4.01	87.2

microtubules. It has been found that this dissociation rate is force dependent, and at forces that are close to these constructs' stall force, the dissociation rate is on the order of $1s^{-1}$ [78]. While this is about an order of magnitude smaller than the mechanical rate at stall, it may be that the slow mechanical rate at stall allows for a higher probability of dissociation from the microtubule before the completion of the full mechanochemical cycle. This link between the mechanical rate of the motor and the stall force may be a way to rationally engineer kinesin motors to have a prescribed stall force. By making the CNB fold forward faster (or have less force sensitivity), it may be possible to make the force at which the mechanical rate becomes limiting higher, and thus increase the stall force.

3.6 Conclusions

The generation of kinesin 1 / Eg5 chimeras using elements from kinesin's proposed force generation mechanism, the CNB (the coverstrand and β9) as well as a loop that is known to interact with β9, L13, have provided additional views into the force generation mechanism of kinesin. This model has been expanded to include effects of L13, which appears to have a stabilizing effect, and that this effect is likely due to contacts between β9 and L13, such as N 327 and T328 of β9 with L290 G291 G292 of L13 (human KHC numbering used) and V329 of β9 with N293 of L13.

Further, by changing the sequences of the components of the CNB and L13, we have been able to change the rate of the mechanical step. This change in mechanical rate may be able to explain the differences in stall force that were observed in this study. The mutation of structural elements of kinesin 1 to that of those containing the sequences from Eg5 have deleterious effect on kinesin force generation capabilities and overall performance, even when all of the elements associated with force generation are mutated to the Eg5 sequences. Comparing these results to the reported force generating capabilities of dimeric constructs of Eg5 show that while the CNB is essential, and that a matched CNB works most efficiently, there must be more to the mechanism of force generation than can simply be explained by the CNB model. It also illustrates that the parts of kinesin are not directly interchangeable, and that the behavior of the motor is more than the sum of its parts, but rather the interaction of these parts with each other in addition to the characteristics of the individual elements.

It is possible that to capture the force generation characteristics of Eg5, a chimera including $\alpha 6$, the alpha helix directly preceding $\beta 9$, needs to also come from Eg5, and that there is a specific way that $\alpha 6$ operates that is unique to Eg5. $\alpha 6$ has been shown to form an extra helical turn prior to CNB formation, thus perhaps the Eg5 CNB requires this part of the Eg5 $\alpha 6$ to operate correctly. A recent study has formed chimeras using kinesin 1 as the main part of the motorhead and adding $\beta 8$, $\alpha 6$, the necklinker and coiled coil from Eg5 [60]. That report showed that the chimera lost activity, however, they did not use a chimera with the Eg5 coverstrand, which as shown here, deeply impacts motor performance. A previous study by the same group [59] also used a chimera that had the Eg5 necklinker with the kinesin 1 motorhead, which they said was not motile. It has been found in this study that a chimera containing just $\beta 9$ of the necklinker is indeed motile, although severely handicapped. Thus, either $\beta 10$ of Eg5 was further deleterious to activity, or there are some problems with the chimeric constructs developed by that group.

It could also be plausible that velocity of each motor has an effect on the apparent stall force. As was very nicely summarized recently [79], the loading rate can markedly affect the rupture force that is measured between molecules. In this case, with Eg5's much slower velocity (when compared to kinesin 1), the loading rate is lower, and thus the stall force that is measured is lower. Under the assumption that the CNB model is complete, and that motorhead gating is the same in the constructs that contain the same necklinkers, it is possible that the slower motion of motors such as the L13 chimera causes an apparent decrease in stall force, due to a lower loading rate. Of course this comes with the caveat that due to dwells that are present in kinesin's stepping motion, especially at high forces, that the loading rate might actually stay the same while the velocity appears to have dropped. The methods and future work for addressing these issues are presented in chapter 5.

Chapter 4

Antibody Inhibition of kinesin activity

4.1 Motivation for kinesin inhibition

Kinesin is the target for a variety of drugs, specifically agents that target proteins associated with mitosis [15, 80, 81, 82, 83, 84, 85]. These drugs are important chemotheraputic agents to arrest mitosis and cease cell division. While the majority of agents that target kinesin for cancer therapy are small molecules, antibodies have been shown to be used to target other proteins in the fight against cancer and other diseases [86, 87]. Antibodies have been generated for targeting kinesin that have the ability to interfere with kinesin's function [88, 89, 90], however these were not specifically targeted to any specific structural element of the motorhead. These antibodies were instead simply raised against the heavy chain. Here we propose to use the CNB mechanism of force generation to design an antibody that inhibit kinesin's motility.

4.2 Results

The antibodies used for this study were generated by Epitomics, Inc using synthesized peptides corresponding to the coverstrand of *D. melanogaster* kinesin 1. Two potential peptide sequences were used (originally specified by Ricardo Gonzalez Rubio), see

Table 4.1: Sequences of synthetic peptide used for immunization of rabbits for polyclonal antibody generation. These sequences correspond to parts of kinesin 1's coverstrand. Version 1 includes the last six residues of the coverstrand plus two glycine residues and cysteine. The cysteine was added to conjugate the peptide to a substrate for immunization, and the glycine residues were added as flexible peptides. Version 2 contains all of the residues of the kinesin 1 coverstrand. As with version 1, a C-terminal cysteine was added for conjugation.

Version 1	-	-	-	-	-	-	-	P	A	E	D	S	I	G	G	C
Version 2	M	S	A	E	R	E	I	P	A	E	D	S	I	C	-	-

Table 4.2: Immunization schedule of the four rabbits used for antibody production. Rabbits were injected with combinations of the peptides shown in table 4.1 four times to illicit an immune response and produce antibodies specific to these peptides.

Rabbit	07/22/2009	08/12/2009	09/02/2009	09/23/2009
E4078	0.4mg	0.2mg	0.2mg	0.2mg
E4079	Version 1 +	Version 1 +	Version 1 +	Version 1 +
E4080	0.4mg	0.2mg	0.2mg	0.2mg
E4081	Version 2	Version 2	Version 2	Version 2

table 4.1. Version 1 includes less of the sequence of the coverstrand and should allow for less specific targeting of the coverstrand. The second version covers the full coverstrand, and was designed so that the antibody would be more specific for the coverstrand. A cysteine residue was added to the C-terminus of each peptide to attach the peptides to a substrate for immunization. Four rabbits were immunized with both version of the peptides with four injections of mixtures of both versions of the peptide in the schedule shown in 4.2 The version 2 peptide was used for affinity purification of the antisera. The rabbits were then bled to obtain antisera that was used to determine which bleeds should be used for antibody purification. It was suggested by Epitomics that testing either bleeds 1 and 3 or 2 and 4 would be satisfactory for purification determination. A western blot was run using bleeds 2 and 4 from each of the four rabbits, which is shown in figure 4-1. From this western blot, it was determined that bleed 4 from rabbits 4078 and 4080 should be used for purification. The purified sera was combined and used for all of the experiments described herein.

The western blot showing that the antibody was only specifically targeted to coverstrands with the WT sequence is shown in figure 4-2. The difference in fluorescence

Table 4.3: Bleed schedule for the production of antibodies. The rabbits were bled four times. Epitomics Inc. sent us the bleeds for determination of which should be used for affinity purification. Bleed 0 was taken as a baseline.

Rabbit	07/21/2009	09/14/2009	09/16/2009	10/05/2009	10/07/2009
E4078 E4079 E4080 E4081	5mL (Bleed 0)	25mL (Bleed 1)	25mL (Bleed2)	25mL (Bleed 3)	25mL (Bleed 4)

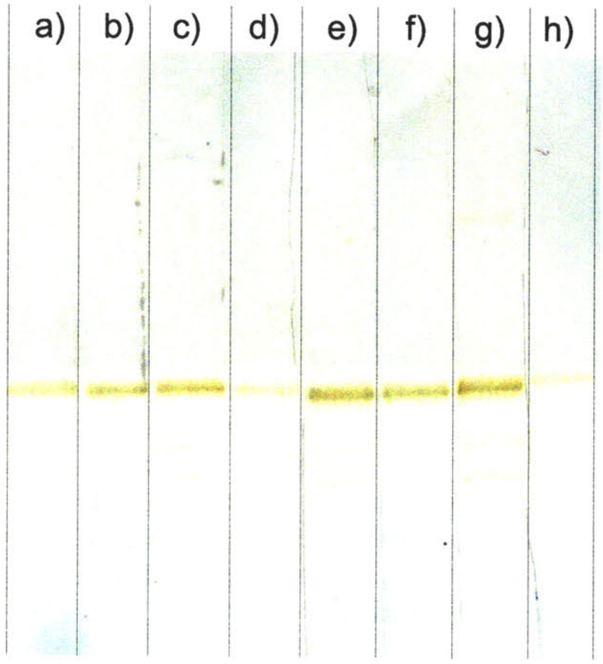


Figure 4-1: Western blot used for determination of bleeds to use for purification. Two bleeds from each of the four rabbits were used, wild type kinesin 1 from *D. melanogaster* was used as the target protein. The lanes are as follows a) bleed 2 from rabbit 4078 b) bleed 2 from 4079 c) bleed 2 from 4080 d) bleed 2 from 4081, e) bleed 4 from 4078 f) bleed 4 from 4079 g) bleed 4 from 4080 h) bleed 4 from 4081. As can be seen most of the bleeds produced good results, except for the bleeds from rabbit 4081.

of the bands is due to unequal loading of kinesin into the gel used for separating the protein. Here it is easily seen that only kinesin constructs in which a kinesin 1 coverstrand exists are targeted by the antibody, and thus become fluorescent. The 2G construct, where two glycine residues were mutated into the coverstrand was also targeted. This was expected as the majority of the residues were the same as the wild type protein, and that the glycines should make the coverstrand more flexible, and thus perhaps make the rest of the residues more easily identified by the antibody.

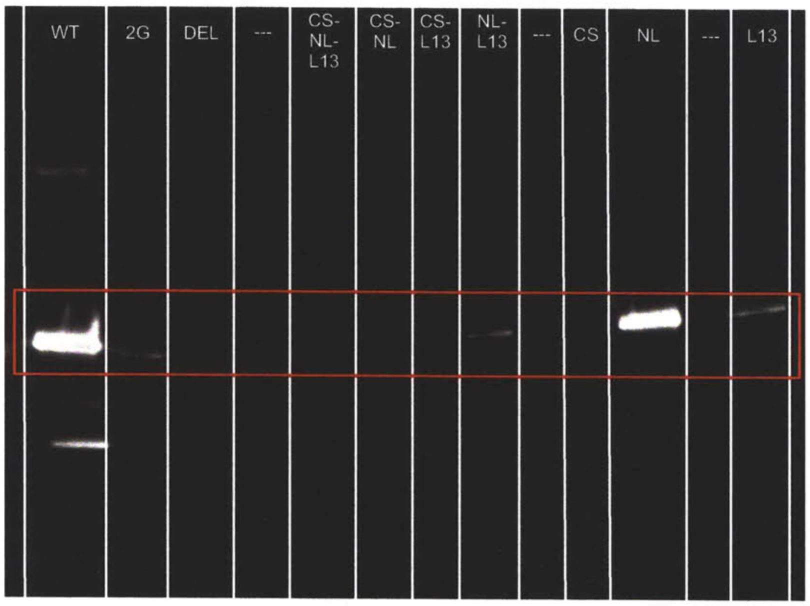


Figure 4-2: Western blot showing the specificity of the antibodies to the kinesin 1 coverstrand of *D. melanogaster*. Only constructs that contain the wildtype coverstrand (WT, NL, L13, NL-L13) show targeting. The construct 2G also was targeted by the antibody, but this was expected as this construct has two residues mutated to glycine, so the majority of the coverstrand contains the wildtype residues, and the glycine residues should act to make the coverstrand more flexible, thus potentially conformally more amenable to detection.

Figure 4-3 shows the concentration dependence on the inhibition of kinesin motion. In this experiment, the kinesin were incubated with the concentration of antibody to be tested for fifteen minutes on ice before use. The concentrations of kinesin used in each of these experiments was slightly above the single molecule limit, where either all or nearly all beads were motile, but not such high concentrations that the beads would simply stick to the microtubule and not move in the absence of antibody. Both the wild type kinesin 1 and CS chimera were used to determine the efficacy of the antibody to inhibit kinesin motion. Only beads that became tethered to the microtubule after being placed next to the microtubule for a few seconds were considered for analysis. Beads that tethered and at some point began running were not considered as being inhibited. Twenty beads were tested at each concentration of antibody. The CS chimera was tested with no antibody and with the highest concentration of antibody that was tested for WT kinesin 1. The CS construct displayed similar motility as in the unloaded assay described in chapter 3, regardless of the concentration of antibody that was present. Thus, no antibody related tethering was noticed with the CS construct, which confirmed that the antibody's effect on kinesin 1's motility is specific to the antibody binding to the coverstrand and not a nonspecific interaction such as interaction with the microtubules or glass slide.

4.3 Discussion

The western blot clearly shows that the antibody can target the kinesin constructs that have the wild type kinesin 1 coverstrand. The concentration dependent inhibition of kinesin motility was fit using

$$y = \frac{[AB]}{[AB] + K_D} \tag{4.1}$$

where y is the fraction of beads that are not motile, and the pseudo-first order approximation is used. Under the pseudo-first order approximation, the concentration of ligand (antibody in this case) is assumed to be in a great enough excess that it can be considered to be constant. The data were also fit with the same relation, but accounting for antibody depletion with the following formula

$$y = \frac{(K_D + [AB] + [kinesin]) - \sqrt{(K_D + [AB] + [kinesin])^2 - 4[kinesin][AB]}}{2[kinesin]} \tag{4.2}$$

Finally the data were also fit using a relation that allows for coopertivity

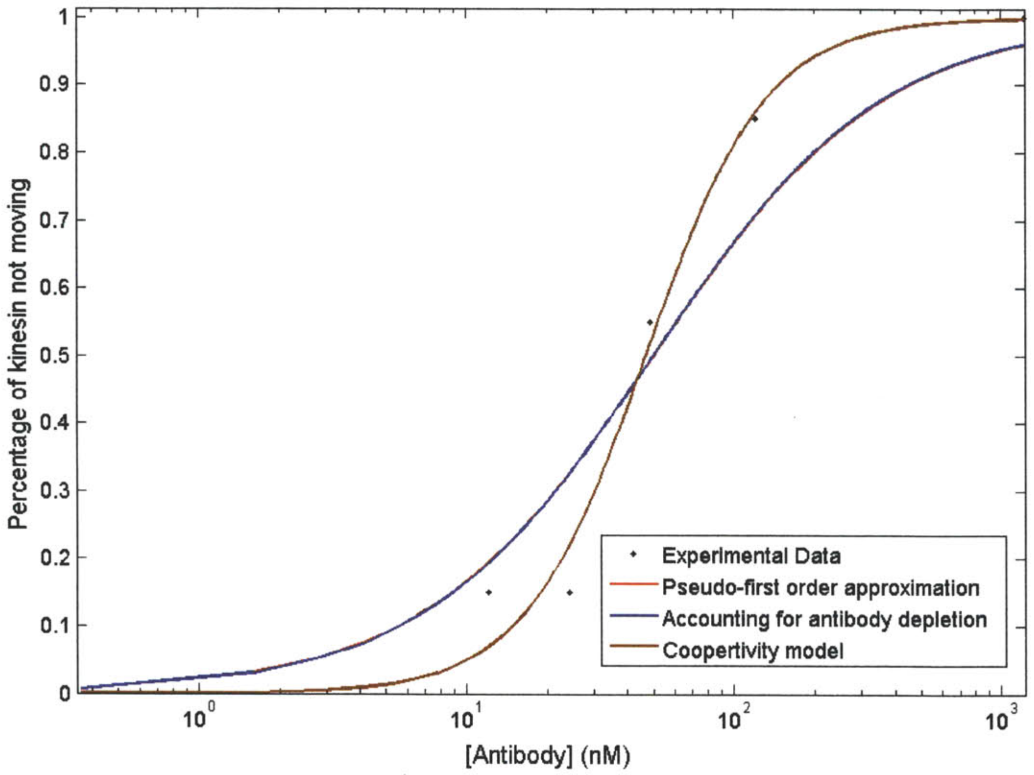


Figure 4-3: Antibody titration curve, which shows the disruption of kinesin 1's motility as a function of the concentration of antibody. Only beads that tethered and did not run at all during the experiment were used for this analysis. The concentration of kinesin used was slightly above the single molecule limit. Fits to the data included a model that does not include coopertivity, but uses the pseudo-first order approximation, where antibody depletion is not accounted for (equation 4.1), a model that does account for antibody depletion, but not coopertivity (equation 4.2), and a model that includes the possibility of coopertivity (equation 4.3). The model allowing for coopertivity fit thè data with the highest fidelity. The estimated equilibrium dissociation constant for the coopertivity model was in the low micromolar range with a coopertivity of nearly two.

Table 4.4: Fit parameters from the antibody titration curve. The two models that do not allow for coopertivity were nearly identical, showing that antibody depletion effects were not significant and the pseudo-first order approximation was valid. The equilibrium dissociation constant found for these noncooperative models was approximately 50nM, which show very good specificity. The coopertivity model fit to a dissociation constant of about 1.6μM, which shows considerably lower affinity. The coopertivity was nearly 2, as could be expected for kinesin, which has two coverstrands per molecule, and thus two binding sites for the antibodies. The coopertivity model fit the data the best. The low K_D could be due to the use of polyclonal antibodies rather than monoclonal antibodies which would be more specific.

	K_D (nM)	n
Pseudo-first order approximation	50.13	N/A
Accounting for ligand depletion	49.42	N/A
Coopertivity model	1593	1.9

$$y = \frac{[AB]^n}{[AB]^n + K_D} \tag{4.3}$$

The parameters determined by these fits are shown in table 4.4. As can be seen the model that allows for coopertivity fits the data the best, and that the depletion of antibody is not significant during the experiment, and thus the pseudo-first order approximation is applicable, as accounting for antibody depletion does not change the fitted equilibrium dissociation constant, K_D. Interestingly, the coopertivity model suggests that two antibodies can bind to kinesin. This result makes sense as there are two motor heads per kinesin molecule, and thus two coverstrands per molecule that can bind antibody. The model that includes coopertivity fits the data most reliably, which suggests that the antibody does not bind very tightly to kinesin, since it has a micromolar equilibrium dissociation constant. This is not completely unexpected, as the antibodies used in this study were polyclonal, and thus less specific than monoclonal antibodies. Future studies can be conducted with monoclonal antibodies, which should increase the specificity.

4.4 Conclusions

The antibody binds to the coverstrand and inhibits the ability for kinesin to move. This appears to be due to the antibody binding to the coverstrand, which then obstructs the formation of the cover neck bundle, thus inhibiting the force generation mechanism of kinesin. It is also known from experiments with the CS chimera, which is identical to the kinesin 1 construct that was tested except for the coverstrand, that the antibody's effect on motility is specific to the CNB formation. Further work must be done to determine the exact mechanism of force generation inhibition in kinesin. The discussion of these further studies can be found in chapter 5. It is expected that the antibody binds to the coverstrand and thus sterically inhibits CNB formation. It is still unknown however, where in the kinesin cycle that the antibody binds to the coverstrand, and how this affects the ATPase cycle of the motor. It would make sense that the antibody would interact with the coverstrand in either the empty or ADP state, as these states correspond to states where the coverstrand is not interacting with $\beta 9$. Since the bead appears to tether as soon as it interacts with the microtubule, and that the kinesin was allowed to incubate with the antibodies for some time before the experiment was started, it is believed that the antibody first binds to the kinesin in the empty state before interaction with the microtubule. This is because the empty state is a strong microtubule binder and the ADP state is not.

Chapter 5

Future Work and Directions

5.1 Chimera study

There are a number of further chimeras that can be made to study the motility mechanism of kinesin. One such chimera would be to insert the sequence for Eg5's L5 into kinesin. This loop has been subject of recent investigation [65]. The coverstrand that was used in this study was the same length of the WT *D. melanogaster* KHC coverstrand, which is about ten amino acids shorter than the WT Eg5 coverstrand. Further chimeras could be generated with the full length Eg5 coverstrand to determine what the impact of these extra residues. The beta sheet formed between the coverstrand and $\beta 9$ is much shorter than the overall length of the coverstrand, so this chimera may not show significant change in force production, but perhaps the extra coverstrand length is related to regulation. The coverstrand could also have all of its residues mutated to alanine. This would determine wether simply having the structural element is enough for force generation, or if there need to be specific contacts made between the coverstrand and $\beta 9$ in order to generate a functional kinesin. To pursue the goal of a motor that can withstand higher forces than kinesin 1, a chimera could be generated that uses kinesin 1's microtubule binding elements and the rest of the Eg5 motor to take advantage of the apparent high force capabilities of Eg5 (which are possibly hampered by Eg5's penchant to release before stalling) and the ability of kinesin 1 to hold on to the microtubule and come to a true stall. One other possible chimera would one that includes some or all of $\alpha 6$ from Eg5 in addition to the matching CNB. A recent report has stated that a chimera using $\alpha 6$ from Eg5 in the context of kinesin 1 was not active, but they did not use the Eg5 CNB, which has been shown here to be necessary reasonable motor function.

A seemingly simple experiment that could be done to determine the effect of loading rate on kinesin's stall force could be done using a force clamp. Generally, a force clamp is used to keep a constant force by maintaining a constant offset between the bead and the trap. Instead of a constant distance, this distance could be modulated during the experiment to cause the load experienced by the motor to change. Similarly this could be done with a fixed trap that was able to modulate the trap stiffness, as was done in [4], though instead of changing the trap stiffness to keep a constant force, this could be changed to produce a desired loading pattern. The effect of the loading pattern on the observed stall force would allow for the determination of the role of loading rate on the stall force of kinesin.

Currently on going is a molecular dynamics investigation of the impact of the mutations on force generation. These simulations are being done in the same manner as in [29]. The force map of the isolated CNB is being generated for each of the chimeras. The molecular dynamics simulation will also be extended to include the L13. From the experimental data generated in this study, it is suggested that L13 can significantly impact the stall force obtained by the motor, and that this reduction in force does not appear to be tied to a "force sensor" mechanism in which the motor dissociates from the microtubule under high force. This is assumed because the distribution of velocities at release from the microtubule for the constructs where L13 was mutated was very similar to that of the wildtype motor. These simulations will help to determine if the stall forces found experimentally can be explained by the CNB model, and to see if there is a way to explain why dimeric Eg5 appears to have high force capabilities, but the chimeras in this study that make use of the Eg5 CNB have reduced force generating capabilities.

Table 5.1: Sequences used for the second batch of antibodies. These sequences correspond to the coverstrand of Eg5. Version 1 is the sequence used for the chimeric kinesin constructs employing the Eg5 coverstrand used in this study. Version 2 is the full length coverstrand from Eg5. As with the first batch of antibodies, a C-terminal cysteine was added for conjugation.

Version 1	-	-	-	-	-	-	M	S	A	K	K	K	E	E	G	K	N	I	C
Version 2	M	A	S	Q	P	N	S	S	A	K	K	K	E	E	G	K	N	I	C

5.2 Antibody Inhibition of kinesin

The antibody generated for this study is specific for the *D. melanogaster* KHC coverstrand. We have ordered a new batch of antibodies that are specific to the coverstrand of *H. sapiens* Eg5. These new antibodies are being generated to be specific for the sequences shown in 5.1. Version 1 is specific for the truncated coverstrand used in the chimeras in this study, and version 2 is specific for the full length Eg5 coverstrand. This second batch of antibodies is intended to show that the inhibition of kinesin's motion by targeting the coverstrand is general, and can be applied to motors other than kinesin 1, as well as showing that the more therapeutically relevant Eg5 motor can be inhibited by specific antibodies.

The impact of these antibodies can also be studied using ATPase assays to determine how the antibody affects kinesin's mechanochemical cycle. It is currently unknown if the antibody locks the motor in the empty state, or if the motor can still bind or hydrolyze ATP while it is attached to the microtubule.

Heterodimers can be generated with one head that has the coverstrand that can be targeted by the antibody and other other head with a coverstrand that cannot be targeted. The antibody titration curve that is observed with these heterodimers will show if there is indeed coopertivity in the binding of antibodies to the motors and inhibition of motion. It would be expected form the results of this study that in the case of heterodimers, the coopertivity should disappear, as only one binding site for the antibody per molecule would be present.

Bibliography

[1] M J Schnitzer and S M Block. Kinesin hydrolyses one atp per 8-nm step. *Nature*, 388(6640):386–90, Jul 1997.

[2] R. D Vale. The way things move: Looking under the hood of molecular motor proteins. *Science*, 288(5463):88–95, Apr 2000.

[3] A Gennerich and RD Vale. Walking the walk: how kinesin and dynein coordinate their steps. *Curr Opin Cell Biol*, 21(1):59–67, 2009.

[4] MD Wang, MJ Schnitzer, H Yin, R Landick, J Gelles, and SM Block. Force and velocity measured for single molecules of rna polymerase. *Science*, 282(5390):902, 1998.

[5] FJ Kull, RD Vale, and RJ Fletterick. The case for a common ancestor: kinesin and myosin motor proteins and g proteins. *J Muscle Res Cell Motil*, 19(8):877–886, 1998.

[6] Nobutaka Hirokawa, Yasuko Noda, Yosuke Tanaka, and Shinsuke Niwa. Kinesin superfamily motor proteins and intracellular transport. *Nat Rev Mol Cell Biol*, 10(10):682–696, Oct 2009.

[7] Megan T Valentine, Polly M Fordyce, and Steven M Block. Eg5 steps it up! *Cell Div*, 1:31, Jan 2006.

[8] I Rayment, HM Holden, M Whittaker, CB Yohn, M Lorenz, KC Holmes, and RA Milligan. Structure of the actin-myosin complex and its implications for muscle contraction. *Science*, 261(5117):58, 1993.

[9] Yongdae Shin, Joseph H Davis, Ricardo R Brau, Andreas Martin, Jon A Kenniston, Tania A Baker, Robert T Sauer, and Matthew J Lang. Single-molecule denaturation and degradation of proteins by the aaa+ clpxp protease. *Proc Natl Acad Sci USA*, 106(46):19340–5, Nov 2009.

[10] Petra Wendler, James Shorter, David Snead, Celia Plisson, Daniel K Clare, Susan Lindquist, and Helen R Saibil. Motor mechanism for protein threading through hsp104. *Molecular Cell*, 34(1):81–92, Oct 2009.

[11] Howard C Berg. The rotary motor of bacterial flagella. *Annu. Rev. Biochem.*, 72(1):19–54, Jul 2003.

[12] M. G. L Van Den Heuvel and C Dekker. Motor proteins at work for nanotechnology. *Science*, 317(5836):333–336, Jul 2007.

[13] Thorsten Fischer, Ashutosh Agarwal, and Henry Hess. A smart dust biosensor powered by kinesin motors. *Nature nanotechnology*, 4(3):162–166, Mar 2009.

[14] S Salinas, LG Bilsland, and G Schiavo. Molecular landmarks along the axonal route: axonal transport in health and disease. *Curr Opin Cell Biol*, 20(4):445–453, 2008.

[15] Dennis Huszar, Maria-Elena Theoclitou, Jeffrey Skolnik, and Ronald Herbst. Kinesin motor proteins as targets for cancer therapy. *Cancer Metastasis Rev*, 28(1-2):197–208, Jun 2009.

[16] Dov Shiffman, Daniel I Chasman, Robert Y.L Zee, Olga A Iakoubova, Judy Z Louie, James J Devlin, and Paul M Ridker. A kinesin family member 6 variant is associated with coronary heart disease in the women's health study. *Journal of the American College of Cardiology*, 51(4):444–448, Jan 2008.

[17] ELF Holzbaur. Motor neurons rely on motor proteins. *Trends Cell Biol*, 14(5):233–240, 2004.

[18] JM Gerdes, EE Davis, and N Katsanis. The vertebrate primary cilium in development, homeostasis, and disease. *Cell*, 137(1):32–45, 2009.

[19] George T Shubeita, Susan L Tran, Jing Xu, Michael Vershinin, Silvia Cermelli, Sean L Cotton, Michael A Welte, and Steven P Gross. Consequences of motor copy number on the intracellular transport of kinesin-1-driven lipid droplets. *Cell*, 135(6):1098–1107, Dec 2008.

[20] C Leduc, F Ruhnow, J Howard, and S Diez. Detection of fractional steps in cargo movement by the collective operation of kinesin-1 motors. *Proceedings of the National Academy of Sciences*, 104(26):10847, 2007.

[21] D Kenneth Jamison, Jonathan W Driver, Arthur R Rogers, Pamela E Constantinou, and Michael R Diehl. Two kinesins transport cargo primarily via the action of one motor: Implications for intracellular transport. *Biophysj*, 99(9):2967–2977, Mar 2010.

[22] MA Welte. Bidirectional transport along microtubules. *Current Biology*, 14(13):R525–R537, 2004.

[23] Steven P Gross. Molecular motors: a tale of two filaments. *Curr Biol*, 17(8):R277–80, Apr 2007.

[24] SP Gross, M Tuma, SW Deacon, AS Serpinskaya, AR Reilein, and VI Gelfand. Interactions and regulation of molecular motors in xenopus melanophores. *The Journal of Cell Biology*, 156(5):855, 2002.

[25] H Miki, M Setou, K Kaneshiro, and N Hirokawa. All kinesin superfamily protein, kif, genes in mouse and human. *Proc Natl Acad Sci USA*, 98(13):7004–11, Jun 2001.

[26] A J Kim and S A Endow. A kinesin family tree. *Journal of Cell Science*, 113 Pt 21:3681–2, Nov 2000.

[27] Kristen J Verhey and Jennetta W Hammond. Traffic control: regulation of kinesin motors. *Nat Rev Mol Cell Biol*, 10(11):765–777, Nov 2009.

[28] RD Vale and RJ Fletterick. The design plan of kinesin motors. *Annu Rev Cell Dev Biol*, 13(1):745–777, 1997.

[29] W Hwang, MJ Lang, and M Karplus. Force generation in kinesin hinges on cover-neck bundle formation. *Structure*, 16(1):62–71, 2008.

[30] G Woehlke and M Schliwa. Walking on two heads: the many talents of kinesin. *Nat Rev Mol Cell Biol*, 1(1):50–8, Oct 2000.

[31] S Sack, J Müller, A Marx, M Thormählen, E M Mandelkow, S T Brady, and E Mandelkow. X-ray structure of motor and neck domains from rat brain kinesin. *Biochemistry*, 36(51):16155–65, Dec 1997.

[32] K Svoboda and S M Block. Force and velocity measured for single kinesin molecules. *Cell*, 77(5):773–84, Jun 1994.

[33] Anatoly B Kolomeisky and Michael E Fisher. Molecular motors: a theorist's perspective. *Annual review of physical chemistry*, 58:675–95, Jan 2007.

[34] Ahmet Yildiz, Michio Tomishige, Ronald D Vale, and Paul R Selvin. Kinesin walks hand-over-hand. *Science*, 303(5658):676–8, Jan 2004.

[35] W Hua, Johnson Chung, and Jeff Gelles. Distinguishing inchworm and hand-over-hand processive kinesin movement by neck rotation measurements. *Science*, 295(5556):844–848, Feb 2002.

[36] Braulio Gutiérrez-Medina, Adrian N Fehr, and Steven M Block. Direct measurements of kinesin torsional properties reveal flexible domains and occasional stalk reversals during stepping. *Proc Natl Acad Sci USA*, 106(40):17007–12, Oct 2009.

[37] Steven M Block. Kinesin motor mechanics: Binding, stepping, tracking, gating, and limping. *Biophysical Journal*, 92(9):2986–2995, Jan 2007.

[38] Ahmad S Khalil, David C Appleyard, Anna K Labno, Adrien Georges, Martin Karplus, Angela M Belcher, Wonmuk Hwang, and Matthew J Lang. Kinesin's cover-neck bundle folds forward to generate force. *Proc Natl Acad Sci USA*, 105(49):19247–52, Dec 2008.

[39] MJ Lang and W Hwang. Motor proteins: kinesin drives with an underhead cam. *Current Biology*, 20(9):R408–R410, 2010.

[40] R B Case, D W Pierce, N Hom-Booher, C L Hart, and R D Vale. The directional preference of kinesin motors is specified by an element outside of the motor catalytic domain. *Cell*, 90(5):959–66, Sep 1997.

[41] R H Wade and F Kozielski. Structural links to kinesin directionality and movement. *Nature Structrual Biology*, 7(6):456–60, Jun 2000.

[42] S Rice, A W Lin, D Safer, C L Hart, N Naber, B O Carragher, S M Cain, E Pechatnikova, E M Wilson-Kubalek, M Whittaker, E Pate, R Cooke, E W Taylor, R A Milligan, and R D Vale. A structural change in the kinesin motor protein that drives motility. *Nature*, 402(6763):778–84, Dec 1999.

[43] Yuichi Taniguchi, Masayoshi Nishiyama, Yoshiharu Ishii, and Toshio Yanagida. Entropy rectifies the brownian steps of kinesin. *Nat Chem Biol*, 1(6):342–7, Nov 2005.

[44] Megan T Valentine and Susan P Gilbert. To step or not to step? how biochemistry and mechanics influence processivity in kinesin and eg5. *Curr Opin Cell Biol*, 19(1):75–81, Feb 2007.

[45] Nicholas R Guydosh and Steven M Block. Backsteps induced by nucleotide analogs suggest the front head of kinesin is gated by strain. *Proc Natl Acad Sci USA*, 103(21):8054–9, May 2006.

[46] Nicholas R Guydosh and Steven M Block. Direct observation of the binding state of the kinesin head to the microtubule. *Nature*, pages 1–5, Feb 2009.

[47] T Mori, RD Vale, and M Tomishige. How kinesin waits between steps. *Nature*, 450(7170):750–754, 2007.

[48] S Rice, Y Cui, C Sindelar, N Naber, M Matuska, R Vale, and R Cooke. Thermodynamic properties of the kinesin neck-region docking to the catalytic core. *Biophysical Journal*, 84(3):1844–1854, Dec 2003.

[49] C. V Sindelar and K. H Downing. An atomic-level mechanism for activation of the kinesin molecular motors. *Proceedings of the National Academy of Sciences*, 107(9):4111–4116, Mar 2010.

[50] A Ashkin. Acceleration and trapping of particles by radiation pressure. *Physical Review Letters*, 24(4):156–159, 1970.

[51] A Ashkin, JM Dziedzic, JE Bjorkholm, and S Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optical angular momentum*, page 196, 1986.

[52] A Ashkin and JM Dziedzic. Optical trapping and manipulation of viruses and bacteria. *Science*, 235(4795):1517–1517, 1987.

[53] K Dholakia, G Spalding, and M MacDonald. Optical tweezers: the next generation. *Physics World*, 15(10):31–36, 2002.

[54] MJ Lang and SM Block. Resource letter: Lbot-1: Laser-based optical tweezers. *American journal of physics*, 71:201, 2003.

[55] Y Harada and T Asakura. Radiation forces on a dielectric sphere in the rayleigh scattering regime. *Optics Communications*, 124(5-6):529–541, 1996.

[56] K Svoboda and SM Block. Biological applications of optical forces. *Annual review of biophysics and biomolecular structure*, 23(1):247–285, 1994.

[57] Joshua W Shaevitz. A practical guide to optical trapping. pages 1–19, Aug 2006.

[58] R Brau, P Tarsa, J Ferrer, P Lee, and M Lang. Interlaced optical force-fluorescence measurements for single molecule biophysics. *Biophysical Journal*, 91(3):1069–1077, Aug 2006.

[59] Nikolina Kalchishkova and Konrad J Böhm. The role of kinesin neck linker and neck in velocity regulation. *Journal of Molecular Biology*, 382(1):127–35, Sep 2008.

[60] Nikolina Kalchishkova and Konrad J Bohm. On the relevance of the core helix alpha 6 to kinesin activity generation. *Biophysical Reviews and Letters*, 4(1 & 2):63–75, Jun 2009.

[61] Stefan Lakämper, Christina Thiede, André Düselder, Stefanie Reiter, Mikhail J Korneev, Lukas C Kapitein, Erwin J G Peterman, and Christoph F Schmidt. The effect of monastrol on the processive motility of a dimeric kinesin-5 head/kinesin-1 stalk chimera. *Journal of Molecular Biology*, 399(1):1–8, May 2010.

[62] C Thiede, S Lakämper, A D Weßel, S Reiter, and C F Schmidt. Tetrameric chimera dk4mer is a tool to study mechanisms of kinesin-5 regulation. a tetrameric chimera of a kinesin 1 and a kinesin 5 is a fast microtubule sliding motor. *Biophysical Journal*, 98(3):165a, Jan 2010.

[63] CV Sindelar, MJ Budny, S Rice, N Naber, R Fletterick, and R Cooke. Two conformations in the human kinesin power stroke defined by x-ray crystallography and epr spectroscopy. *Nature Structural & Molecular Biology*, 9(11):844–848, 2002.

[64] Andrew J Bodey, Masahide Kikkawa, and Carolyn A Moores. 9-ångström structure of a microtubule-bound mitotic motor. *J Mol Biol*, 388(2):218–224, Jan 2009.

[65] Adam G Larson, Nariman Naber, Roger Cooke, Edward Pate, and Sarah E Rice. The conserved l5 loop establishes the pre-powerstroke conformation of the kinesin-5 motor, eg5. *Biophysj*, 98(11):2619–2627, Feb 2010.

[66] Z Maliga, TM Kapoor, and TJ Mitchison. Evidence that monastrol is an allosteric inhibitor of the mitotic kinesin eg5. *Chemistry & biology*, 9(9):989–996, 2002.

[67] Mikhail J Korneev, Stefan Lakämper, and Christoph F Schmidt. Load-dependent release limits the processive stepping of the tetrameric eg5 motor. *Eur Biophys J*, 36(6):675–81, Jul 2007.

[68] Megan T Valentine, Polly M Fordyce, Troy C Krzysiak, Susan P Gilbert, and Steven M Block. Individual dimers of the mitotic kinesin motor eg5 step processively and support substantial loads in vitro. *Nature Cell Biology*, 8(5):470–476, May 2006.

[69] E Berliner, E C Young, K Anderson, H K Mahtani, and J Gelles. Failure of a single-headed kinesin to track parallel to microtubule protofilaments. *Nature*, 373(6516):718–21, Feb 1995.

[70] M E Fisher and A B Kolomeisky. The force exerted by a molecular motor. *Proc Natl Acad Sci USA*, 96(12):6597–602, Jun 1999.

[71] M E Fisher and A B Kolomeisky. Simple mechanochemistry describes the dynamics of kinesin molecules. *Proc Natl Acad Sci USA*, 98(14):7748–53, Jul 2001.

[72] SM Block, CL Asbury, JW Shaevitz, and MJ Lang. Probing the kinesin reaction cycle with a 2d optical force clamp. *Proceedings of the National Academy of Sciences*, 100(5):2351–2356, 2003.

[73] C M Coppin, D W Pierce, L Hsu, and R D Vale. The load dependence of kinesin's mechanical cycle. *Proc Natl Acad Sci USA*, 94(16):8539–44, Aug 1997.

[74] M J Schnitzer, K Visscher, and S M Block. Force production by single kinesin motors. *Nature Cell Biology*, 2(10):718–23, Oct 2000.

[75] Ryo Nitta, Yasushi Okada, and Nobutaka Hirokawa. Structural model for strain-dependent microtubule activation of mg-adp release from kinesin. *Nat Struct Mol Biol*, 15(10):1067–75, Oct 2008.

[76] R B Case, S Rice, C L Hart, B Ly, and R D Vale. Role of the kinesin neck linker and catalytic core in microtubule-based motility. *Curr Biol*, 10(3):157–60, Feb 2000.

[77] Kuniyoshi Kaseda, Hideo Higuchi, and Keiko Hirose. Coordination of kinesin's two heads studied with mutant heterodimers. *Proc Natl Acad Sci USA*, 99(25):16058–63, Dec 2002.

[78] K S Thorn, J A Ubersax, and R D Vale. Engineering the processive run length of the kinesin motor. *The Journal of Cell Biology*, 151(5):1093–100, Nov 2000.

[79] ME Aubin-Tam, DC Appleyard, E Ferrari, V Garbin, OO Fadiran, J Kunkel, and MJ Lang. Adhesion through single peptide aptamers. *The Journal of Physical Chemistry A*, page 2672, 2011.

[80] K. M.R Bhat and V Setaluri. Microtubule-associated proteins as targets in cancer chemotherapy. *Clinical Cancer Research*, 13(10):2849–2854, May 2007.

[81] Kuan Yoow Chan, Keith R Matthews, and Klaus Ersfeld. Functional characterisation and drug target validation of a mitotic kinesin-13 in trypanosoma brucei. *PLoS Pathog*, 6(8):e1001050, Aug 2010.

[82] S DeBonis, DA Skoufias, L Lebeau, R Lopez, G Robin, RL Margolis, RH Wade, and F Kozielski. In vitro screening for inhibitors of the human mitotic kinesin eg5 with antimitotic and antitumor activities. *Molecular cancer therapeutics*, 3(9):1079, 2004.

[83] E Koller, S Propp, H Zhang, C Zhao, X Xiao, MY Chang, SA Hirsch, PJ Shepard, S Koo, and C Murphy. Use of a chemically modified antisense oligonucleotide library to identify and validate eg5 (kinesin-like 1) as a target for antineoplastic drug development. *Cancer Research*, 66(4):2059, 2006.

[84] Frank Kozielski, Dimitrios A Skoufias, Rose-Laure Indorato, Yasmina Saoudi, Peter R Jungblut, Hanne K Hustoft, Margarita Strozynski, and Bernd Thiede. Proteome analysis of apoptosis signaling bys-trityl-l-cysteine, a potent reversible inhibitor of human mitotic kinesin eg5. *Proteomics*, 8(2):289–300, Jan 2008.

[85] R Nakai, S.-I Iida, T Takahashi, T Tsujita, S Okamoto, C Takada, K Akasaka, S Ichikawa, H Ishida, H Kusaka, S Akinaga, C Murakata, S Honda, M Nitta, H Saya, and Y Yamashita. K858, a novel inhibitor of mitotic kinesin eg5 and antitumor agent, induces cell death in cancer cells. *Cancer Research*, 69(9):3901–3909, May 2009.

[86] Ole Henrik Brekke and Inger Sandlie. Therapeutic antibodies for human diseases at the dawn of the twenty-first century. *Nat Rev Drug Discov*, 2(1):52–62, Jan 2003.

[87] M Harris. Monoclonal antibodies as therapeutic agents for cancer. *The lancet oncology*, 5(5):292–302, 2004.

[88] S T Brady, K K Pfister, and G S Bloom. A monoclonal antibody against kinesin inhibits both anterograde and retrograde fast axonal transport in squid axoplasm. *Proc Natl Acad Sci USA*, 87(3):1061–5, Feb 1990.

[89] A L Ingold, S A Cohn, and J M Scholey. Inhibition of kinesin-driven microtubule motility by monoclonal antibodies to kinesin heavy chains. *The Journal of Cell Biology*, 107(6 Pt 2):2657–67, Dec 1988.

[90] V A Lombillo, C Nislow, T J Yen, V I Gelfand, and J R McIntosh. Antibodies to the kinesin motor domain and cenp-e inhibit microtubule depolymerization-dependent motion of chromosomes in vitro. *The Journal of Cell Biology*, 128(1-2):107–15, Jan 1995.

Appendix A

Protocols

A.1 Kinesin expression and purification

A.1.1 Materials

1. LB broth (with 100μg/mL ampicillin + 25μg/mL chloramphenicol)

2. LB agar plates (with 100μg/mL ampicillin + 25μg/mL chloramphenicol)

3. TB broth, Add 47.6g TB (Difco Terrific Broth) and 4mL glycerol into 1L deionized water and autoclave. Once cooled add ampicillin, chloramphenicol and biotin to final concentrations of 100μg/mL, 25μg/mL and 100μM (24mg), respectively.

4. 1M IPTG, prepared in water and stored at -20°C

5. Rifampicin, prepared 20mM (16.5mg/mL) in methanol, 100X stock stored at -20°C

6. Lysis buffer, 20mM imidazole, 4mM $MgCl_2$, pH 7 (0.680g imidazole, 0.408mL 4.9M $MgCl_2$ for 500mL)

7. β-mercaptoethanol

8. PMSF, (Sigma-Aldrich P7626), 200mM in isopropanol, stored at -20°C

9. Pepstatin A, (Sigma-Aldrich P4265), 5mg/mL in DMSO, stored at -20°C

10. TPCK, (Sigma-Aldrich T4365), 10mg/mL in DMSO, stored at -20°C

11. TAME, (Sigma-Aldrich T4626), 40mg/mL in deionized water, stored at -20°C

12. Leupeptin, (Sigma-Aldrich L9875), 5mg/mL in deionized water, stored at -20°C

13. DNAse I (Sigma-Aldrich D4527), Grade II

14. RNAse A (Sigma-Aldrich R5000), Type II-A

15. Ni-NTA Resin, (Qiagen Ni-NTA Superflow)

16. TCEP, (Molecular Probes T-2566), 10mM in deionized water, prepared fresh before use

17. Vivaspin 15 spin column, (Vivascience VS1522), 30,000 MWCO

18. Protease Inhibitor Cocktail, PI, prepare 4mL of PI and store at -20°C. Contains: 160μL 0.2mg/mL Pepstatin A, 800μL 2mg/mL TPCK, 200μL 2mg/mL TAME, 160μL 0.2mg/mL Leupeptin, 2$mu$L 2mg/mL Soybean IT, 1880 μL deionized water

19. Econo-Column Chromatography Columns, (Bio-Rad 737-1512), 1.5x10cm, 18mL

20. nuPAGE 4-12% Bis-Tris Gels, (Invitrogen NP0321BOX), 1mm x 10 well

21. Kinesin Storage Buffer, 50mM imidazole, 100mM $NaCl_2$, 1mM $MgCl_2$, 20μM ATP, 0.1mM EDTA, 5% sucrose, pH 7

A.1.2 Day 0

1. Streak fresh colonies on LB-agar plates containing ampicillin and chloramphenicol from frozen glycerol stocks stored at -80°C

2. Incubate upside down (agar at the top) at 37°C overnight

A.1.3 Day 1

1. Pick a single colony and add to 20mL of LB (with ampicillin and chloramphenicol) in a 250mL flask

2. Shake overnight at 37°C

A.1.4 Day 2

1. Inoculate 500mL of TB broth (with ampicillin and chloramphenicol) with 10mL of the overnight LB culture

2. Shake at 37°C

3. Induce expression at OD_{600} = 0.53 - 0.60 by adding IPTG to a final concentration of 1mM

4. Upon induction, lower shaker temperature to 22°C and shake overnight

A.1.5 Day 3

1. Centrifuge the cells at 5,000g and 4°C for 10 minutes

2. While centrifuging, add β-mercaptoethanol (to a final concentration of 10mM), 1/100 volume of PI, and 1/100 volume of PMSF to lysis buffer to make full lysis buffer. Make 5mL for each 500mL culture of cells

3. After centrifugation, drain the supernatant and resuspend the pellet in 5mL of lysis buffer

4. Pipette the resuspended cells into a 15mL centrifuge tube and incubate on ice for 30 minutes to allow the internal lysozyme to degrade the cell walls

5. Flash freeze the 15mL tube in liquid nitrogen and store overnight at -80°C

A.1.6 Day 4

1. Thaw frozen cells with alternating incubations in a 37°C water bath and ice (1 minute in each, do not let the lysate warm up). Once completely thawed, flash freeze in liquid nitrogen. This process is repeated for a total of three thaws. After the last thaw, the lysate should be very viscous.

2. Add 500μL 10mg/mL RNAse (final concentration 1mg/mL) and 250μL 10mg/mL DNAse (final concentration 0.5mg/mL). Incubate on ice for 30 minutes with occasional mixing by inversion. The viscosity should substantially decrease

3. Centrifuge at 21,800g and 4°C for 20 minutes and retain the low speed supernatant. This step pellets out cellular debris

4. Centrifuge at 180,000g and 4°C for 30 minutes and retain the high speed supernatant

5. Add 2mL of Ni-NTA equilibrated in full lysis buffer. To equilibrate Ni-NTA, wash the resin in full lysis buffer three times by centrifuging at 10,000g and 4°C for 10 minutes to pellet the resin. Remove supernatant and wash with full lysis buffer

6. Incubate the resin high speed supernatant mixture at 4°C overnight with mixing by inversion

A.1.7 Day 5

1. Equilibrate the chromatography column by washing with 10mL of full lysis buffer

2. Prepare 100mL of elution buffer 1 and 2 (elution buffer 1 is the same as lysis buffer, but with β-mercaptoethanol, add 7-μL of β-mercaptoethanol to 100mL of lysis buffer). To make elution buffer 2, add 3.268g of imidazole to 100mL of lysis buffer and adjust the pH to 7 with HCl, add 70μL of β-mercaptoethanol

Final [imidazole] (mM)	Elution Buffer 1 (mL)	Elution Buffer 2 (mL)
70	8.96	1.04
100	8.33	1.67
150	7.29	2.71
200	6.25	3.75
500	0	10

3. Load Ni-NTA high speed supernatant mixture into the column and collect the flow through
4. Wash five times with 10mL of lysis buffer
5. After washing, run the imidazole gradient with increasing concentration of imidazole
6. Run samples from the flow through, washes, and gradient on an SDS-PAGE gel to determine which fractions contain kinesin and should be combined. Pool these fractions
7. Concentrate and buffer exchange the pooled fractions into kinesin storage buffer with a vivaspin concentrator, centrifuge at 4°C
8. Aliquot the concentrated kinesin solution into 10μL volumes and flash freeze in liquid nitrogen. Store at -80°C

A.2 Microtubule polymerization

A.2.1 Materials

1. PEM80, 80mM Pipes, 1mM EGTA, 4mM $MgCl_2$, pH adjusted to 6.9 with KOH
2. PEM104, 103.6mM Pipes, 13mM EGTA 6.3mM $MgCl_2$, pH adjusted to 6.9 with KOH
3. STAB, 34.1μL PEM80, 5μL 10mM GTP stock (Cytoskeleton BST06), 4.7μL 60g/L NaN_3, 1.2μL 10mM Taxol stock (Cytoskeleton TXD01), 5μM DMSO

4. Tubulin (Cytoskeleton T237 (now discontinued), replaced with TL238)

A.2.2 Polymerization

1. Centrifuge tubulin aliquot at 10,000g and 4°C for 30 minutes
2. Combine 15.2μL PEM104 and 2μL 10mM GTP to make PEM/GTP
3. Combine 15.2μL PEM/GTP with 2.2μL DMSO and vortex to mix. Add 4.8μL of 10mg/mL tubulin to make TUB
4. Place TUB in a water bath set to 37°C for 30 minutes
5. Remove TUB from the water bath and add 2μL of STAB
6. Store polymerized microtubules at room temperature, NOTE: reconstituted tubulin looses its polymerization ability after about a month at -80°C

A.3 Coverslide Preparation

A.3.1 Materials

1. KOH pellets
2. 200 proof ethanol
3. Deionized water
4. Poly-l-lysine solution (Sigma-Aldrich P8920)

A.3.2 KOH Etching

1. Add 100g of KOH pellets to 300mL of ethanol (in a 1,000mL beaker), use a magnetic stir bar to mix
2. Fill two 1,000mL beakers with 300mL deionized water and one with 300mL of ethanol

3. Using a bath sonicator, degas all of the solutions for five minutes
4. Place coverslips in teflon racks and immerse one rack at a time in the KOH solution, sonicate for five minutes
5. Rinse etch slide in one of the beakers of ethanol, then in deionized water
6. Sonicate the rinsed slides in deionized water
7. Rinse slides with deionized water in using a squeeze bottle
8. Rinse slides with ethanol in using a squeeze bottle
9. Dry slides in an oven for 30 minutes

A.3.3 Poly-lysine coating

1. Add 1mL of poly-l-lysine solution to 300mL of ethanol
2. Place a rack of KOH etched coverslides into the poly-lysine ethanol solution
3. Incubate at room temperature for 15 minutes
4. Dry in an oven for 15 minutes

A.4 Kinesin motility assay

A.4.1 Materials

1. PEM80
2. Phosphate buffered saline (PBS)
3. Taxol stock (10mM in DMSO, stored at -20°C)
4. DTT, 0.5M in 10mM potassium acetate (stored at -20°C)
5. ATP (100mM in PEM80, stored at -80°C)

6. Potassium acetate (3M, stored at 4°C)

7. Casein (10mg/mL in PBS with 0.1% tween 20 (PBST), made fresh the day of the experiment, filtered using a vacuum filter)

8. Kinesin stock (stored at -80°C)

9. Streptavidin coated beads

10. Poly-lysine coated KOH etched coverslides

11. Double sided sticky tape

12. Glucose oxidase (100X stock (Calbiochem 345386), 25mg/mL in PBST, stored at -80°C)

13. β-D-glucose (100X stock, 500mg/mL in PBST, stored at -80°C)

14. Catalase (100X stock (Calbiochem 219261), 3mg/mL in PBST, stored at -80°C)

A.4.2 Assay preparation

1. Make PemTax: Add 1,000μL PEM80 and 2μL Taxol, store at room temperature

2. Make assay buffer (AB, final concentrations 0.1mM DTT, 20μM Taxol, 0.2mg/mL Casein, 1mM ATP, 50mM potassium acetate), store on ice

   (a) 2,848μL PEM80

   (b) 6μL 0.5M DTT

   (c) 6μL 10mM Taxol

   (d) 30μL 100mM ATP

   (e) 50μL 3M potassium acetate

   (f) 60μL 10mg/mL casein

3. Make C-Tax: 80μL PemTax and 20μL 10mg/mL casein, store on ice

4. Made bead dilutions

    (a) Dilute 20μL of 0.44μm streptavidin coated beads into 80μL of PBS

    (b) Wash beads five times with PBS by centrifuging at 10,000 RPM for 6 minutes, discarding the supernatant and resuspending in 100μL of PBS

    (c) Sonicate the beads twice for two minutes using a cup horn sonicator filled with water and ice

    (d) Make EM/AB by adding 8μL of washed and sonicated beads to 392μL of AB, store on ice

5. Make kinesin dilutions

    (a) K/100: 2μL kinesin stock into 98μL AB (note, this is actually twice as concentrated as the label suggests, but will be come correct upon adding beads in subsequent steps)

    (b) K/1000: 10μL of K/100 into 90μL AB

    (c) Continue diluting in this manner until experiments show that half or less of the beads move at a given dilution, sometimes K/10^{7-8} were necessary. Store all dilutions on ice

6. Make Kinesin + Bead dilutions (KDB/###)

    (a) KDB/100: 50μL EM/AB added to 50μL K/100

    (b) Continue making these dilutions for the kinesin concentrations to be tested

    (c) Incubate for 1 hour at 4°C

7. Make MT/150 by adding 1μL of polymerized microtubules to 149μL PemTax, do NOT place on ice

8. Make flow cells for the assay

    (a) Place two pieces of double sided tape perpendicular to the long axis of a thick glass slide

(b) Place a poly-lysine coated coverslide on top of the tape to make a chamber

(c) Flow 15μL of MT/150 and allow to incubate at room temperature for 10 minutes

(d) Wash the chamber with 20μL of PemTax

(e) Flow in 15μL of C-Tax and incubate at room temperature for 5 minutes

(f) Wash with 50μL of PemTax followed by 80μL of AB

(g) Flow in 20μL of the KDB dilution to be assayed

A.4.3 Stall Force Assay

1. Turn on the trapping and detection lasers
2. Turn on the AOD amplifier
3. Load the flow cell slide containing the kinesin sample to be assayed
4. Turn on monitors and camera controller
5. Make adjustments with the microscope (focus, condenser height, filter position) to image the microtubules and beads
6. Unblock the trapping laser
7. Run the VI to initialize the AODs
8. Test beads for movement

   (a) Trap a diffusing bead

   (b) Hold the bead near a microtubule for a few seconds, if it moves, go on, if not, then try the bead on two different microtubules to test for motility

      i. For moving beads, run AOD line sweep which is used to align the AODs with the position detection system. Adjust the micrometers for the detection branch to bring the AODs and detection branch into alignment

ii. Once aligned, run the calibration VI, which sets the calibration for conversion between QPD voltage and nanometer space, as well as running the calibration for trap stiffness, which uses the variance method

iii. After calibration, start the VI to record the voltage signals from the VI

iv. Place the bead near a microtubule, as before, and record the movement of the bead

A.4.4 Unloaded Assay

1. Turn on the trapping and detection lasers
2. Turn on the AOD amplifier
3. Load the flow cell slide containing the kinesin sample to be assayed
4. Turn on monitors and camera controller
5. Make adjustments with the microscope (focus, condenser height, filter position) to image the microtubules and beads
6. Unblock the trapping laser
7. Run the VI to initialize the AODs
8. Test beads for movement

   (a) Trap a diffusing bead

   (b) Hold the bead near a microtubule for a few seconds, if it moves, go on, if not, then try the bead on two different microtubules to test for motility

      i. For moving beads, run AOD line sweep which is used to align the AODs with the position detection system. Adjust the micrometers for the detection branch to bring the AODs and detection branch into alignment. This does not have to be as exact as with the stall force assay, it is just to ensure that the voltage signal used to actuate the trapping laser's shutter is not anomalous

ii. Place the bead near a microtubule, as before, and run the VI to shutter the trapping laser after movement of kinesin, the switch that switches between foot switch actuation and computer actuation needs to be switched to computer control

iii. Once the bead begins to run, the trapping laser will be shuttered and the bead will be free to walk on the microtubule. After the bead dissociates from the microtubule, un-shutter the trapping laser using the VI and try to recapture the bead for further runs

Appendix B

Code

B.1 Stall Force Data Analysis

B.1.1 Run Analysis

```
% Before starting program: (1) Make directory called "events"
%                          (2) Make subdirectories called "stalls" &
%                                                             "discards"

function KEval
clear; clc; close all

Vfile = 'QPD Voltage Data.txt'; %Run data
Calfile = 'Fits from fit V to nm.txt'; %21 fit parameters for 5th order fit
StiffFile = 'Variance data for stiffness calibration.txt'; %trace for var

startEvent = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Set MT Direction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
direct=1; % Either +1 or -1 to make displacements positive

% Threshold force
```

```
fcut=0.7;

% Time ahead of the event that is plotted out (sec).
time_ahead = 5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Read in data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Read in for the XY plot version
%Read in data file
tdata=dlmread(Vfile);
len1=length(tdata(:,1));
tme=2000; % Acquisition rate


%Create matrix with time, Voltage X, Voltage Y
t=0:1/tme:(1/tme*(len1-1));
x=tdata(1:len1,1);
y=tdata(1:len1,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Converts the Voltage to Position, then multiplies by stiffness to get
%   Force.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rawnmdata=ConvertVtoNM(x,y,Calfile);
rawnmdata(:,1:2)=rotateCoords(rawnmdata(:,1:2),-45);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Plot the position data and allow user to select a representative
%   baseline segment.  Subtract the baseline off so that it is "zeroed"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
figure
plot(1:length(rawnmdata(:,1)),rawnmdata(:,1),'-r',1:...
    length(rawnmdata(:,2)),rawnmdata(:,2),'-b')

xlabel('Index')
ylabel('Position (nm)')
legend('X','Y')
axis tight

fh(1)=gcf;
xlims=get(gca,'XLim'); %get the new x limits

%Create cursors
cnums(1)=CreateCursor(fh(1),'BG 1',[26/255 133/255 5/255]);
cnums(2)=CreateCursor(fh(1),'BG 2',[26/255 133/255 5/255]);

SetCursorLocation(cnums(1),(xlims(2)-xlims(1))/3);
SetCursorLocation(cnums(2),2*(xlims(2)-xlims(1))/3);

disp('Place cursors to indicate desired regions (BG = Background)');
input('Hit enter once you are satisfied with your selection');

for iCnt=1:2
    cps(iCnt)=floor(GetCursorLocation(cnums(iCnt)));
end

xadj=mean(rawnmdata(cps(1):cps(2),1));
yadj=mean(rawnmdata(cps(1):cps(2),2));
rawnmdata(:,1)=(rawnmdata(:,1)-xadj)*direct;
rawnmdata(:,2)=(rawnmdata(:,2)-yadj)*direct;

disp(['Adjusting X baseline by ', num2str(xadj)]);
disp(['Adjusting Y baseline by ', num2str(yadj)]);
close(fh(1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%   Decimating the nmdata and time data
%   Decimation runs a chebyzhev filter on the data - boxcar might be
%   better?
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

decval=50;  %A 20 decimation should take this to a "100Hz" sampling
nmdata(:,1)=decimate(rawnmdata(:,1),decval);
nmdata(:,2)=decimate(rawnmdata(:,2),decval);
nmt=decimate(t,decval)';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Read in the stiffness file, convert to nm space, and calculate
%   stiffness from the variance method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Parameters
kb=1.3806503*10^-23; %J/K
T=300; %K

stifftrace=dlmread(StiffFile);
stiff_pos_data=ConvertVtoNM(stifftrace(:,1),stifftrace(:,2),Calfile);
stiff_pos_data=rotateCoords(stiff_pos_data,-45);

varx=var((stiff_pos_data(:,1)-mean(stiff_pos_data(:,1)))*10^-9);
kxvar=(kb*T/(varx)*10^3);%pN/nm

vary=var((stiff_pos_data(:,2)-mean(stiff_pos_data(:,2)))*10^-9);
kyvar=(kb*T/(vary)*10^3);%pN/nm

disp(['X stiffness is: ',num2str(kxvar),' pN/nm'])
disp(['Y stiffness is: ',num2str(kyvar),' pN/nm'])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Convert run data to force
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
fdata(:,1)=nmdata(:,1)*kxvar;
fdata(:,2)=nmdata(:,2)*kyvar;


% Rotates data if is off the 45 degree axis by using a least-squares fit.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This returns a subplot of X vs. Y after rotation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
odeg=FindMTRotationBoxcar(nmdata(:,1),nmdata(:,2));


%Reset data to account for MT orientation
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Makes a linear ajustment in position and force with the rotation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdata(:,1:2)=rotateCoords(fdata(:,1:2),-1*odeg);
nmdata(:,1:2)=rotateCoords(nmdata(:,1:2),-1*odeg);

rotrawnmdata(:,1:2)=rotateCoords(rawnmdata(:,1:2),-1*odeg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Creates yvel, or velocity in the paralell to MT position.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dt=nmt(2)-nmt(1);
yvel=diff(nmdata(:,2))/dt;
btnum=length(nmdata(:,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   The release point is defined by having a high "snapback" value (>2000
%   nm/s), a high force value (1 pN), and that points are not next to
%   one another (an anomoly mostly due to averaging).  Finally, the
%   immediate preceding velocity is looked at to determine if slowing down.
%   For WT, %   this value is 200 nm/s, or roughly 1/2 max.  The analysis
%   went back 10 %   points, or 0.1 sec.  This is enough for 1-2 steps to
%   occur.  For the heterodimer the max velocity will be 20 nm/s, going
%   back 100 points (1s).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Snapback velocity cutoff
sncut=500;

%Find points that are above the force and snapback cutoff
releasepoint=(abs(yvel(:))>sncut) & (abs(fdata(1:btnum-1,2))>fcut);

%Assures that no release points are next to each other
releasepointp=[logical(0) releasepoint(1:length(releasepoint)-1)'];
releasepoint=logical(releasepoint-(releasepoint&releasepointp'));

rpindex=find(releasepoint);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Readjust the release points to occur at the maximum force near the
%calculated release point (sometimes these seem to occur in the midst of
%a drop (may want to change the force cutoff)

%window to look for maximum force +/-
fwindow=10;
for icnt=1:length(rpindex)
    ind=rpindex(icnt);
    if (ind-fwindow)>0 && (ind+fwindow)<length(releasepoint)
        [mxval,mxind]=max(fdata((ind-fwindow):(ind+fwindow),2));
    end
    if fdata(ind,2) < mxval
        releasepoint(ind)=logical(0);
        releasepoint(ind + mxind - fwindow)=logical(1);
    end
end
rpindex=find(releasepoint);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Plot out the force vs time curve and show where all the peaks are
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
plot(nmt(:),fdata(:,1),'r',nmt(:),fdata(:,2),'b',...
    nmt(releasepoint),fdata(releasepoint,2),'ok')
xlabel('Time (s)')
ylabel('Force (pN)')
hold
for iInt=1:length(rpindex)
    text(nmt(rpindex(iInt)),fdata(rpindex(iInt),2)+0.7,num2str(iInt));
end
title(['Force vs. Time - ',Vfile])
axis tight

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Extracting individual peaks.
%   Find data where force is above the fcut/2
%   Pull data where force is above the fcut/2 and has a release point
%   Take X points before and after this region (to get baselines)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

above_force=abs(fdata(1:btnum-1,2))<fcut;
af_index=find(above_force);

pts_ahead=floor(time_ahead/dt);
pts_behind=floor(.5/dt); %look 5/10ths of a second after

for iCnt=startEvent:length(rpindex)
    i_start=af_index(find(af_index<rpindex(iCnt),1,'last'))-pts_ahead;
    if i_start ≤ 0
        i_start = 1;
    end
    i_start=i_start*decval;
    p_start=pts_ahead*decval;
    i_end=af_index(find(af_index≥rpindex(iCnt),1))+pts_behind;
    if i_end>length(nmt)
        i_end=length(nmt);
    end
    i_end=i_end*decval;
    p_end=(i_end-i_start)-pts_behind*decval;
    if ¬(isempty(i_start)|isempty(i_end))
        disp(['Evaluating event #',num2str(iCnt)])
        ecounter(iCnt,1)=iCnt;
        ecounter(iCnt,2)=i_start;
        ecounter(iCnt,3)=i_end;
        [FV keep]=EvalEvent(t(i_start:i_end)',rawnmdata(i_start:i_end,...
            :),[kxvar kyvar],[p_start p_end],odeg,direct,...
            [Vfile(1:length(Vfile)-4),'_Event_',num2str(iCnt)]);
    else
        ecounter(iCnt,:)=[iCnt 0 0];
    end
end

%% PC
dlmwrite(['.\events\',Vfile(1:length(Vfile)-4),'_Event_Indices.txt'],...
    ecounter);

%% MAC
%dlmwrite([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'],ecounter);

fitlen=9;
rpindex=find(releasepoint);
stallslopes=zeros(length(rpindex),1);

for I = 1:length(rpindex)

    stallslope=polyfit(nmt(rpindex(I)-fitlen:rpindex(I)-1),nmdata(...
        rpindex(I)-fitlen:rpindex(I)-1,2),1);
    stallslopes(I)=stallslope(1);
```

```
end


stalltest=stallslopes<200;
stallpoint=rpindex(stalltest);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Plots a Force vs. Velocity curve.  Overlaid in blue circles are the
%   "Releasepoints" that have been described previously.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure
plot(abs(fdata(1:btnum-1,2)),yvel(:),'k.',abs(fdata(releasepoint,2)),...
    yvel(releasepoint),'o')
xlabel('Force (pN)')
ylabel('Velocity (nm/s)')
title(['Force vs Velocity - ',Vfile])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Three figure plots showing Velocity, Position and Force vs. Time,
%   respectfully.  Overlaid in blue circles are the "Releasepoints" that
%   have been described previously.  These circles are "Stallpoints" in the
%   Force vs. Time plot.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure
plot(nmt(releasepoint),yvel(releasepoint),'o',nmt(1:btnum-1),yvel(:),'k')
xlabel('Time (s)')
ylabel('Velocity (nm/s)')
title(['Velocity vs. Time - ',Vfile])

figure
plot(nmt,nmdata(:,1),'r.',nmt,nmdata(:,2),'k.',...
    nmt(releasepoint),nmdata(releasepoint,2),'o')
xlabel('Time (s)')
ylabel('Position (nm)')
```

```
legend('X','Y')
title(['Position vs. Time - ',Vfile])
text(nmt(releasepoint),nmdata(releasepoint,2)+5.0,num2str(abs(yvel...
    (releasepoint))));

figure
plot(nmt(:),fdata(:,1),'r',nmt(:),fdata(:,2),'k',...
    nmt(stallpoint),fdata(stallpoint,2),'o')
xlabel('Time (s)')
ylabel('Force (pN)')
hold
text(nmt(releasepoint),fdata(releasepoint,2)+0.2,num2str(stallslopes));
text(nmt(releasepoint),fdata(releasepoint,2)+0.4,num2str(fdata(...
    releasepoint,2)));
title(['Force vs. Time - ',Vfile])

figure
stallforces=fdata(stallpoint,2);
histfit(stallforces);
xlabel('Stall Force (pN)')
title(['Stall Force Distribution - Mean: ',num2str(mean(stallforces)),...
    ' pN'])

function nm=ConvertVtoNM(Vx,Vy,CalFile)

AODtonmx=1148.1; %[nm/MHz]
AODtonmy=1041.1; %[nm/MHz]

% Load Calibration Parameters
cal=load(CalFile); %V to AOD Space Calibration Parameters
calx=cal(:,1);
caly=cal(:,2);

nm(:,1)=AODtonmx*(calx(1)+calx(2)*Vx+calx(3)*Vy+calx(4)*Vx.^2+calx(5)...
    *Vy.^2+calx(6)*Vx.^3+calx(7)*Vy.^3+calx(8)*Vx.^4+calx(9)*Vy.^4+...
    calx(10)*Vx.^5+calx(11)*Vy.^5+calx(12)*Vx.*Vy+calx(13)*Vx.^2.*Vy...
    +calx(14)*Vx.*Vy.^2+calx(15)*Vx.^3.*Vy+calx(16)*Vx.^2.*Vy.^2+...
    calx(17)*Vx.*Vy.^3+calx(18)*Vx.^4.*Vy+calx(19)*Vx.^3.*Vy.^2+...
    calx(20)*Vx.^2.*Vy.^3+calx(21)*Vx.*Vy.^4-26);
nm(:,2)=AODtonmy*(caly(1)+caly(2)*Vx+caly(3)*Vy+caly(4)*Vx.^2+caly(5)...
    *Vy.^2+caly(6)*Vx.^3+caly(7)*Vy.^3+caly(8)*Vx.^4+caly(9)*Vy.^4+...
    caly(10)*Vx.^5+caly(11)*Vy.^5+caly(12)*Vx.*Vy+caly(13)*Vx.^2.*Vy...
    +caly(14)*Vx.*Vy.^2+caly(15)*Vx.^3.*Vy+caly(16)*Vx.^2.*Vy.^2+...
    caly(17)*Vx.*Vy.^3+caly(18)*Vx.^4.*Vy+caly(19)*Vx.^3.*Vy.^2+...
    caly(20)*Vx.^2.*Vy.^3+caly(21)*Vx.*Vy.^4-26);

function newxy=rotateCoords(oldxy,angle)

%angle is positive when rotating CCW and is in degrees
if length(oldxy(:,1))==2
    x=oldxy(1,:);
    y=oldxy(2,:);
else
    x=oldxy(:,1);
    y=oldxy(:,2);
end

newx=x*cos(deg2rad(angle))+y*sin(deg2rad(angle));
newy=-x*sin(deg2rad(angle))+y*cos(deg2rad(angle));

if length(oldxy(:,1))==2
    newxy(1,:)=newx;
    newxy(2,:)=newy;
else
    newxy(:,1)=newx;
    newxy(:,2)=newy;
end

function R=deg2rad(D)

%DEG2RAD Converts angles from degrees to radians
%
```

```
%  rad = DEG2RAD(deg) converts angles from degrees to radians.
%
%  See also RAD2DEG, DEG2DMS, ANGLEDIM, ANGL2STR

%  Copyright 1996-2000 Systems Planning and Analysis, Inc. and The
%  MathWorks, Inc.
%  Written by:  E. Byrns, E. Brown
%   $Revision: 1.8 $    $Date: 2000/01/18 02:00:07 $


if nargin==0
    error('Incorrect number of arguments')
elseif ¬isreal(D)
     warning('Imaginary parts of complex ANGLE argument ignored')
     D = real(D);
end

R = D*pi/180;

function NumberOfCursor=CreateCursor(fhandle, CursorName,bgcolor)
% scCreateCursor creates or replaces a vertical cursor
%
% Examples
% n=CreateCursor()
% n=CreateCursor(fighandle)
% n=CreateCursor(fighandle, CursorNumber)
%
% Cursors are specific to a figure. If the figure handle is not specified
% in fighandle, the current figure (returned by gcf) will be used.
% CursorNumber is any positive number. If not specified the lowest numbered
% free cursor number will be used.
%
% Returns n, the number of the cursor created in the relevant figure.
%
% A record is kept of the cursors in the figure's application data
% area. Cursor n occupies the nth element of a cell array. Each element is
% a structure containing the following fields:
%           Handles:    a list of objects associated with this cursor -
%                       one line for each axes and one or more text objects
%           IsActive:   true if this cursor is currently being moved. False
%                       otherwise. For manual cursors, IsActive is set by a
%                       button down on the cursor and cleared by a button
%                       up.
% Functions that affect the position of a cursor must explicitly update all
% the objects listed in Handles. A cursor is not an object itself.
%
%
% -------------------------------------------------------------------------
% Author: Malcolm Lidierth 01/07
% Copyright   The Author & King's College London 2007
% -------------------------------------------------------------------------
%

% Note: These functions presently work with 2D views only

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Dave mod to get around errors with additional arguments in function..
%can't number cursors directly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NumberOfCursor=7;
% Figure details
if nargin<1
    fhandle=gcf;
end
AxesList=sort(findobj(fhandle, 'type', 'axes'));


% Get cursor info
newhandles=zeros(1, length(AxesList));
Cursors=getappdata(fhandle, 'VerticalCursors');

% Deal with this cursor number
```

```
if isempty(Cursors)
    NumberOfCursor=1;
else
  %  if nargin<2
        FirstEmpty=0;
        if length(Cursors)>0
            for i=1:length(Cursors)
                if isempty(Cursors{i})
                    FirstEmpty=i;
                    break
                end
            end
        end
        if FirstEmpty==0
            NumberOfCursor=length(Cursors)+1;
            Cursors{NumberOfCursor}=[];
        else
            NumberOfCursor=FirstEmpty;
        end
%    end
end


if ¬isempty(Cursors) && ¬isempty(Cursors{NumberOfCursor}) &&...
        isfield(Cursors{NumberOfCursor},'CursorIsActive');
    if Cursors{NumberOfCursor}.CursorIsActive==true
        disp(['CreateCursor: Ignored attempt to delete the currently'...
            'active cursor']);
        return
    else
        delete(Cursors{NumberOfCursor}.Handles);
    end
end
Cursors{NumberOfCursor}.IsActive=false;
Cursors{NumberOfCursor}.Handles=[];
```

```
Cursors{NumberOfCursor}.Type='';


XLim=get(AxesList(end),'XLim');
xhalf=XLim(1)+(0.5*(XLim(2)-XLim(1)));
for i=1:length(AxesList)
    % For each cursor there is a line object in each axes
    subplot(AxesList(i));
        YLim=get(AxesList(i),'YLim');
        newhandles(i)=line([xhalf xhalf], [YLim(1) YLim(2)]);
        Cursors{NumberOfCursor}.Type='2D';
end


% Put a label at the top and make it behave as part of the cursor
axes(AxesList(1));
YLim=get(AxesList(1),'YLim');
th=text(xhalf, YLim(2), CursorName,...%['C' num2str(NumberOfCursor)],...
    'Tag','Cursor',...
    'UserData', NumberOfCursor,...
    'FontSize',6,...
    'HorizontalAlignment','center',...
    'VerticalAlignment','bottom',...
    'Color','w',...
    'BackgroundColor',bgcolor,...
    'Clipping','off',...
    'ButtonDownFcn',{@CursorButtonDownFcn});

% Set line properties en bloc
% Note UserData has the cursor number
if ispc==1
    % Windows
    EraseMode='xor';
else
    % Mac etc: 'xor' may cause problems
    EraseMode='normal';
```

```
end
set(newhandles, 'Tag', 'Cursor',...
    'Color', bgcolor,...
    'UserData', NumberOfCursor,...
    'Linewidth',1,...
    'Erasemode', EraseMode,...
    'ButtonDownFcn',{@CursorButtonDownFcn});

Cursors{NumberOfCursor}.IsActive=false;
Cursors{NumberOfCursor}.Handles=[newhandles th];

setappdata(fhandle, 'VerticalCursors', Cursors);
set(gcf, 'WindowButtonMotionFcn',{@CursorWindowButtonMotionFcn});
return

function CursorWindowButtonMotionFcn(fhandle, EventData)
% CursorWindowButtonMotionFcn - figure motion callback to support cursors
%
% To activate, use
% set(figurehandle, 'WindowButtonMotionFcn',{@CursorWindowButtonMotionFcn})
% This property is set automatically vy CreateCursor
%
% CursorWindowButtonMotionFcn does not need to be active - if is interferes
% with graphics rendering clear the WindowButtonMotionFcn property
%
% CursorWindowButtonMotionFcn displays the cursor movement pointer when the
% cursor location is over a line object that belongs to a cursor created by
% CreateCursor. Over a cursor means within 3 pixels of it.
%
% Note: When the pointer is not over a cursor CursorWindowButtonMotionFcn
% resets the cursor to the default arrow. It will not restore the pointer
% to e.g. an hour glass.
%
% -------------------------------------------------------------------------
% Author: Malcolm Lidierth 01/07
% Copyright   The Author & King's College London 2007
% -----------------------------------------------------------------------------

CData=[NaN  NaN NaN NaN NaN 2   2   2   2   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN 2   1   2   1   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN 2   1   2   1   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN 2   2   1   2   1   2   2   NaN NaN NaN NaN NaN;...
    NaN NaN NaN 2   1   2   1   2   1   2   1   2   NaN NaN NaN NaN;...
    NaN NaN 2   1   1   2   1   2   1   2   1   1   2   NaN NaN NaN;...
    NaN 2   1   1   1   1   1   2   1   1   1   1   1   2   NaN NaN;...
    2   1   1   1   1   1   1   2   1   1   1   1   1   1   2   NaN;...
    NaN 2   1   1   1   1   1   2   1   1   1   1   1   2   NaN NaN;...
    NaN NaN 2   1   1   2   1   2   1   2   1   1   2   NaN NaN NaN;...
    NaN NaN NaN 2   1   2   1   2   1   2   1   2   NaN NaN NaN NaN;...
    NaN NaN NaN NaN 2   2   1   2   1   2   2   NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN 2   1   2   1   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN 2   1   2   1   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN 2   2   2   2   2   NaN NaN NaN NaN NaN NaN;...
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN];


% Get objects associated with the cursors
Cursors=getappdata(fhandle, 'VerticalCursors');
h=[];
for i=1:length(Cursors)
    if ¬isempty(Cursors{i})
        h=horzcat(h,Cursors{i}.Handles);
    end
end

% Return if there are none
if isempty(h)
    return
end

h=findobj(h, 'Type', 'line','Tag', 'Cursor', 'Parent', gca);
% We may be in the process of deleting the objects or their parents - use
% only valid handles
h=h(ishandle(h));

% Return if there are none
if isempty(h)
    return
end

%Check pointer is over an axes....
AxesList=sort(findobj(fhandle, 'type', 'axes'));
saveunits=get(AxesList(1), 'Units');
set(AxesList, 'Units', 'pixels');
coords=get(AxesList,'Position');
set(AxesList, 'Units', saveunits);

saveunits=get(fhandle, 'Units');
set(fhandle, 'Units', 'pixels');
pixelpos=get(fhandle, 'CurrentPoint');
set(gcf, 'Units', saveunits);

%... and get its number
axnumber=whichaxes(pixelpos, coords);

if axnumber==0
    % Not over an axes - so restore the default cursor and return
    set(fhandle, 'Pointer', 'arrow');
    return
else
    % Work on the axes that the pointer is over
    ax=AxesList(axnumber);

    % Get the axes position in pixels
    saveunits=get(ax, 'Units');
    set(ax, 'Units', 'pixels');
    axpos=get(ax, 'Position');
    set(ax, 'Units', saveunits);
    % Current point (axis units)
    pos=get(ax,'CurrentPoint');
    % Get the cursor positions (axis units)
    CursorPos=get(h,'XData');
    if iscell(CursorPos)
        CursorPos=cell2mat(CursorPos);
    end

    % Remove offset and scale 0 to 1
    XLim=get(ax, 'XLim');
    CursorPos=(CursorPos-XLim(1))/(XLim(2)-XLim(1));
    pos=(pos-XLim(1))/(XLim(2)-XLim(1));
    % Convert to pixels
    CursorPos=CursorPos*axpos(3);
    pos=pos*axpos(3);

    % MATLAB selects an item when the pointer is within 5 pixels.
    % Change the pointer when we are within 3 pixels
    idx=find(abs(CursorPos(:,1)-pos(1))<3,1);
    if ¬isempty(idx)
        % Yes - activate the cursor pointer
        %setptr(fhandle,'lrdrag');
        set(fhandle,'PointerShapeCData',CData);
        set(fhandle,'Pointer','custom');
        set(fhandle, 'PointerShapeHotSpot',[8 8]);
    else
        % No -
        set(fhandle, 'Pointer', 'arrow');
    end
end

%---------------------------------------------------------------------------
function ax = whichaxes(pts,rect)
%---------------------------------------------------------------------------
% whichaxes determines which axes the cursor is over
%  Example:
%  ax=whichaxes(pts, rect)
%  pts are the cursor coordinates 1x2
%  rect are the coordinates for the axes positions Nx4
%
% This is a modification of the pinrect function that is used in several
% MATLAB files

ax=0;
rect=cell2mat(rect);
rect(:,3)=rect(:,1)+rect(:,3);
rect(:,4)=rect(:,2)+rect(:,4);
for k=1:size(rect,1)
    bool = (pts(1)>rect(k,1) && pts(1)<rect(k,3)) &&...
        (pts(2)>rect(k,2) && pts(2)<rect(k,4));
    if bool
        ax=k;
        return
    end
end

%-------------------------------------------------------------------------
function CursorButtonDownFcn(hObject, EventData)
%-------------------------------------------------------------------------

% Make sure we have a left button click
type=get(ancestor(hObject,'figure'), 'SelectionType');
flag=strcmp(type, 'normal');
if flag==0
    % Not so, return and let matlab call the required callback,
    % e.g. the uicontextmenu, which will be in the queue
    return
end;

% Flag the active cursor
Cursors=getappdata(gcf, 'VerticalCursors');
```

```
for i=1:length(Cursors)
    if isempty(Cursors{i})
        continue
    end
    if any(Cursors{i}.Handles==hObject)
            % Set flag
            Cursors{i}.IsActive=true;
            break
    end
end
setappdata(gcf, 'VerticalCursors', Cursors);

% Set up
StoreWindowButtonDownFcn=get(gcf,'WindowButtonDownFcn');
StoreWindowButtonUpFcn=get(gcf,'WindowButtonUpFcn');
StoreWindowButtonMotionFcn=get(gcf,'WindowButtonMotionFcn');
% Store these values in the CursorButtonUpFcn persistent variables so they
% can be used/restored later
CursorButtonUpFcn({hObject,...
    StoreWindowButtonDownFcn,...
    StoreWindowButtonUpFcn,...
    StoreWindowButtonMotionFcn});
% Motion callback needs only the current cursor number
CursorButtonMotionFcn({hObject});

% Set up callbacks
set(gcf, 'WindowButtonUpFcn',{@CursorButtonUpFcn});
set(gcf, 'WindowButtonMotionFcn',{@CursorButtonMotionFcn});
return

%--------------------------------------------------------------------------
function CursorButtonUpFcn(hObject, EventData)
%--------------------------------------------------------------------------
% These persistent values are set by a call from CursorButtonDownFcn
persistent ActiveHandle;
persistent StoreWindowButtonDownFcn;
persistent StoreWindowButtonUpFcn;
persistent StoreWindowButtonMotionFcn;

% Called from CursorButtonDownFcn - hObject is a cell with values to seed
% the persistent variables
if iscell(hObject)
    ActiveHandle=hObject{1};
    StoreWindowButtonDownFcn=hObject{2};
    StoreWindowButtonUpFcn=hObject{3};
    StoreWindowButtonMotionFcn=hObject{4};
    return
end

% Called by button up in a figure window - use the stored CurrentCursor
% value
UpdateCursorPosition(ActiveHandle)

% Restore the figure's original callbacks - make sure we do this in the
% same figure that we had when the mouse button-down was detected
h=ancestor(ActiveHandle,'figure');
set(h, 'WindowButtonDownFcn', StoreWindowButtonDownFcn);
set(h, 'WindowButtonUpFcn', StoreWindowButtonUpFcn);
set(h, 'WindowButtonMotionFcn',StoreWindowButtonMotionFcn);


% Remove the active cursor flag
Cursors=getappdata(gcf, 'VerticalCursors');
for i=1:length(Cursors)
    if isempty(Cursors{i})
        continue
    end
    if any(Cursors{i}.Handles==ActiveHandle)
        Cursors{i}.IsActive=false;
        break
    end
end
```

```
setappdata(gcf, 'VerticalCursors', Cursors);

return

function CursorButtonMotionFcn(hObject, EventData)
%--------------------------------------------------------------------------
% This replaces the CursorWindowButtonMotionFcn while a cursor is being
% moved
persistent ActiveHandle;

% Called from CursorButtonDownFcn
if iscell(hObject)
    ActiveHandle=hObject{1};
    return
end
%Called by button up
UpdateCursorPosition(ActiveHandle)
return

%--------------------------------------------------------------------------
function UpdateCursorPosition(ActiveHandle)
%--------------------------------------------------------------------------
% Get all lines in all axes in the current figure that are
% associated with the current cursor....
CursorHandles=findobj(gcf, 'Type', 'line',...
    'Tag', 'Cursor',...
    'UserData', get(ActiveHandle,'UserData'));
% ... and update them:
% Get the pointer position in the current axes
cpos=get(gca,'CurrentPoint');
if cpos(1,1)==cpos(2,1) && cpos(1,2)==cpos(2,2)
    % 2D Cursor
    % Limit to the x-axis limits
    XLim=get(gca,'XLim');
    if cpos(1)<XLim(1)
        cpos(1)=XLim(1);
    end
    if cpos(1)>XLim(2)
        cpos(1)=XLim(2);
    end
    % Set
    set(CursorHandles,'XData',[cpos(1) cpos(1)]);
else
    % TODO: Include support for 3D cursors
    return
end

% Now update the cursor Label
LabelHandle=findobj(gcf, 'Type', 'text', 'Tag', 'Cursor', 'UserData',...
    get(ActiveHandle,'UserData'));
tpos=get(LabelHandle,'position');
tpos(1)=cpos(1);
set(LabelHandle,'position',tpos);
return

function SetCursorLocation(varargin)
% SetCursorPos returns the position of a cursor in x-axis units
%
% Example:
% x=SetCursorLocation(CursorNumber, Position)
% x=SetCursorLocation(fhandle, CursorNumber, Position)
%
% fhandle defaults to the current figure if not supplied or empty
%
% Sets the x-axis position for the cursor in the figure
%
%--------------------------------------------------------------------------
% Author: Malcolm Lidierth 01/07
% Copyright   The Author & King's College London 2007
%--------------------------------------------------------------------------

switch nargin
```

```
    case 2
        fhandle=gcf;
        CursorNumber=varargin{1};
        Pos=varargin{2};
    case 3
        fhandle=varargin{1};
        CursorNumber=varargin{2};
        Pos=varargin{3};
    otherwise
        return
end

Cursors=getappdata(fhandle,'VerticalCursors');
if isempty(Cursors)
    % No cursors in figure
    x=[];
    return
end

try
    %Use try/catch as the selected cursor may not exist
    idx=strcmpi(get(Cursors{CursorNumber}.Handles,'type'),'line');
    set(Cursors{CursorNumber}.Handles(idx), 'XData', [Pos Pos]);
    idx=strcmpi(get(Cursors{CursorNumber}.Handles,'type'),'text');
    for i=1:length(idx)
        if idx(i)==1
            p=get(Cursors{CursorNumber}.Handles(i),'Position');
            p(1)=Pos;
            set(Cursors{CursorNumber}.Handles(i), 'Position', p);
        end
    end
catch
    return
end
```

```
return

function x=GetCursorLocation(varargin)
% GetCursorLocation returns the position of a cursor in x-axis units
%
% Example:
% x=GetCursorLocation(CursorNumber)
% x=GetCursorLocation(fhandle, CursorNumber)
%
% fhandle defaults to the current figure if not supplied or empty
%
% Returns x,  the x-axis position for the cursor in the figure
%
%-------------------------------------------------------------------------
% Author: Malcolm Lidierth 01/07
% Copyright   The Author & King's College London 2007
%-------------------------------------------------------------------------

switch nargin
    case 1
        fhandle=gcf;
        CursorNumber=varargin{1};
    case 2
        fhandle=varargin{1};
        CursorNumber=varargin{2};
end

Cursors=getappdata(fhandle,'VerticalCursors');
if isempty(Cursors)
    % No cursors in figure
    x=[];
    return
end

try
    %Use try/catch as the selected cursor may not exist
    pos=get(Cursors{CursorNumber}.Handles(1),'XData');
    x=pos(1);
catch
    x=[];
end

return

function DeleteCursor(varargin)
% DeleteCursor deletes a cursor created by CreateCursor
%
% Examples:
% scDeleteCursor(CursorNumber)
% scDeleteCursor(fhandle, CursorNumber)
%
% -------------------------------------------------------------------------
% Author: Malcolm Lidierth 01/07
% Copyright   The Author & King's College London 2007
% -------------------------------------------------------------------------

if nargin==1
    fhandle=gcf;
    CursorNumber=varargin{1};
else
    fhandle=varargin{1};
    CursorNumber=varargin{2};
end

% Retrieve cursor info
Cursors=getappdata(fhandle, 'VerticalCursors');
% Delete associated lines and text
delete(Cursors{CursorNumber}.Handles);
% Empty the cell array element - can be re-used
Cursors{CursorNumber}={};
% Trim if last cell is empty
if isempty(Cursors{end})
```

```
    Cursors(end)=[];
end
% Update in application data area
setappdata(fhandle, 'VerticalCursors', Cursors);

function degs=FindMTRotationBoxcar(bcarx,bcary)
% This function assumes you already have filtered your data with a boxcar
% average

% this bizzare thing seems to be a lot happier if we plot x vs y rather
% than y vs x..  nlinfit is "special"

%pco=nlinfit(bcary,bcarx,@lineq,[.3,0]);
[slp icpt]=twoIndVars(bcarx,bcary);
theta=90-rad2deg(atan(slp));

disp(['Your data was rotated ',num2str(theta),' degrees to account for'...
    'MT orientation'])

plot(bcarx,bcary,'.',bcarx,slp*bcarx+icpt,'r')
axis equal
xlabel('Position X (nm)')
ylabel('Position Y (nm)')
title('Y-X plot to determine MT orientation')

degs=theta;

function [slope intercept]=twoIndVars(xs,ys)

mux=mean(xs);
muy=mean(ys);

Sxx=sum((xs-mux).^2);
Syy=sum((ys-muy).^2);

Sxy=sum((xs-mux).*(ys-muy));
```

```
slope=2*Sxy/(Sxx-Syy+((Sxx-Syy)^2+4*Sxy^2)^(1/2));
intercept=(2*Sxy/(Syy-Sxx-((Sxx-Syy)^2+4*Sxy^2)^1/2))*mux+muy;

function [FV keep]=EvalEvent(tdata,nmdata,stiff,eventindicies,odeg,...
    direction,savename)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Function that adjusts baseline & rotates a single event
%   Fits to find instantaneous force velocity data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
event.Rotation=odeg;
event.XStiffness=stiff(1);
event.YStiffness=stiff(2);
event.Direction=direction;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Baseline Adjust
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt=tdata(2)-tdata(1);
bcarwin=50;
btdata=mboxcar(tdata,bcarwin);
figure
tnmdata=rotateCoords(nmdata(:,:),-1*odeg);  %Temporary nm data rotated
ph=plot(tdata,tnmdata(:,1),'.r',tdata,tnmdata(:,2),...
    '.b',btdata,mboxcar(tnmdata(:,1),bcarwin),...
    '-r',btdata,mboxcar(tnmdata(:,2),bcarwin),'-b');
set(ph,'MarkerSize',0.5);
xlabel('Time (s)')
ylabel('Position (nm)')
legend('X','Y')
axis tight

fh(1)=gcf;
xlims=get(gca,'XLim'); %get the new x limits

%Create cursors
```

```
cnums(1)=CreateCursor(fh(1),'BG 1',[26/255 133/255 5/255]);
cnums(2)=CreateCursor(fh(1),'BG 2',[26/255 133/255 5/255]);
cnums(3)=CreateCursor(fh(1),'FV 1',[170/255 26/255 5/255]);
cnums(4)=CreateCursor(fh(1),'FV 2',[170/255 26/255 5/255]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Set cursor positions
%   Cursors 3 & 4 get set where the event was detected
%   The background cursors are set to indicate the buffer between events
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sampling rate
rawrate=2000;

%buffer between events
buffertime=0.5; %1/2 second
bufferpts=rawrate*buffertime;

%buffer between beginning of event and background
splittime=0.05;
splitpts=rawrate*splittime;

if eventindicies(1)>length(tdata)
    eventindicies(1)=bufferpts+splitpts+1;
end

SetCursorLocation(cnums(1),tdata(eventindicies(1)-bufferpts-splitpts));
SetCursorLocation(cnums(2),tdata(eventindicies(1)-splitpts));
SetCursorLocation(cnums(3),tdata(eventindicies(1)));
SetCursorLocation(cnums(4),tdata(eventindicies(2)-splitpts));


disp(['Place cursors to indicate desired regions (BG = Background, FV'...
    ' = Force Velocity)']);
input('Hit enter once you are satisfied with your selection');

%Find out the desired positions
```

```
for iCnt=1:4
    cps(iCnt)=GetCursorLocation(cnums(iCnt));
end

event.BGStartIndex=floor((cps(1)-tdata(1))/dt);
event.BGEndIndex=floor((cps(2)-tdata(1))/dt);

xadj=mean(nmdata(floor((cps(1)-tdata(1))/dt):floor((cps(2)-tdata(1))...
    /dt),1));
yadj=mean(nmdata(floor((cps(1)-tdata(1))/dt):floor((cps(2)-tdata(1))...
    /dt),2));
nmdata(:,1)=(nmdata(:,1)-xadj);
nmdata(:,2)=(nmdata(:,2)-yadj);

disp(['Adjusting X baseline by ', num2str(xadj)]);
disp(['Adjusting Y baseline by ', num2str(yadj)]);

%%%%% Get the background variance %%%%%%
bgvar(1)=var(nmdata(floor((cps(1)-tdata(1))/dt):floor((cps(2)-tdata(1))...
    /dt),1));
bgvar(2)=var(nmdata(floor((cps(1)-tdata(1))/dt):floor((cps(2)-tdata(1))...
    /dt),2));
event.BGVarianceX=bgvar(1);
event.BGVarianceY=bgvar(2);

FStartIndex=floor((cps(3)-tdata(1))/dt);
FEndIndex=floor((cps(4)-tdata(1))/dt);
event.FVStartIndex=FStartIndex;
event.FVEndIndex=FEndIndex;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Convert to Force
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdata(:,1)=nmdata(:,1)*stiff(1);
fdata(:,2)=nmdata(:,2)*stiff(2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Rotate all of the data (have to do this over, because the baseline may
%   have been adjusted)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdata(:,1:2)=rotateCoords(fdata(:,1:2),-1*odeg);
nmdata(:,1:2)=rotateCoords(nmdata(:,1:2),-1*odeg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Check and plot the variance for the trace - also want to note stall
%   variance and the background variance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[vtdat vnmdat]=VarianceBox(tdata(FStartIndex:FEndIndex),...
    nmdata(FStartIndex:FEndIndex,:));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Coppin et al force velocity curve determination
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CopinTime=0.05;
event.CopinWindow=CopinTime;
regpts=ceil(CopinTime/dt); %300 ms window for averging / FV calc
iInd=1;
for iCnt=FStartIndex:FEndIndex-regpts
    cp=polyfit(tdata(iCnt:iCnt+regpts),nmdata(iCnt:iCnt+regpts,2),1);
    FV(iInd,:)=[mean(fdata(iCnt:iCnt+regpts,2)) cp(1)];
    iInd=iInd+1;
end

% Figure out the "stall" velocity and "stall" force
[stallforce stallind]=max(FV(:,1));
stallvel=FV(stallind,2);
event.StallForce=stallforce;
event.StallVelocity=stallvel;
% Summary Figures
figure
fh(2)=gcf;
```

```
set(fh(2),'Position',[100 100 650 800])

subplot(3,2,1)  %Position vs time (w avgd)
ph=plot(tdata,nmdata(:,1),'.r',tdata,nmdata(:,2),...
    '.b',btdata,mboxcar(nmdata(:,1),bcarwin),...
    '-r',btdata,mboxcar(nmdata(:,2),bcarwin),'-b');
set(ph,'MarkerSize',0.5);
xlabel('Time (s)')
ylabel('Position (nm)')
title('Event Position vs Time')
axis tight

subplot(3,2,2)  %Force vs time (w avgd)
ph=plot(tdata,fdata(:,1),'.r',tdata,fdata(:,2),...
    '.b',btdata,mboxcar(fdata(:,1),bcarwin),...
    '-r',btdata,mboxcar(fdata(:,2),bcarwin),'-b');
set(ph,'MarkerSize',0.5);
xlabel('Time (s)')
ylabel('Force (pN)')
title('Event Force vs Time')
axis tight

subplot(3,2,3)  %Force vs time, event only
ph=plot(tdata(FStartIndex:FEndIndex),fdata(FStartIndex:FEndIndex,2),'.b',...
    mboxcar(tdata(FStartIndex:FEndIndex),regpts),...
    mboxcar(fdata(FStartIndex:FEndIndex,2),regpts),'-k');
set(ph,'MarkerSize',0.5);
xlabel('Time (s)')
ylabel('Force (pN)')
%legend('RawData','Polyfit')
title('FV Segment Only')
axis tight

subplot(3,2,4)  % FV curve with stall force and velocity
plot(FV(:,1),FV(:,2),'-b',stallforce,stallvel,'og')
xlabel('Force (pN)')
```

```
ylabel('Velocity (nm/s)')
%text(FV(1,1)+0.05,stallvel+10,['Force: ',num2str(stallforce),' pN']);
%text(FV(1,1)+0.05,stallvel+32,['Velocity: ',num2str(stallvel),' nm/s']);
title(['F-V Curve, SF:',num2str(stallforce,2),' pN, SV:',num2str(...
    stallvel,2),'nm/s'])
axis tight

subplot(3,2,5)  % FV curve with stall force and velocity
plot(vtdat,vnmdat(:,1),'-r',vtdat,vnmdat(:,2),'-b')
event.FinalVarianceX=vnmdat(length(vnmdat(:,1)),1);
event.FinalVarianceY=vnmdat(length(vnmdat(:,2)),2);

xlabel('Time (s)')
ylabel('Variance (nm^2)')
text(vtdat(1,1),bgvar(2),['+ Bg Var: ',num2str(bgvar(2))]);

title('Variance vs Time')
axis tight

dats=FV;
keep=input('Keep event? 1=yes 2=no:  ');
if keep==1
    dlmwrite(['.\events\stalls\',savename,'_FV.txt'],FV);
    dlmwrite(['.\events\stalls\',savename,'_BufferedEvent.txt'],[tdata...
        nmdata fdata]);
    dlmwrite(['.\events\stalls\',savename,'_ClippedEvent.txt'],...
        [tdata(FStartIndex:FEndIndex)...
        nmdata(FStartIndex:FEndIndex,:) fdata(FStartIndex:FEndIndex,:)]);
    save(['.\events\stalls\',savename,'_EventData.mat'],'event');
    disp(['Wrote ',savename,'_... to the \events\stalls directory.'])
else
    dlmwrite(['.\events\discards\',savename,'_FV.txt'],FV);
    dlmwrite(['.\events\discards\',savename,'_BufferedEvent.txt'],[tdata...
        nmdata fdata]);
    dlmwrite(['.\events\discards\',savename,'_ClippedEvent.txt'],...
        [tdata(FStartIndex:FEndIndex)...
        nmdata(FStartIndex:FEndIndex,:) fdata(FStartIndex:FEndIndex,:)]);
    save(['.\events\discards\',savename,'_EventData.mat'],'event');
    disp(['Wrote ',savename,'_... to the \events\discards directory.'])
end

close(fh)


function [tdat vnmdat]=VarianceBox(tdata,nmdata)

j=1;

for i=1:50:length(nmdata(:,1))
    if i+49 > length(nmdata(:,1))
        ef=length(nmdata(:,1));
    else
        ef=i+49;
        vnmdat(j,:)=var(nmdata(i:ef,:));
        tdat(j)=mean(tdata(i:ef));
        j=j+1;

    end
end

function fdat=mboxcar(dat,m)

n=length(dat);
z=[0 cumsum(dat)'];
fdat=(z(m+1:n+1)-z(1:n-m+1))/m;

% % for i=1:(length(dat)-window+1)
% %     sum=0;
% %     for j=0:(window-1)
% %         sum=sum+dat(i+j);
% %     end
% %     fdat(i)=sum/window;
% % end
```

B.1.2 Force-Velocity Calculation

```
clear; close all; clc

sampleRate = 2000;          % Hz
coppinTime = 15e-3;         % [s]: time window over which to evaluate
%                                                               velocities
Δ = 1.0;
FBins = [1.0:Δ:8]';     % Final Force Bins

%noRuns = input('Largest Run # in Directory? ')
FStalls = [];
FStallsTemp = [];
VStalls = [];
VStallsTemp = [];
ystiff = [];
i = 1;

Vfile = ['04222010-CS-1E6-4b.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        [VStallsTemp FstallsTemp] = FVRun_v4( Vfile,FBins,sampleRate,...
            coppinTime,Δ );
        FStalls = [FStalls FStallsTemp];
        VStalls = [VStalls VStallsTemp];
        stiff = dlmread(['C:\Documents and Settings\Bill\Desktop\'...
            'Research\Kinesin\data\CS\04222010\04222010-CS-1E6-4-3.txt']);
        ystiff = [ystiff stiff(3)*ones(1,size(VStallsTemp,2))];
        i = i+1;
    else
        cd ..
        i = i+1;
    end

i = 1;
while i < 6
Vfile = ['07212010-CS-1E5-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        [VStallsTemp FstallsTemp] = FVRun_v4( Vfile,FBins,sampleRate,...coppinTime,Δ );
        FStalls = [FStalls FStallsTemp];
        VStalls = [VStalls VStallsTemp];
        stiff = dlmread(['C:\Documents and Settings\Bill\Desktop\'...
            'Research\Kinesin\data\CS\Loaded-Unloaded test\07212010\'...
            'stall_force data\',Vfile(1:length(Vfile)-4),'-3.txt']);
        ystiff = [ystiff stiff(3)*ones(1,size(VStallsTemp,2))];

        i = i+1;
    else
        cd ..
        i = i+1;
    end
end

i = 1;
while i < 6
Vfile = ['07232010-CS-1E5-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        [VStallsTemp FstallsTemp] = FVRun_v4( Vfile,FBins,sampleRate,...
            coppinTime,Δ );
        FStalls = [FStalls FStallsTemp];
        VStalls = [VStalls VStallsTemp];
        stiff = dlmread(['C:\Documents and Settings\Bill\Desktop\'...
            'Research\Kinesin\data\CS\Loaded-Unloaded test\07232010\'
            ...'stall_force data\',Vfile(1:length(Vfile)-4),'-3.txt']);
        ystiff = [ystiff stiff(3)*ones(1,size(VStallsTemp,2))];

        i = i+1;
    else
        cd ..
        i = i+1;
    end
end

i = 1;
while i < 6
Vfile = ['07232010-CS-5E4-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        [VStallsTemp FstallsTemp] = FVRun_v4( Vfile,FBins,sampleRate,...
            coppinTime,∆ );
        FStalls = [FStalls FStallsTemp];
        VStalls = [VStalls VStallsTemp];
        stiff = dlmread(['C:\Documents and Settings\Bill\Desktop\'...
            'Research\Kinesin\data\CS\Loaded-Unloaded test\07232010\'...
            'stall_force data\',Vfile(1:length(Vfile)-4),'-3.txt']);
        ystiff = [ystiff stiff(3)*ones(1,size(VStallsTemp,2))];
        i = i+1;
    else
        cd ..
        i = i+1;
    end
end

i = 1;
while i < 6
Vfile = ['07272010-CS-2E4-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        [VStallsTemp FstallsTemp] = FVRun_v4( Vfile,FBins,sampleRate,...
            coppinTime,∆ );
        FStalls = [FStalls FStallsTemp];
        VStalls = [VStalls VStallsTemp];
        stiff = dlmread(['C:\Documents and Settings\Bill\Desktop\'...
            'Research\Kinesin\data\CS\Loaded-Unloaded test\07272010\'...
            'stall_force data\',Vfile(1:length(Vfile)-4),'-3.txt']);
        ystiff = [ystiff stiff(3)*ones(1,size(VStallsTemp,2))];
        i = i+1;
    else
        cd ..
        i = i+1;
    end
end

% b=1;
% for z = 1:length(FBins)-1
%     for a=1:size(VStalls,2)
%         if VStalls(z,a)≥0.7*VStalls(z+1,a)
%             VStalls_fix(:,b)=VStalls(:,a);
%             b = b+1;
%         end
%     end
%     VStalls = VStalls_fix;
%     b=1;
% end

numzeros=zeros(length(FBins),1);
VStalls_fix = zeros(size(VStalls,1),size(VStalls,2));

stiffs = zeros(size(VStalls,2),length(FBins));
for z = 1:length(FBins)-1
    zeroloc=find(VStalls(z,:)==0 & VStalls(z+1,:)>0);
    ystifftemp = ystiff;
    ystifftemp(zeroloc) = [];
    stiffs(z,1:length(ystifftemp)) = ystifftemp;
    VStalls_Temprow = VStalls(z,:);
    VStalls_Temprow(zeroloc)=[];
    numzeros(z)=size(VStalls,2)-length(VStalls_Temprow);
    VStalls_fix(z,1:length(VStalls_Temprow))=VStalls_Temprow;
end
VStalls_fix(length(FBins),:)=VStalls(length(FBins),:);
VStalls = VStalls_fix;

avgV = zeros(size(VStalls,1),1);
stdevV = zeros(size(VStalls,1),1);
num = size(VStalls,2);
for q = 1:size(VStalls,1)
    avgV(q) = mean(VStalls(q,1:length(VStalls)-numzeros(q)));
    stdevV(q) = std(VStalls(q,1:length(VStalls)-numzeros(q)));
end
% avgV = zeros(size(VStalls,1),1);
% stdevV = zeros(size(VStalls,1),1);
% num = size(VStalls,2);
% for q = 1:size(VStalls,1)
%     avgV(q) = mean(VStalls(q,:));
%     stdevV(q) = std(VStalls(q,:));
% end

VBins = avgV;
VBinsSTD = stdevV;
VBinsSEM = stdevV/sqrt(num);


p = zeros(length(FBins),2);
for i = 1:length(FBins)
    p(i,:) = polyfit(stiffs(i,1:size(stiffs(i,:),2)-numzeros(i)),...
        VStalls(i,1:size(VStalls,2)-numzeros(i)),1);
end
% VBins = zeros(length(FBins),1);
% for i = 1:length(VBins)
%     iBins = 0;
%     for j = 1:iRun
%         if (isnan(VBinsRun(i,j)) == 0)
%             iBins = iBins + 1;
%             VBins(i,1) = VBins(i,1) + VBinsRun(i,j);
%         end
%     end
% end
% VBins = VBins ./ iBins;
close all;
figure; %plot(FBins,VBins,'o');
errorbar(FBins,VBins,VBinsSEM,'o');
dlmwrite('FV_data.txt',[FBins VBins VBinsSEM])
%figure; hist(Fstalls)


function [VBinsRun FstallsRun] = FVRun_v4( Vfile,FBins,sampleRate,...
    coppinTime,Δ )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cd events;
eventCounter = size(dlmread([Vfile(1:length(Vfile)-4),...
    '_Event_Indices.txt']),1);
cd ..
iEvent = 0;
clear FstallsRun;

for i = 1:eventCounter

    cd events; cd stalls

    rawDataFile = [Vfile(1:length(Vfile)-4),'_Event_',num2str(i),...
        '_ClippedEvent.txt'];

    if exist( rawDataFile )
```

```
        iEvent = iEvent + 1;
        VBinsEvent(1:length(FBins),iEvent) = zeros;
        cd ..; cd ..
        [VBinsEvent(:,iEvent) FstallsRun(iEvent,1)] = FVEvent_v4...
            ( rawDataFile,FBins,sampleRate,coppinTime,∆ );
     else
        cd ..; cd ..
    end


end

% avgV = zeros(size(VBinsEvent,1),1);
% stdevV = zeros(size(VBinsEvent,1),1);
% for q = 1:size(VBinsEvent,1)
%     avgV(q) = mean(VBinsEvent(q,:));
%     stdevV(q) = std(VBinsEvent(q,:));
% end
%
% VBinsRun = avgV;
% VBinsSTD = stdevV;
VBinsRun = VBinsEvent;

%end
%VBinsRun = zeros(length(FBins),1);
%VBinsRun = VBinsRun ./ iEvent;
%VBinsRun = mean(VBinsRun);
%figure; plot( FBins,VBinsRun,'o');



function [VBinsEvent FstallsEvent] = FVEvent_v4( rawDataFile,FBins,...
    sampleRate,coppinTime,∆ )

cd events
cd stalls
```

```
rawData = dlmread( rawDataFile );
tdata = [0:1/sampleRate:(1/sampleRate*(length(rawData)-1))]';
dt = tdata(2) - tdata(1);
nmdata = rawData(:,2:3);
fdata = rawData(:,4:5);

% Local Fit
regpts = coppinTime * sampleRate;
iInd=1;
if regpts ≥ length(rawData)
    regpts = length(rawData)-1;
end
for iCnt=1:regpts:length(rawData)-regpts
    pl=polyfit(tdata(iCnt:iCnt+regpts),nmdata(iCnt:iCnt+regpts,2),1);
    Fl(iInd,1) = mean(fdata(iCnt:iCnt+regpts,2));
    Vl(iInd,1) = pl(1);
    iInd=iInd+1;
end
FstallsEvent = max(Fl);

% Bin Forces
VBinsEvent = zeros(length(FBins),1);
VCnt = zeros(length(FBins),1);
for iBins = 1:length(FBins)
    if FBins(iBins)+Δ/2 < max(Fl)
        for j = 1:length(Fl)
            if ( Fl(j,1) ≥ 0.25 & Fl(j,1) < FBins(iBins)+Δ/2) &...
                    (Fl(j,1) ≥ FBins(iBins)-Δ/2)
                if Vl(j,1) ≥ 0;
                    VBinsEvent(iBins,1) = VBinsEvent(iBins,1) + Vl(j,1);
                    VCnt(iBins,1) = VCnt(iBins,1) + 1;
                end
            end
        end
    else
        VBinsEvent(iBins,1) = 0;
        VCnt(iBins,1) = VCnt(iBins,1)+1;
    end
end

for i = 1:length(VBinsEvent)
    if VCnt(i) > 0
        VBinsEvent(i) = VBinsEvent(i)/VCnt(i);
    else
        VBinsEvent(i) = 0;
    end
end
cd ..
cd ..
```

B.1.3 Velocity at Release

```
clear; close all; clc

sampleRate = 2000;          % Hz
coppinTime = 15e-3;         % [s]: time window over which to evaluate
%                                                          velocities

%noRuns = input('Largest Run # in Directory? ')
iRun = 0;
%for i = 1:noRuns
F = [];
FTemp = [];
V = [];
VTemp = [];
i = 1;

while i < 6
    Vfile = ['02042010-CS-1E6-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
```

```
        cd ..;
        iRun = iRun + 1;
        [VTemp FTemp] = vatstall_run( Vfile,sampleRate );
        F = [F FTemp];
        V = [V VTemp];
        i = i+1;
    else
        cd ..
        i = i+1;
    end
end
i = 1;

while i < 6
    Vfile = ['02052010-CS-1E5-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        iRun = iRun + 1;
        [VTemp FTemp] = vatstall_run( Vfile,sampleRate );
        F = [F FTemp];
        V = [V VTemp];
        i = i+1;
    else
        cd ..
        i = i+1;
    end
end
i = 1;

while i < 7
    Vfile = ['04222010-CS-1E6-',num2str(i),'.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        iRun = iRun + 1;
        [VTemp FTemp] = vatstall_run( Vfile,sampleRate );
        F = [F FTemp];
        V = [V VTemp];
        i = i+1;
    else
        cd ..
        i = i+1;
    end
end
i = 1;

    Vfile = ['04222010-CS-1E6-4b.txt'];
    cd events;
    if exist([Vfile(1:length(Vfile)-4),'_Event_Indices.txt'])
        cd ..;
        iRun = iRun + 1;
        [VTemp FTemp] = vatstall_run( Vfile,sampleRate );
        F = [F FTemp];
        V = [V VTemp];
        i = i+1;
    else
        cd ..
        i = i+1;
    end

close all;

figure; hist(V)
xlabel('Velocity at release (nm/s)')
ylabel('Counts')
figure; hist(F)
xlabel('Force at release (pN)')
ylabel('Counts')
dlmwrite('vatstall_data.txt',[V F])
```

```
function [V F] = vatstall_run( Vfile,sampleRate )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cd events;
eventCounter = size(dlmread([Vfile(1:length(Vfile)-4),...
    '_Event_Indices.txt']),1);
cd ..
iEvent = 0;
clear FstallsRun;

for i = 1:eventCounter

    cd events; cd stalls

    rawDataFile = [Vfile(1:length(Vfile)-4),'_Event_',num2str(i),...
        '_ClippedEvent.txt'];

    if exist( rawDataFile )
        iEvent = iEvent + 1;
        cd ..; cd ..
        [V(:,iEvent) F(:,iEvent)] = vatstall_event( rawDataFile,...
            sampleRate,i );
     else
        cd ..; cd ..
    end


end



function [V F] = vatstall_event( rawDataFile,sampleRate,i )

cd events
cd stalls
```

```
rawData = dlmread( rawDataFile );
tdata = [0:1/sampleRate:(1/sampleRate*(length(rawData)-1))]';
dt = tdata(2) - tdata(1);
nmdata = rawData(:,2:3);
fdata = rawData(:,4:5);

% Local Fit
test = 2;
while test == 2
    plot(sgolayfilt(nmdata(:,2),0,21),'g')
    [x,y] = ginput(2);
    x = ceil(x);
    if x(2) > length(nmdata)
        while x(2) > length(nmdata)
            [x,y] = ginput(2);
        end
    else
    end
    pl=polyfit(tdata(x(1):x(2)),nmdata(x(1):x(2),2),1);
    F = fdata(length(fdata)-30:length(fdata)); %Average force over 15ms
%                                                                   (¬1 step)
    V = pl(1);
    if V < 0
        V = 0;
    else
    end

    plot(tdata,nmdata(:,2),'b',tdata,sgolayfilt(nmdata(:,2),0,21),'g',...
        tdata,pl(1)*tdata+pl(2),'r')
    title(num2str(['Event ',num2str(i)]))
    test = input('Are you satisfied with the fit? Yes=1 No=2 ')
end

cd ..
cd ..
```

B.2 Unloaded Data Analysis

B.2.1 Video Resizing

```
close all
clear all
clc
warning('off')

fname='07132010-CS-1E5-11';
ainfo=aviinfo([fname,'.avi']);
numframes=ainfo.NumFrames;
fps=ainfo.FramesPerSecond;
v = aviread([fname,'.avi'],1);
disp('Pick the corners of the part of the movie to crop')
imshow(ind2gray(v.cdata,v.colormap))
[x y] = ginput(2);
newname = [fname,'_resize.avi'];
aviobj = avifile(newname,'fps',fps);
    for findexnew = 1:1355
        frame = aviread(fname,findexnew);
        frame.cdata = frame.cdata(y(1):y(2),x(1):x(2));
%         imshow(ind2gray(v.cdata,v.colormap))
        aviobj = addframe(aviobj,frame);
    end
    aviobj = close(aviobj);
```

B.2.2 Video Segmentation and Bead Tracking

```
close all
clear all
clc
warning('off')
```

```
fname='07132010-CS-1E5-11_resize';
ainfo=aviinfo([fname,'.avi']);
numframes=ainfo.NumFrames;
fps=ainfo.FramesPerSecond;
pxnm = 30.0295; %pixel to nm conversion, nm/px

check = input(['Create segments or load previously cut segments? 1 for'...
    'create 2 for load. ']);

if check == 1
go = 1;
num = 1;
while go == 1
%     lind = 1;
%     for findex=begin:numframes
%         f = aviread(fname,findex);
%         imshow(ind2gray(f.cdata,f.colormap))
%         title(['frame ',num2str(findex)])
%         if lind == 1
%             start = input('Start at this frame?  1 for yes 2 for no. ');
%             if start == 1
%                 start = findex;
%                 lind = 2;
%             end
%         end
%         if lind == 2
%             stop = input('End at this frame?  1 for yes 2 for no. ');
%             if stop == 1
%                 stop = findex;
%                 lind = 3;
%             break
%             end
%         end
%     end
    start = input('Enter the frame at which to begin analysis. ');
    stop = input('Enter the frame at which to stop analysis. ');
```

```
        newname = [fname,'_cut',num2str(num),'.avi'];
        aviobj = avifile(newname,'fps',fps);
        for findexnew = start:stop
            frame = aviread(fname,findexnew);
            aviobj = addframe(aviobj,frame);
        end
        aviobj = close(aviobj);
        go = input(['Would you like to create another run  segment?  1 for'...
            ' yes 2 for no. ']);
        begin = stop+1;
        num = num+1;
    end

    vids = num-1;
    else
        vids = input('Number of segments. ');
    end
    vel = [];
    dist = [];
    run = [];
    colors=prism(vids);
    for num = 1:vids
        x = [];
        y = [];
        DATA = [];
        fnametemp=[fname,'_cut',num2str(num)];
        ainfo=aviinfo([fnametemp,'.avi']);
        numframes=ainfo.NumFrames;
        fps=ainfo.FramesPerSecond;
        for findex = 1:numframes
%           Step through frame by frame - loading entire avi into memory is not
%           great since most of these are > 1/2 Gb
            fram=aviread(fnametemp,findex);
%           Spatial Filtering - seems like 20-30 pixels is a good number
            bw = im2bw(double(fram.cdata),fram.colormap,0.85); % Binary
            bw2 = bwmorph(bw,'majority',Inf); %use this for all other vids
```

```
%         bw2 = bwmorph(bw,'majority',1000); %use these for #3
        bw2 = bwmorph(bw2,'dilate',4);
%         imshow(bw2)
%         pause
        [L,qty] = bwlabeln(bw2,8); % Label objects on frame
        s = regionprops(L, 'Centroid'); % Objects' properties
        centrd = cat(1, s.Centroid); % Centroid
        ArrayParticle = [1:qty]'; % Object number
        DATA_temp.frame = findex; % Frame number
        DATA_temp.position = [ArrayParticle, centrd]; % Objects' positions
        DATA = [DATA, DATA_temp]; % Accumulate data
        x = [x; DATA(1,findex).position(:,2)]; %Creates array that shows x
%                                    position of all beads in all frames
        y = [y; DATA(1,findex).position(:,3)]; %Creates array that shows x
%                                    position of all beads in all frames
        %sfram=bpass(fram.cdata,1,25);

%       Peak finding - want to set a max height to detect at about 70%
%       of the max height.  Looking for objects ¬ 20 pixels in diameter
%       should drop below if we start seeing multiple objects

%         imMax=max(max(sfram));
%         pk=pkfnd(sfram,imMax*0.7,20);

%       Find the centroid of the blob
%       Good choice of size is ¬2*bpass setting, must be odd
%         cnt=cntrd(sfram,pk,45);

        disttemp = pxnm*sqrt(((x(findex)-x(1))^2)+((y(findex)-y(1))^2));
        dist = [dist disttemp];
    end
    t = 0:1/fps:(numframes-1)/fps;
    subplot(2,1,1),plot(x,y,'Color',colors(num,1:3))
    xlabel('X position (nm)')
    ylabel('Y position (nm)')
    hold on
    subplot(2,1,2),plot(t,dist,'Color',colors(num,1:3))
    xlabel('Time (s)')
    ylabel('Distance (nm)')
    hold on
    veltemp = polyfit(t,dist,1);
    vel = [vel; veltemp(1)];
    run = [run; dist(length(dist))];
    dlmwrite([fname,'_run',num2str(num),'.txt'],[t' dist'])
    dist = [];
end
hold off
% dlmwrite([fname,'_unloaded.txt'],[vel run])


% plot(bp(:,1),480-bp(:,2))  %Images upper left corner is 0,0 - adjust to
% make these agree
% xlabel('X position (pixels)')
% ylabel('Y position (pixels)')
```