Open source single molecule force spectroscopy

A Thesis

Submitted to the Faculty

of

Drexel University

by

William Trevor King

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

June 2013

Dedications

Heaven Malaika King-Bailey

January 21st, 2006 – June 6th, 2010

What a bundle of joy.

We'll miss you, love.

Acknowledgments

If I have seen further it is by standing on ye sholders of Giants.

Isaac Newton[1]

Newton's competitive spirit gives the heroic phrasing in the quote above, but I don't see the need to use giants (Fig. 1). Any furtherance of knowledge due to my research is due to consolidating, reorganizing, and tweaking the work of those that came before me. To all of you in the rest of this innumerable bundle, thanks!

On a professional level, my high school physics teacher Tom Hoch started this whole experience with a very engaging class, and I continue to use his block counting analogy for energy conservation when teaching new students. Since then, I have been privileged to work under Joe Amato, Dan Schult, Jeff Buboltz, the late Marc Feldman, and the late Guoliang Yang, all of whom have helped mold me into the scientist and person I am today. Prof. Yang deserves extra credit, and his ability to shift topics from force spectroscopy to geopolitics and back in a minute helped keep things fresh. Thanks to Luis Cruz Cruz for shepherding me through the final stages of my thesis. My lab partners Marisa Roman and Runcong Liu provided lively discussions and a (mostly) willing audience for smoke testing my software.

On a personal level, my wife Emily and broader family and friends have kept me sane throughout this process. Thank you all.

Thanks to my committee members—Jian-Min Yuan, Jun Xi, Michel Vallières, Som Tyagi, and Prof. Cruz—for helpful criticism and review. Also thanks to Mom, for lots of pro bono editing! Any errors that remain are clearly my fault, and corrections are welcome[1].

This work was produced using the following open source software projects:

TEXlive/CTAN Typesetting.

PGF/Asymptote Graphics programming.

[1] Write me at wking@tremily.us. You can browse the thesis source at `http://git.tremily.us/?p=thesis.git`.

Figure 1: A raft of 500 fire ants, reproduced from Mlot et al. [3].

Python General purpose scripting.

SCons Build manager.

GNU Emacs Text editor.

Git Version control.

GNU/Linux/Gentoo Operating system and distribution.

and many, many more. I am deeply indebted to all of the smart, generous people who produce such wonderful tools. Besides providing the tools, they also provide very supportive communities to help haul complete newbies like me up the learning curve.

sawsim work was supported by National Institutes of Health Grants R01-GM071793.

Table of Contents

List of Tables

List of Figures

Abstract

Open source single molecule force spectroscopy
William Trevor King
Guoliang Yang, Ph.D. and Luis Cruz Cruz, Ph.D.

Single molecule force spectroscopy (SMFS) experiments provide an experimental benchmark for testing simulated and theoretical predictions of protein unfolding behavior. Despite it use since 1997[30], the labs currently engaged in SMFS use in-house software and procedures for critical tasks such as cantilever calibration and Monte Carlo unfolding simulation. Besides wasting developer time producing and maintaining redundant implementations, the lack of transparency makes it more difficult to share data and techniques between labs, which slows progress. In some cases it can also lead to ambiguity as to which of several similar approaches, correction factors, etc. were used in a particular paper.

In this thesis, I introduce an SMFS sofware suite for cantilever calibration (calibcant), experiment control (unfold-protein), analysis (Hooke), and postprocessing (sawsim) in the context of velocity clamp unfolding of I27 octomers in buffers with varying concentrations of $CaCl_2$[2,5,20,21]. All of the tools are licensed under open source licenses, which allows SMFS researchers to centralize future development. Where possible, care has been taken to keep these packages operating system (OS) agnostic. The experiment logic in unfold-protein and calibcant is still nominally OS agnostic, but those packages depend on more fundamental packages that control the physical hardware in use[4]. At the bottom of the physical-interface stack are the Comedi drivers from the Linux kernel[31]. Users running other operating systems should be able to swap in analogous low level physical-interface packages if Linux is not an option.

Chapter 1: Introduction

Single molecule force spectroscopy (SMFS) is the study of folding and unfolding transitions in proteins under tension. By measuring these transitions, we hope to gain insight into fundamental protein behavior. SMFS is an attempt to bridge the gap between chemists studying folding and unfolding kinetics in bulk solutions and theorists simulating protein behavior at the amino-acid level. An increased understanding of protein folding would guide researchers in developing drugs targeting biologically significant receptors and enzymes. In this chapter, I describe the protein folding problem in a general sense (Section 1.1), discuss theoretical frameworks for understanding protein folding (Section 1.2), highlight the role of SMFS in extending this understanding (Section 1.3), and explain the role of unfolding experiments in understanding protein folding (Section 1.4). The last section in this chapter gives a roadmap for the rest of the thesis (Section 1.5).

1.1 The protein folding problem

In biological systems the most important molecules, such as proteins, nucleic acids, and polysaccharides, are all polymers. Understanding the properties and functions of these polymeric molecules is crucial in understanding the molecular mechanisms behind structures and processes in cells.

An organism's genetic code is stored in DNA in the cell nucleus. DNA sequencing is a fairly well developed field, with fundamental work such as the Human Genome Project seeing major development in the early 2000s[32–34]. It is estimated that human genetic information contains approximately 25,000 genes, each encoding a protein[35,36]. Knowing the amino acid sequence for a particular protein, however, does not immediately shed light on the protein's role in the body, or even the protein's probable conformation. Indeed, a protein's conformation is often vitally important in executing its biological tasks (Fig. 1.1). Unfortunately predicting a protein's stable conformations from it's amino acid sequence has proven to be remarkably difficult, as has the inverse problem of finding sequences that form a given conformation.

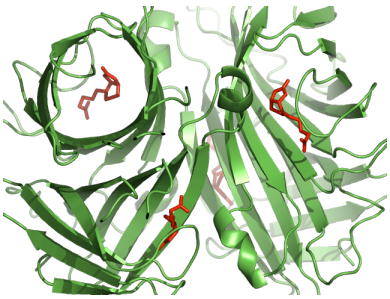


Figure 1.1: Complex of biotin (red) and a streptavidin tetramer (green) (PDB ID: 1SWE)[9]. The correct streptavidin conformation creates the biotin-specific binding pockets. Biotin-streptavidin is a model ligand-receptor pair isolated from the bacterium *Streptomyces avidinii*. Streptavidin binds to cell surfaces, and bound biotin increases streptavidin's cell-binding affinity[10]. Figure generated with PyMol.

1.2 Protein folding energy landscapes

Finding a protein's lowest energy state via a brute force sampling of all possible conformations is impossibly inefficient, due to the exponential scaling of possible conformations with protein length, as outlined by Levinthal[37]. This has lead to a succession of models explaining the folding mechanism. For a number of years, the "pathway" model of protein folding enjoyed popularity (Fig. 1.2a)[37]. More recently, the "landscape" or "funnel" model has come to the fore (Fig. 1.2b)[13]. Both of these models reduce the conformation space to a more approachable analog, and their success depends on striking a useful balance between simplicity and accuracy.

When the choice of theoretical approach becomes murky, you must gather experimental data to help distinguish between similar models. Separating the pathway model from the funnel model is only marginally within the realm of current experimental techniques, but with higher throughput and increased automation it should be easier to make such distinctions in the near future.

1.3 Why *single* molecule?

The large size of proteins relative to simpler molecules limits the information attainable from bulk measurements, because the macromolecules in a population can have diverse conformations and behaviors. Bulk measurements average over these differences, producing excellent statistics for the mean, but making it difficult to understand the variation. The individualized, and sometimes rare,

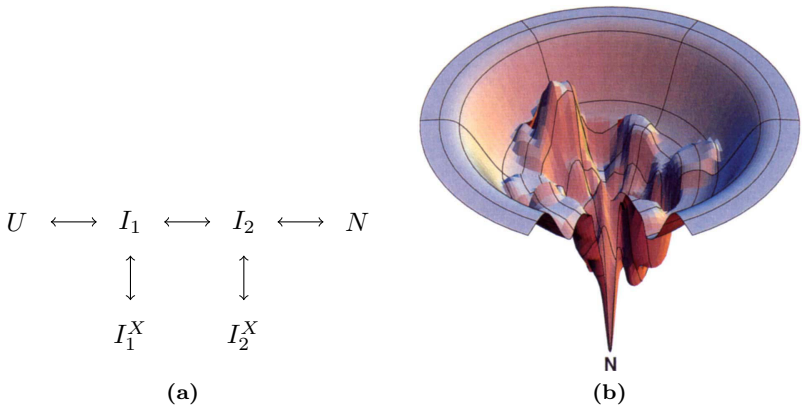


Figure 1.2: (a) A "double T" example of the pathway model of protein folding, in which the protein proceeds from the native state N to the unfolded state U via a series of metastable transition states I_1 and I_2 with two "dead end" states I_1^X and I_2^X. Adapted from Bédard et al.[12]. (b) The landscape model of protein folding, in which the protein diffuses through a multi-dimensional free energy landscape. Separate folding attempts may take many distinct routes through this landscape on the way to the folded state. Reproduced from Dill and Chan[13].

behaviors of macromolecules can have important implications for their functions inside the cell. Single molecule techniques, in which the macromolecules are studied one at a time, allow direct access to the variation within the population without averaging. This provides important and complementary information about the functional mechanisms of several biological systems[38].

Single molecule techniques provide an opportunity to study protein folding and unfolding at the level of a single molecule, where the distinction between the pathway model and funnel model is clearer. They also provide a convenient benchmark for verifying molecular dynamics simulations, because it takes lots of computing power to simulate even one biopolymer with anything close to atomic resolution over experimental time scales. Even with significant computing resources, comparing molecular dynamics results with experimental data remains elusive. For example, experimental pulling speeds are on the order of μm/s, while simulation pulling speeds are on the order of m/s[39–43].

Single molecule techniques for manipulating biopolymers include optical measurements, *i.e.*, single molecule fluorescence microscopy and spectroscopy, and mechanical manipulations of individual macromolecules, *i.e.*, force microscopy and spectroscopy using atomic force microscopes (AFMs), laser tweezers[44,45], magnetic tweezers[46], biomembrane force probes[47], and centrifugal mi-

croscopes[48]. These techniques cover a wide range of approaches, and even when the basic approach is the same (e.g. force microscopy), the different techniques span orders of magnitude in the range of their controllable parameters.

1.4 Why *un*folding?

There's a lot of talk about protein *folding* in this chapter, while the rest of the thesis (and the title) are about *unfolding*. If you understand protein folding, you can use your understanding to design drugs with a particular conformation, or predict the conformation of a biologically important receptor (Section 1.1). Understanding protein unfolding is less directly useful, because unfolded proteins are rarely biologically relevant (although it does happen[49]).

The focus on unfolding is mainly because it's easier to unravel proteins by pulling on their ends (Section 2.4) than it is to fold them into their native state by pushing on those ends (Figs. 1.1 and 2.3). For proteins with smooth enough energy landscapes, the folding and unfolding routes will be similar, so knowledge about the unfolding behavior *does* shed light on the folding behavior.

Practically, the distinction between folding and unfolding makes little difference, because drug designers and doctors are not consuming SMFS results directly. For researchers calibrating molecular dynamics simulations, it doesn't matter if you compare simulated folding experiments with experimental folding experiments, or simulated unfolding experiments with experimental unfolding experiments. The important thing is to compare your simulation against *some* experimental benchmarks. If your molecular dynamics simulation successfully predicts a protein's unfolding behavior, it makes me more confident that it will correctly predict the protein's native folding behavior.

1.5 Thesis outline

Chapter 2 of this thesis outlines the apparatus and methods for single molecule force spectroscopy with an atomic force microscope. Chapter 3 presents my sawsim Monte Carlo simulation for modeling unfolding/refolding behavior. By comparing model simulations with experimental measurements, we can gain insight into the protein's kinetics. After Chapter 3, you should have a pretty firm grasp of the underlying physics, so we'll move on to Chapter 4 and discuss my pyafm and unfold-protein

experiment control software. With both the kinetic theory and procedure taken care of, Chapter 5 discusses thermal cantilever calibration, deriving the theoretical approach and presenting my calibcant automatic calibration software.

Moving away from experiment control, Chapter 6 presents the Hooke suite for extracting unfolding force histograms (for comparison with sawsim simulations). In Chapter 7, I pull all the pieces together (experiment control, post processing, and simulation) to carry out unfolding experiments on the immunoglobulin-like domain 27 from human Titin (I27) in buffers with different ionic strength. We close with Chapter 8, which summarizes my conclusions and discusses possible directions for future work.

Chapter 2: Mechanical protein unfolding via AFM

In this chapter we will review the basic methods and procedures for mechanically unfolding proteins with an atomic force microscope. We will discuss the working principle behind an AFM (Section 2.1) and outline the procedure for synthesizing protein chains (Section 2.2). With the groundwork out of the way, we will look at sample preparation (Section 2.3) and the velocity clamp force spectroscopy procedure (Section 2.4). Finally, we will give a summary of cantilever calibration (Section 2.5) which is discussed in more detail in Appendix A.

Everything discussed in this chapter, with the possible exception of cantilever calibration, is fairly standard practice in the field of force spectroscopy. See Appendix A for the development of the cantilever calibration theory from first principles and references to related papers.

2.1 Instrumentation

Of the mechanical manipulation methods listed in Section 1.3, AFM is the most widely used due to the availability of user-friendly commercial instruments. AFM has been employed on several types of biological macromolecules, mechanically unfolding proteins[50] and forcing structural transitions in DNA[27,51] and polysaccharides[30].

An AFM uses a sharp tip integrated at the end of a cantilever to interact with the sample[52]. Cantilever bending is measured by a laser reflected off the cantilever and incident on a position sensitive photodetector[53] (Fig. 2.1a). When the bending force constant of the cantilever is known[54], the force applied to the sample can be calculated using Hooke's law (Eq. (3.3)).

The substrate is mounted in a fluid cell[55,56] on a three dimensional piezoelectric actuator so that the tip may be positioned on the surface with sub-nanometer resolution (although signal drift and piezo hysteresis can cause larger errors in the positioning accuracy). Our tubular piezo has a range of 1.6 μm in the horizontal directions and a range of 3.5 μm in the vertical (Fig. 2.1b).

The forces that can be applied and measured with an AFM range from tens of piconewtons to

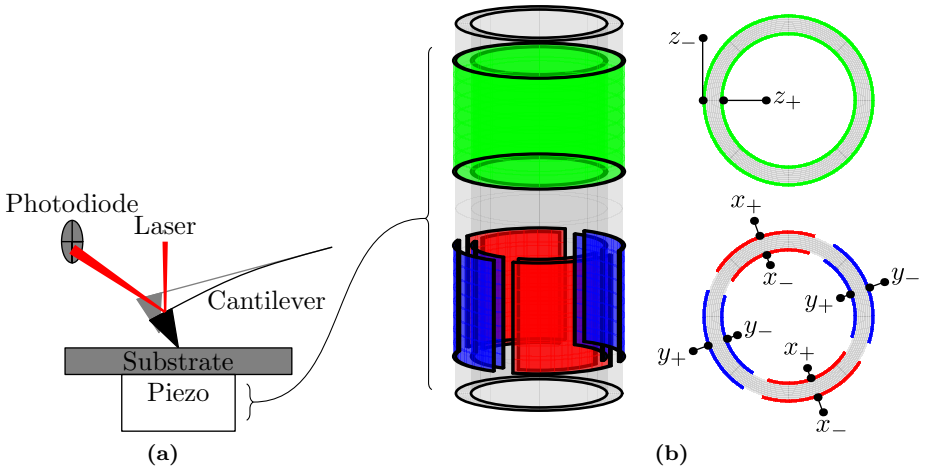


Figure 2.1: (a) Operating principle for an Atomic Force Microscope. A sharp tip integrated at the end of a cantilever interacts with the sample. Cantilever bending is measured by a laser reflected off the cantilever and incident on a position sensitive photodetector. (b) Schematic of a tubular piezoelectric actuator. In our AFM, the substrate is mounted on the top end of the tube, and the bottom end is fixed to the microscope body. This allows the piezo to control the relative position between the substrate and the AFM cantilever. The electrodes are placed so radial electric fields can be easily generated. These radial fields will cause the piezo to expand or contract axially. The z voltage causes the tube to expand and contract uniformly in the axial direction. The x and y voltages cause expansion on one side of the tube, and contraction (because of the reversed polarity) on the other side of the tube. This tilts the tube, shifting the sample horizontally.

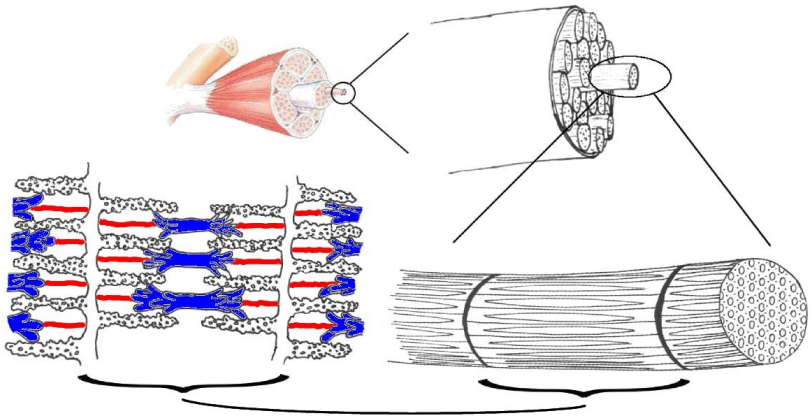


Figure 2.2: Biological role of titin. Moving clockwise from the upper left you can see a bone/muscle group, a muscle fiber, a myofibril, and a sarcomere. In the sarcomere, the white, knobbly filaments are actin. The myosin bundles are blue, and the titin linkers are red. When the muscle contracts, the myosin heads walk up the actin filiaments, shortening the sarcomere. When the muscle relaxes, the myosin heads release the actin filimants and slide back, lengthening the sarcomere. Titin functions as an entropic spring that keeps the myosin from falling out of place during the passive, relaxed stage. This figure is adapted from Wikipedia [14].

hundreds of nanonewtons. The investigation of the unfolding and refolding processes of individual protein molecules by the AFM is feasible because many globular proteins unfold under external forces in this range. Since elucidating the mechanism of protein folding is currently one of the most important problems in biological sciences, the potential of the AFM for revealing significant and unique information about protein folding has stimulated much effort in both experimental and theoretical research.

2.2 Protein polymer synthesis—Titin I27

Early experiments in force spectroscopy involved DNA [27,57], but before long they were also investigating proteins. Native titin was one of the first proteins studied with force spectroscopy [30]. Titin is a muscle protein involved in passive elasticity (Fig. 2.2), so it is an ideal subject when examining the effect of mechanical force [58]. Titin is also interesting because, while it is one of the largest known proteins, it is composed of a series of globular domains. When Rief et al. [30] carried out their seminal unfolding experiment, they observed a very characteristic sawtooth as the domains unfolded (see Section 2.4 for a discussion of these sawteeth).

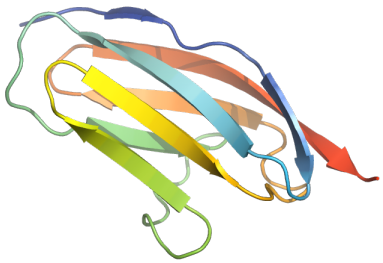


Figure 2.3: I27, the immunoglobulin-like domain 27 from human titin (PDB ID: 1TIT)[15]. The entire domain is 4.7 nm from end to end. Figure generated with PyMol.

Unfortunately, it is difficult to analyze the unfolding of native titin, because the heterogenous globular domains make it hard to attribute a particular subdomain to a partuclar unfolding event. Unfolding a single domain is not feasable because the large radius of curvature of an AFM tip ($\sim$ 20 nm[59]) dwarfs the radius of a globular domain ($\sim$ 2 nm[15]). When such a large tip is so close to the substrate, van der Waals forces and non-specific binding with the surface dominate the tip-surface interaction. In order to increase the tip-surface distance while preserving single molecule analysis, Carrion-Vazquez et al.[6] synthesized a protein composed of eight repeats of immunoglobulin-like domain 27 (I27), one of the globular domains from native titin (Fig. 2.3). Octameric I27 produced using their procedure is now available commercially[60].

Synthetic proteins are generally produced by creating a plasmid coding for the target protein, inserting the plasmid in a bacteria, waiting while the bacteria produce your protein, and then purifying your proteins from the resulting culture. In this case, Carrion-Vazquez et al.[6] extracted messenger RNA coding for titin from human cardiac tissue[30], and used reverse transcriptase to generate a complementary DNA (cDNA) library from human cardiac muscle messenger RNA. This cDNA is then amplified using the polymerase chain reaction (PCR), with special primers that allow you to splice the resulting cDNA into a plasmid (which ends up with one I27). Then they ran another PCR on the plasmid, linearized the plasmid with two restriction enzymes, and grafted two I27-containing sections together to form a new plasmid (now with two I27s, Fig. 2.4). Another PCR-split-join cycle produced a plasmid with four I27s, and a final cycle produced a plasmid with eight. The eventual plasmid vector has the eight I27s and a host-specific promoter that causes the

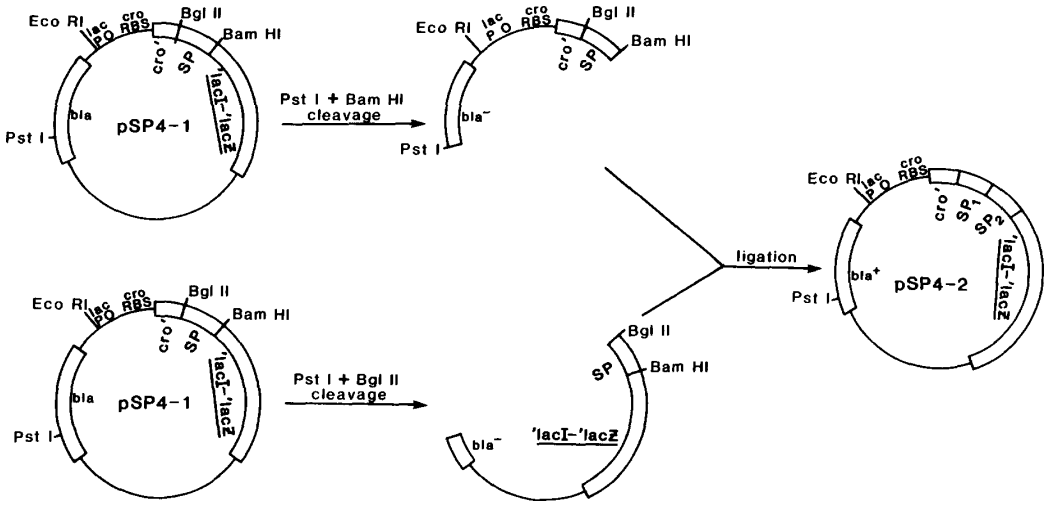


Figure 2.4: Example of gene duplication via plasmid splicing (Kempe et al. [16] Fig. 2). Kempe et al. [16] use a different gene, but some of the restriction enzymes are shared with Carrion-Vazquez et al. [6]. The overall approach is identical.

bacteria to produce large quantities of I27. The exact structure of the generated octamer is [6]

$$\text{Met-Arg-Gly-Ser-(His)}_6\text{-Gly-Ser-(I27-Arg-Ser)}_7\text{-I27-...-Cys-Cys}$$

The plasmid is then transformed into the host, usually *Escherichia coli* [6,61,62] or a proprietary equivalent such as Agilent's SURE 2 Supercompetent Cells [63,64]. The infected cells are cultured to express the protein.

The octamer is then purified from the culture using immobilized metal ion affinity chromatography (IMAC), where the His-tagged end of the octamer covalently bonds to a metal ion that is bound to the column media (e.g. Ni-NTA coated beads) [61,62,64]. Once the rest of the broth has been washed out of the chromatography column, the octamer is eluted via either another molecule which competes for the metal ions [62] or by changing the pH so the octamer is less attracted to the metal ion.

2.3 Sample preparation

In mechanical unfolding experiments, one end of the protein is bound to a substrate and the other binds to the AFM tip. This allows you to stretch the protein by increasing the tip-substrate distance using the piezo. A common approach is to synthesize proteins with cystine residues on one end

(Section 2.2) and allow the cystines to bind to a gold surface[6,64,65].

We prepare gold-surfaces by sputtering gold onto freshly cleaved mica sheets in a vacuum. The mica keeps the gold surface protected from contamination until it is needed. In order to mount the gold on our AFM, we glue glass coverslips to the gold using a two part epoxy. Instead of using mica to protect the gold surface, some labs evaporate the gold directly onto the coverslips immediately before running an experiment[6].

When it is time to deposit proteins on the surface, we peel a coverslip off the gold-coated mica, exposing the gold surface that had previously been attached to the mica. We dispense 5 μL of I27 solution (65 g/μL) on the freshly-exposed gold, followed by 5 μL of phosphate buffered saline (PBS). We allow the protein to bind to the gold surface for 30 mintues and then load the coated coverslips into our AFM fluid cell. There are a number of similar PBS recipes in common use[27,64,66,67], but our PBS is diluted from 10x PBS stock composed of 1260 mM NaCl, 72 mM Na_2HPO_4, and 30 mM NaH_2PO_4[17].

As an alternative to binding proteins to gold, others have used EGTA[68], Ni-NTA[43,69–71], or silanized glass[62,72]. Some groups have also functionalized the cantilever tips by coating them with molecules designed to bind to the protein[73]. Of these, a Ni-NTA coating is the most popular[74].

2.4 Mechanical unfolding experiments

In a mechanical unfolding experiment, a protein polymer is tethered between two surfaces: a flat substrate and an AFM tip. The polymer is stretched by increasing the separation between the two surfaces (Fig. 2.5a). The most common mode is the constant speed experiment in which the substrate surface is moved away from the tip at a uniform rate. The tethering surfaces, *i.e.*, the AFM tip and the substrate, have much larger radii of curvature than the dimensions of single domain globular proteins that are normally used for folding studies. This causes difficulties in manipulating individual protein molecules because nonspecific interactions between the AFM tip and the substrate may be stronger than the forces required to unfold the protein when the surfaces are a few nanometers apart. To circumvent these difficulties, globular protein molecules are linked into polymers, which are then used in the AFM studies[17,19,50]. When such a polymer is pulled from

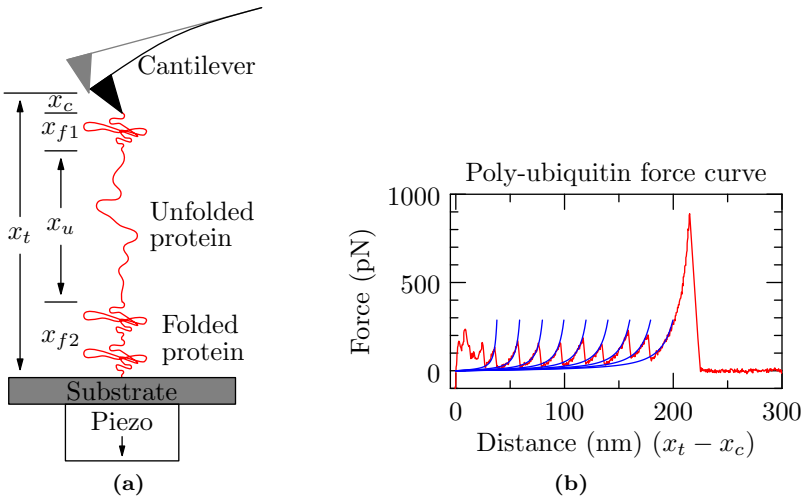


Figure 2.5: (a) Schematic of the experimental setup for mechanical unfolding of proteins using an AFM (not to scale). An experiment starts with the tip in contact with the substrate surface, which is then moved away from the tip at a constant speed. x_t is the distance traveled by the substrate, x_c is the cantilever deflection, x_u is the extension of the unfolded polymer, and $x_f = x_{f1} + x_{f2}$ is the extension of the folded polymer. (b) An experimental force curve from stretching a ubiquitin polymer (—) with the rising parts of the peaks fitted to the WLC model (—, Section 3.2.1)[17]. The pulling speed used was 1 μm/s. The irregular features at the beginning of the curve are due to nonspecific interactions between the tip and the substrate surface, and the last high force peak is caused by the detachment of the polymer from the tip or the substrate surface. Note that the abscissa is the extension of the protein chain $x_t - x_c$.

its ends, each protein molecule feels the externally applied force, which increases the probability of unfolding by reducing the free energy barrier between the native and unfolded states. The unfolding of one molecule in the polymer causes a sudden lengthening of the polymer chain, which reduces the force on each protein molecule and prevents another unfolding event from occurring immediately. The force versus extension relationship, or *force curve*, shows a typical sawtooth pattern (Fig. 2.5b), where each peak corresponds to the unfolding of a single protein domain in the polymer. Therefore, the individual unfolding events are separated from each other in space and time, allowing single molecule resolution despite the use of multi-domain test proteins.

2.5 Cantilever spring constant calibration

In order to measure forces accurately with an AFM, it is important to measure the cantilever spring constant κ. The force exerted on the cantilever can then be deduced from its deflection via Hooke's

law,

$$F = -\kappa x \ , \tag{2.1}$$

where x is the perpendicular displacement of the cantilever tip (x_c in Fig. 2.5a).

The basic idea is to use the equipartition theorem, which gives the thermal energy per degree of freedom. For a simple harmonic oscillator, the only degree of freedom is x, so we have[28]

$$\frac{1}{2}\kappa \left\langle x^2 \right\rangle = \frac{1}{2} k_B T \ , \tag{2.2}$$

where k_B is Boltzmann's constant, T is the absolute temperature, and $\left\langle x^2 \right\rangle$ is the average value of x^2 measured over a long time interval.

To calculate the spring constant κ using Eq. (2.2), we need to measure the buffer temperature T and the thermal vibration variance $\left\langle x^2 \right\rangle$. We measure the temperature with a thermocouple inserted into the AFM fluid cell, and we measure the thermal vibration by monitoring the cantilever during thermal oscillation when it is far from the substrate surface.

The raw cantilever deflection data will have sources of noise that are not due to the cantilever's thermal vibration (e.g. electronic noise in the detector). To avoid biasing κ, there is a fairly elaborate theory behind extracting $\left\langle x^2 \right\rangle$. For more detail, see Chapter 5, where I discuss the $\left\langle x^2 \right\rangle$ extraction in detail and present my open source calibcant tool for automated cantilever calibration.

Chapter 3: Monte Carlo mechanical unfolding simulation

Unfolding proteins by pulling on them with an AFM yields a series of force curves (Fig. 2.5b). How do we get from the unfolding curves to a deeper understanding of the unfolding physics? In this chapter, I present my sawsim simulator, which performs discrete-state unfolding simulations using theoretical models of protein unfolding. By finding the models and parameters that best reproduce the experimental data, we can find unforced unfolding rates and other unfolded polymer distances that can be cross-checked in chemical unfolding experiments and used to validate more detailed molecular dynamics simulations. The sawsim simulations discussed here are carried out after the unfolding experiments (Chapter 4), cantilever calibration (Chapter 5), and post-processing (Chapter 6), but I discuss sawsim first because it provides the cleanest description of the underlying physical processes. After we cover the theory here, we will be better prepared for the realistic complications discussed in the following chapters.

Much theoretical and computational work has been done in order to extract information about the structural, kinetic, and energetic properties of the protein molecules from the experimental data of force-induced protein unfolding measurements. Steered molecular dynamics simulations[39,40], as well as calculations and simulations using lattice[75,76] and off-lattice models[77,78], have provided insights into structural and energetic changes during force-induced protein unfolding. However, these simulations often involve time scales that are orders of magnitude smaller than those of the experiments (Section 1.3), and the parameters used in the calculations are often neither experimentally controllable nor measurable. As a result, a Monte Carlo simulation approach based on a simple two-state kinetic model for the protein is usually used to analyze data from mechanical unfolding experiments. A comparison of the force curves measured experimentally and those generated from simulations can yield the unfolding rate constant of the protein in the absence of force as well as the distance from the native state to the transition state along the pulling direction. The Monte Carlo simulation method has been used since the first report of mechanical unfolding experiments

using the AFM[6,30,79–83], but these previous implementations are neither fully described nor publicly available.

To fill this gap, I developed the sawsim simulation program[2]. In this chapter, I provide a detailed description of the simulation procedure, including the theories, approximations, and assumptions involved. I also explain the procedure for extracting kinetic properties of the protein from experimental data and introduce a quantitative measure of fit quality between simulation and experimental results. In addition, the effects of various experimental parameters on force curve appearance are demonstrated, and the errors associated with different methods of data pooling are discussed. These results should be useful in future experimental design, artifact identification, and data analysis for single molecule mechanical unfolding experiments.

3.1 Review of current research

There is a long history of protein unfolding and unbinding simulations. Early work by Grubmüller et al.[84] and Izrailev et al.[85] focused on molecular dynamics (MD) simulations of receptor-ligand breakage. Around the same time, Evans and Ritchie[86] introduced a Monte Carlo Kramers' simulation in the context of receptor-ligand breakage. The approach pioneered by Evans and Ritchie[86] was used as the basis for analysis of the initial protein unfolding experiments[30]. However, none of these earlier implementations were open source, which made it difficult to reuse or validate their results.

Within the Monte Carlo simulation approach, there are two main models for protein domain unfolding under tension: Bell's and Kramers'[87–89]. Bell introduced his model in the context of cell adhesion[90], but it has been widely used to model mechanical unfolding in proteins[6,30,87] due to its simplicity and ease of use[88] (Section 3.2.2). Kramers introduced his theory in the context of thermally activated barrier crossings, which is how we use it here (Section 3.2.2).

Evans introduced the saddle-point Kramers' approximation in a protein unfolding context in 1997 (Evans and Ritchie[86] Eq. (3)). However, early work on mechanical unfolding focused on the simpler Bell model[30]. In the early 2000's, the saddle-point/steepest-descent approximation to Kramer's model (Hänggi et al.[91] Eq. (4.56c)) was introduced into our field[92,93]. By the mid 2000's,

the full-blown double-integral form of Kramer's model (Hänggi et al.[91] Eq. (4.56b)) was in use[87].

There have been some tangential attempts towards even fancier models: Dudko et al.[92] attempted to reduce the restrictions of the single-unfolding-path model and Hyeon and Thirumalai[93] attempted to measure the local roughness using temperature dependent unfolding. However, further work on these lines has been slow, because the Bell model fits the data well despite its simplicity. For more complicated transition rate models to gain ground, we need larger, more detailed datasets that expose features which the Bell model doesn't capture.

3.2 Methods

In simulating the mechanical unfolding process, a force curve is generated by calculating the amount of cantilever bending as the substrate surface moves away from the tip. The cantilever bending is obtained by balancing the tension in the protein polymer and the Hookean force of the bent cantilever. The unfolding probability of the protein molecules in the polymer is then calculated for that tension, and whether an unfolding event occurs is determined according to a Monte Carlo method. The simulation was implemented in C, and there are a number of Python modules to facilitate running several simulations in parallel[1].

In the following sections, we'll discuss models used to determine the tension of a chain composed of several types of "domains" (e.g. one cantilever, three folded I27 domains, and seven unfolded I27 domains) (Section 3.2.1). We'll also work through a number of models for calculating the probability that a domain will transition from one state (e.g. folded I27) to another (e.g. unfolded I27) (Section 3.2.2).

3.2.1 Modeling polymer tension

The fundamental abstraction of the simulation is the "domain", which represents a discrete chunk of the flexible chain between the substrate and the cantilever holder (Fig. 3.1a). Each of these domains is assigned a particular state; for example, the domain representing the cantilever is assigned to the "cantilever" state, and the domains representing protein molecules are assigned to either the "folded" or the "unfolded" state. When balancing the tension along the chain, we assume that

[1] Source code available at `http://blog.tremily.us/posts/sawsim/`.

the spatial order of domains along the chain is irrelevant[96], so the domains can be rearranged and grouped by state (Fig. 3.1b). To determine the tension in the chain and the amount of cantilever bending when N states are populated, a system of $N+1$ equations with $N+1$ unknowns must be solved

$$F_i(x_i) = F_t \tag{3.1}$$

$$\sum_i x_i = x_t \ , \tag{3.2}$$

where F are tensions, x are extensions, and the subscripts i and t represent a particular state group and the total chain respectively (Fig. 2.5a). $F(x_t)$ may be computed from this system of equations using any multi-dimensional root-finding algorithm.

Hooke's law

Inside this framework, we chose a particular extension model $F_i(x_i)$ for each domain state. Cantilever elasticity is described by Hooke's law, which gives

$$F = \kappa_c x_c \ , \tag{3.3}$$

where κ_c is the bending spring constant and x_c is the deflection of the cantilever (Fig. 2.5a).

Wormlike chains

Unfolded domains are modeled as wormlike chains (WLCs)[30,64], which treat the unfolded polymer as an elastic rod of persistence length p and contour length L (Fig. 3.2). The relationship between tension F and extension (end-to-end distance) x is given by Bustamante's interpolation formula[57,97].

$$F_{\mathrm{WLC}}(x,p,L) = \frac{k_B T}{p}\left[\frac{1}{4}\left(\frac{1}{(1-x/L)^2}-1\right)+\frac{x}{L}\right] \ , \tag{3.4}$$

where p is the persistence length. This interpolation formula is accurate to within 7% of the exact F_{WLC} for $F_{\mathrm{WLC}} \approx k_B T/p_u$[97]. Because most unfolded proteins studied have persistence lengths on the order of the size of an amino acid ($p_u \approx 3.8$ Å[6,30,64]), this characteristic force works out to be

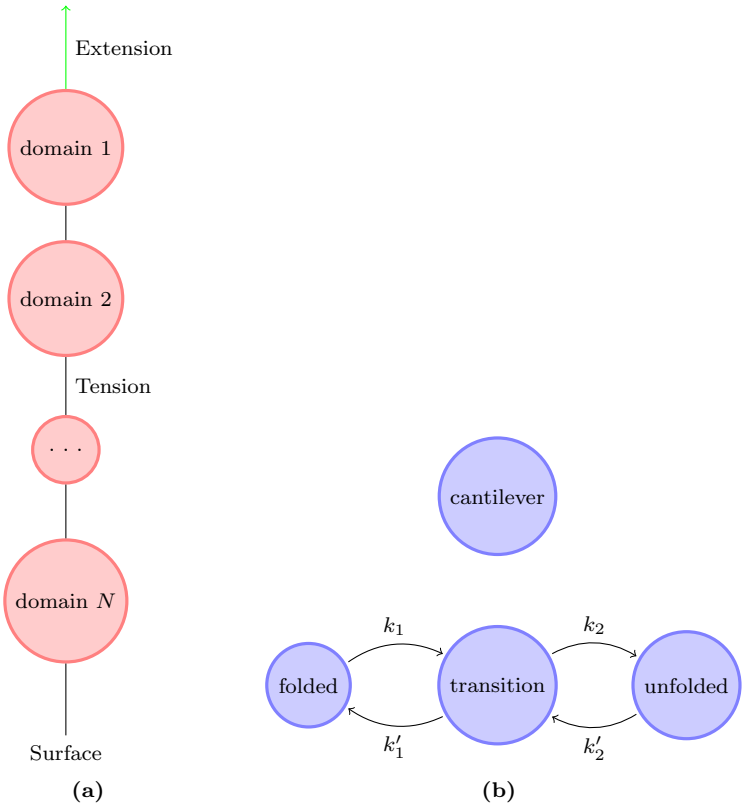


Figure 3.1: (a) Extending a chain of domains. One end of the chain is fixed, while the other is extended at a constant speed. The domains are coupled with rigid linkers, so the domains themselves must stretch to accomodate the extension. Compare with Fig. 2.5a. (b) Each domain exists in a discrete state. At each timestep, it may transition into another state following a user-defined state matrix such as this one, showing a metastable transition state and an explicit "cantilever" domain.

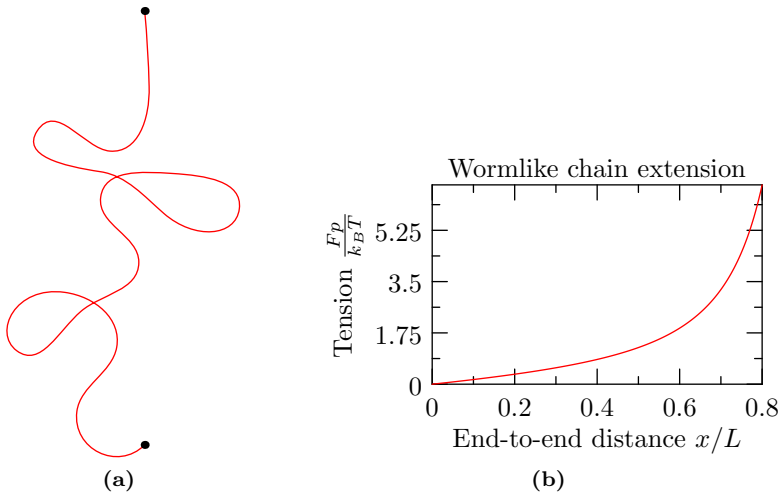


Figure 3.2: (a) The wormlike chain models a polymer as an elastic rod with persistence length p and contour length L. (b) Force vs. extension for a WLC using Bustamante's interpolation formula.

around 11 pN. Most proteins studied using force spectroscopy have unfolding forces in the hundreds of piconewtons, by which point the interpolation formula is in it's more accurate high-extension regime.

For chain with N_u unfolded domains sharing a persistence length p_u and per-domain contour lengths L_{u1}, the tension of the WLC is determine by summing the contour lengths

$$F(x, p_u, L_u, N_u) = F_{\mathrm{WLC}}(x, p_u, N_u L_{u1}) \; . \tag{3.5}$$

Folded domains

Short chains of folded proteins, however, are not easily described by polymer models. Several studies have used WLC and FJC models to fit the elastic properties of the modular protein titin[98,99], but native titin contains hundreds of folded and unfolded domains. For the short protein polymers common in mechanical unfolding experiments (Section 2.2), the cantilever dominates the elasticity of the polymer-cantilever system before any protein molecules unfold. After the first unfolding event occurs, the unfolded portion of the chain is already longer and softer than the sum of all the remaining folded domains, and dominates the elasticity of the whole chain. Therefore, the details

of the tension model chosen for the folded domains has negligible effect on the unfolding forces (Eq. (3.2)), which was also suggested by Staple et al. [100]. Force curves simulated using different models to describe the folded domains yielded almost identical unfolding force distributions (data not shown).

As an alternative to modeling the folded domains explicitly or ignoring them completely, another approach is to subtract the end-to-end length of the folded protein from the contour length of the unfolded protein to create an effective contour length for the unfolding [6]. This effectively models the folded domains as WLCs with the same persistence length as the unfolded domains.

Other models

The unfolded polypeptide chain has been shown to follow the WLC model quite well [30] (Section 3.2.1), though other polymer models have been tried. One alternative is the freely-jointed chain (FJC) [44,99,101,102], which models the polymer as a series of N rigid links, each of length l (the Kuhn length), which are free to rotate about their joints (Fig. 3.3).

$$F_{\mathrm{FJC}}(x, l, L) = \frac{k_B T}{l} \mathcal{L}^{-1}\left(\frac{x}{L}\right) , \tag{3.6}$$

where $L = Nl$ is the total length of the chain, and $\mathcal{L}(\alpha) \equiv \coth \alpha - \frac{1}{\alpha}$ is the Langevin function [103].

More exotic models such as elastic WLCs [101,104], elastic FJCs [101,105], and freely rotating chains [104] (FRCs) have also been used to model DNA and polysaccharides, but are rarely used to model the relatively short and inextensible synthetic proteins used in force spectroscopy.

Assumptions

In the simulation, the protein polymer is assumed to be stretched in the direction perpendicular to the substrate surface, which is a good approximation in most experimental situations, because the unfolded length of a protein molecule is much larger than that of the folded form. Therefore, after one molecule unfolds, the polymer becomes much longer and the angle between the polymer and the surface approaches 90 degrees [64].

The joints between domain groups are also assumed to lie along a line between the surface

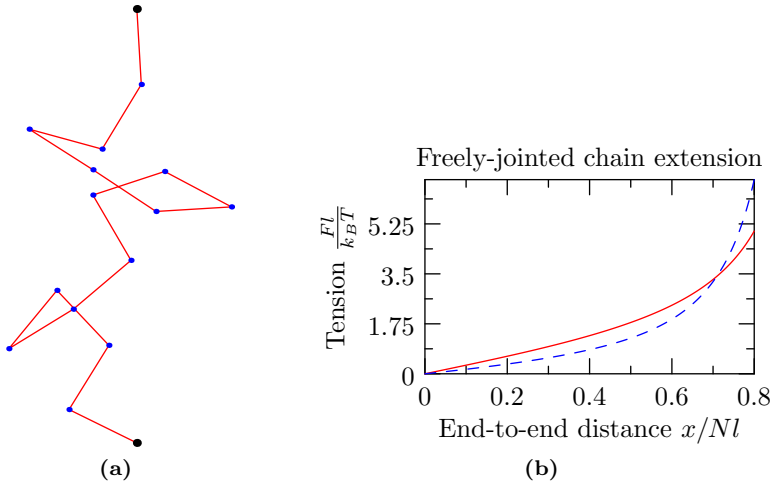


Figure 3.3: (a) The freely-jointed chain models the polymer as a series of N rigid links, each of length l, which are free to rotate about their joints. Each polymer state is a random walk, and the density of states for a given end-to-end distance is determined by the number of random walks that have such an end-to-end distance. (b) Force vs. extension for a hundred-segment FJC. The WLC extension curve (with $p = l$) is shown as a dashed line for comparison.

tether point and the position of the tip (Eq. (3.2) is scalar, not vector, addition). The effects of this assumption are also minimized due to greater length of the unfolded domain compared with the other domains (folded proteins and cantilever deflection). For example, a 0.050 N/m cantilever under 200 pN of tension will bend $x_c = F/\kappa_c = 4$ nm. The entire end-to-end length of folded domains such as I27 are also around 5 nm (Fig. 2.3). A single unfolded I27, with its 89 amino acids [15], should have an unfolded contour length of 89 aa $\cdot$ 0.38 nm = 33.8 nm, equivalent to a cantilever and five folded domains.

Velocity-clamp example

Consider an experiment pulling a polymer with N identical protein domains at a constant speed. At the start of an experiment, the chain is unstretched ($x_t = 0$), which means all the domains are unstretched, the cantilever is undeflected, and the tip is in contact with the surface. There is one domain in the cantilever state, N in the folded state, and none in the unfolded state. As the surface moves away from the tip at a constant speed v, the chain becomes more extended (Fig. 2.5a), such

that

$$x_t = \sum_i x_i = vt \ . \tag{3.7}$$

The simulation assumes that the pulling takes discrete steps in space and treats x_t as constant over the duration of one time step Δt. Because of the adaptive time steps discussed in Section 3.2.3, the space steps $\Delta x_t = v\Delta t$ may have different sizes, but each step will be "small". At each step, the total extension is calculated using Eq. (3.7), and the tension $F(x_t = vt)$ is determined by numerically solving Eqs. (3.1) and (3.2) using the models Eqs. (3.3) and (3.4) for known values of the parameters in the various states $(N_u, N_f, v, \kappa_c, L_{u1}, p_u)$. When one of the molecules in the polymer unfolds (Section 3.2.2), there will be one domain in the unfolded state and $N-1$ in the folded state. In the next step, a newly balanced tension between the cantilever and the polymer is determined by solving for $F(x_t)$ as discussed above, but with the total extension x_t incremented by $v\Delta t$ and the new unfolded contour length L_{u1} and folded contour length $(N-1)L_{f1}$. The sudden lengthening of the polymer chain results in a corresponding abrupt drop in the force, leading to the formation of one sawtooth in the force curve. As the pulling continues and more domains unfold, force curves with a series of sawteeth are generated (Fig. 3.7a).

Equlibration time scales

The tension calculation assumes an equilibrated chain, so consideration must be given to the chain's relaxation time, which should be short compared to the loading time scale. The relaxation time for a WLC is given by

$$\tau \approx \eta \frac{k_B T p}{F^2} \ , \tag{3.8}$$

where η is the dynamic viscosity, F is the tension, and p is the persistence length[106]. For forces greater than 1 pN, with $\eta_{\text{water}}/k_B T = 2.45 \cdot 10^{-10}$ s/nm^3, $\tau < 2$ ns for the protein polymer used in the simulation. Therefore, the polymer chain is equilibrated almost instantaneously within a time step, which is on the order of tens of microseconds. The relaxation time of the cantilever can be determined by measuring the cantilever deflection induced by liquid motion and fitting the time dependence of the deflection to an exponential function[107]. For a 200 μm rectangular cantilever with

a bending spring constant of 20 pN/nm, the measured relaxation time in water is ~ 50 μ/s (data not shown). This relatively large relaxation time constant makes the cantilever act as a low-pass filter and also causes a lag in the force measurement.

3.2.2 Unfolding protein molecules by force

In the previous section, we discussed methods for calculating the tension of a chain composed of several domains in series (Fig. 3.1a). Those methods allow us to calculate the tension of the chain for any given extension. We use that tension to calculate transition rates between states (Fig. 3.1b). In this section, we'll introduce the Bell model for unfolding (Section 3.2.2) and mention a few more exotic models. We'll wrap up by pointing out some of the approximations and assumptions we make when we use these simple models (Section 3.2.2).

Bell model

According to the theory developed by Bell[90] and extended by Evans and Ritchie[106], an external stretching force F increases the unfolding rate constant of a protein molecule[2]

$$k_u = k_{u0} e^{\frac{F \Delta x_u}{k_B T}} \ , \tag{3.9}$$

where k_{u0} is the unfolding rate in the absence of an external force, and Δx_u is the distance between the native state and the transition state along the pulling direction.

Monte Carlo transitions

We can use the Bell model (or other models, see Section 3.2.2) to calculate the unfolding rate k_u at a given force for a single domain. The probability for that single protein domain to unfold under applied force is

$$P_1 = k_u \Delta t \ , \tag{3.10}$$

where Δt is the time duration for each pulling step, over which F is constant. This expression is accurate for $P_1 \ll 1$. From the binomial distribution, the probability of at least one of a group of

[2] Also in Hummer and Szabo[88] Eq. (1), the first paragraphs of Dudko et al.[89] and Dudko et al.[108], and many other SMFS articles.

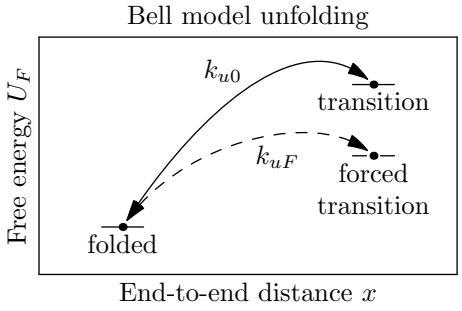


Figure 3.4: Energy landscape schematic for Bell model unfolding (Eq. (3.9)), which models folded domains as two-state systems parameterized by an unforced unfolding rate k_{u0} and a distance Δx between the folded and transition states.

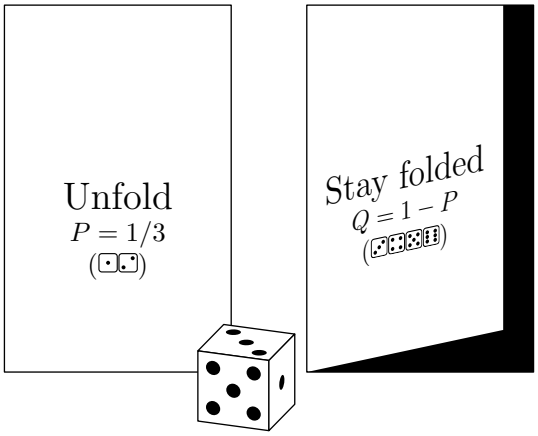


Figure 3.5: Once the unfolding probability has been caculated, we need to determine whether or not a domain should unfold. We do this by generating a random number, and comparing that number to the unfolding probability P. The random number determines which of the possible paths we should follow for the current simulation. Such "statistical sampling" is the hallmark of the Monte Carlo approach[18]. This cartoon translates the idea into the more familiar doors (possible paths) and dice (random numbers).

N_f identical domains to unfold in a given time step is

$$P = 1 - (1 - P_1)^{N_f} \approx N_f P_1 \,, \tag{3.11}$$

where the approximation is valid when $N_f P_1 \ll 1$.

To determine if an unfolding event occurs in a particular time step, the probability calculated using Eq. (3.11) is compared with a randomly generated number uniformly distributed between 0 and 1 (Fig. 3.5). If P is bigger than the random number, a domain unfolds, changing the population of each tension state, and a new balance between the polymer and the cantilever is determined. If

no unfolding event occurs the pulling continues and the unfolding probability is calculated again in the next step at a higher force. When all the molecules in the polymer have unfolded, the pulling continues until a pre-determined force level is reached, where the polymer is assumed to detach from one of the tethering surfaces. The cantilever deflection becomes zero after this point.

Other models

Although the Bell model (Eq. (3.9)) is the most widely used unfolding model due to its simplicity and its applicability to various biopolymers[80], other theoretical models have been proposed to interpret mechanical unfolding data. For example, Walton et al.[109] uses a stiffness-corrected Bell model. Schlierf and Rief[87] used the mechanical unfolding data of the protein ddFLN4 to demonstrate that Kramers' diffusion model (in the spatial-diffusion-limited case, a.k.a. the Smoluchowski limit)[86,91,110–112] fit the measured unfolding force data better than the Bell model for proteins with broad free energy barriers.

$$\frac{1}{k_u} = \frac{1}{D}\int_{-\infty}^{\infty} e^{\frac{U_F(x)}{k_B T}} \int_{-\infty}^{x} e^{\frac{-U_F(x')}{k_B T}} \mathrm{d}x'\mathrm{d}x \ , \tag{3.12}$$

where D is the diffusion coefficient and $U_F(x)$ is the free energy along the unfolding cordinate x (Fig. 3.6).

When you are simulating the double integral form of Kramers' model, you obviously need to parameterize $U_F(x)$ somehow. There has not been much research done in this direction, but Schlierf and Rief[87] used cubic splines with 15 variable knots. Shillcock and Seifert[111] used a cubic free energy with variable coefficients. The amount of information you can extract from fitting such a model to your data is limitless, but you run the risk of over-specifying if you add too many parameters when you're fitting noisy data.

There are alternative formulations of Kramers' model besides the full double integral approach. You can use a Gaussian steepest-descent approximation (a.k.a. stationary phase method or saddle-point method) to reduce the integral to a formula that only depends on the free energy landscape via the curvature $\frac{\partial^2 U_F}{\partial x^2}$ evaluated at the folded state and transition state[91]. This approach makes

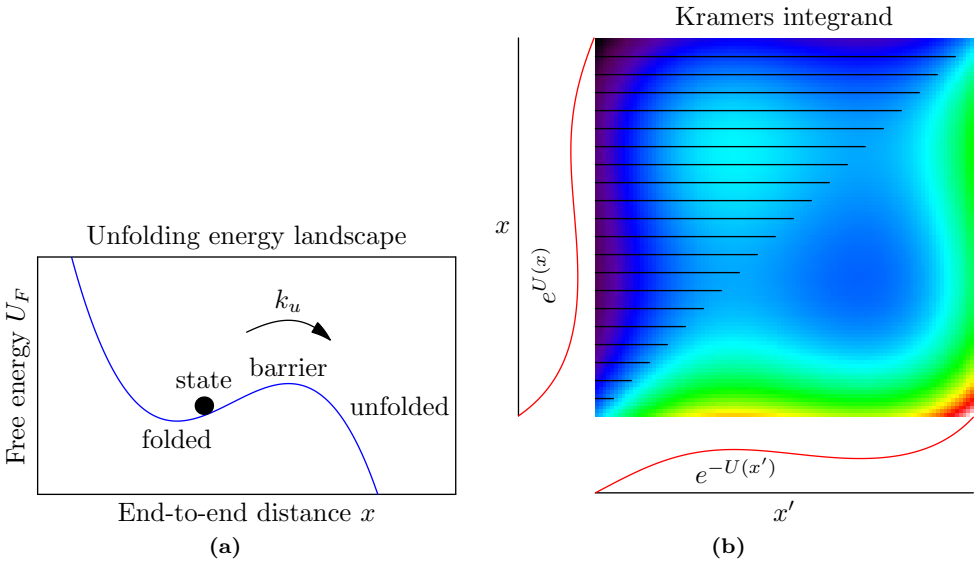


Figure 3.6: (a) Energy landscape schematic for Kramers integration (compare with Fig. 3.4). (b) A map of the magnitude of Kramers' integrand, with black lines tracing the integration region. The bulk of the contribution to the integral comes from the bump in the upper left, with x near the boundary and x' near the folded state. This is why you can calculate a close approximation to this integral by restricting the integration to $x_{\min}$ and $x_{\max}$, located a few k_BT beyond the folded and transition states respectively. The restricted integral is much easier to calculate numerically than one bound by $\pm\infty$. (Eq. (3.12)).

sense for sufficiently sharp folded and transition states, where these two measurements will capture the shape of the large-integrand region (Fig. 3.6b). The steepest-descent formulation has less to say about the underlying energy landscape, but it may be more robust in the face of noisy data.

How to choose which unfolding model to use? For proteins with relatively narrow folded and transition states, the Bell model provides a good approximation, and it is the model used by the vast majority of earlier work in the field. I will use the Bell model in my analysis of ion-dependent unfolding (Chapter 7), but analyzing my unfolding data with a different transition rate model is just a matter of changing some command line options and rerunning the sawsim simulations.

Assumptions

The interactions between different parts of the polymer and between the chain and the surface (except at the tethering points) are often ignored. This is usually reasonable since these interactions should not make substantial contributions to the force curve at the force levels of interest, where

the polymer is in a relatively extended conformation. However, Li et al.[96] showed that while the unfolding properties of I27 are not effected by I28 flankers, I28 *is* stabilized by neighboring I27. The unforced Bell model unfolding rate for I28 in $(I28)_8$ was $2.8 \cdot 10^{-5}$ s^{-1}, while in $(I27\text{-}I28)_4$ it dropped to $2.5 \cdot 10^{-6}$ s^{-1} [96].

3.2.3 Choosing the simulation time steps

The demands on the time step vary throughout a simulated pull due to the non-linear elasticity of the polymer. Within a specified time duration (or pulling distance), the force change is small at low force levels and large at high force levels. To be efficient, the simulation algorithm adapts the time step to keep the time steps large where large time steps have little effect, while shrinking the time step where smaller steps are necessary.

Within each time step, the total chain extension x_t is treated as a constant and a force balance is reached very quickly among the various domains (Section 3.2.1). This force is used to determine the unfolding probability (Eqs. (3.10) and (3.11)), which determines the domain state populations in the next time step. Therefore, the chain tension must not change appreciably over the course of the time step ($\Delta F < 1$ pN), and the unfolding probability is only calculated once for the entire step. The time step must also be short enough that the probability of unfolding in a single time step is low ($P < 10^{-3}$). Besides ensuring that the approximations made in Eqs. (3.10) and (3.11) are valid, this restriction makes time steps which should have multiple unfoldings in a single time step highly unlikely. Experimentally measured unfolding are temporally separated, because the unfolding transition is characterized by multiple, Markovian attempts over a large energy barrier, where the probability of crossing the barrier in a single attempt is very low. A successful attempt quickly extends the chain contour length, reducing the tension, dramatically reducing the likelihood of a second escape in that time step. The time step used is recalculated for each step so that both of these criteria are satisfied.

Table 3.1: (a) Model for $I27_8$ domain states and (b) transitions between them (compare with Fig. 3.1). The models and parameters are those given by Carrion-Vazquez et al.[6]. Carrion-Vazquez et al.[6] don't list their cantilever spring constant (or, presumably, use it in their simulations), but we can estimate it from the rebound slope in their Fig. 2.a and 2.b, see Fig. 3.9.

(a)

Domain states			
Domain name	Initial count	Tension model	Model parameters
AFM cantilever	1	Hooke (Eq. (3.3))	$k_c = 0.05$ N/m
Folded I27	8	WLC (Eq. (3.4))	$p = 3.9$ Å, $L = 5.1$ nm
Unfolded I27	0	WLC (Eq. (3.4))	$p = 3.9$ Å, $L = 33.8$ nm

(b)

Transition rates				
Transition	Source	Target	Rate model	Model parameters
Unfolding	Folded I27	Unfolded I27	Bell (Eq. (3.9))	$k_{u0} = 3.3 \cdot 10^{-4}$ s^{-1}, $\Delta x = 0.35$ nm.

3.3 Results and discussion

3.3.1 Force curves generated by simulation

Figure 3.7a shows three simulated force curves from pulling a polymer composed of eight identical protein molecules using parameters from typical experimental settings. The order of the peaks in the force curves reflects the temporal sequence of the unfolding events instead of the positions of the protein molecules in the polymer[96]. As observed experimentally (Fig. 2.5b), the forces at which identical protein molecules unfold fluctuate, revealing the stochastic nature of protein unfolding since no instrumental noise is included in the simulation.

After aquiring a series of experimental unfolding curves, we need to fit the data to an explanatory model. For velocity-clamp experiments (Sections 2.4 and 3.2.1), we extract unfolding forces from the sawtooth curves (Chapter 6) and generate histograms of unfolding forces. Then we construct a parameterized model of the experimental system (Table 3.1). We can then run *in silico* experiments mimicking our *in vitro* experiments (Fig. 3.7b). We extract the model parameters which provide the best fit using a "fit quality" metric and a nonlinear optimization routine (or a full parameter space sweep, for low-dimensional parameter spaces).

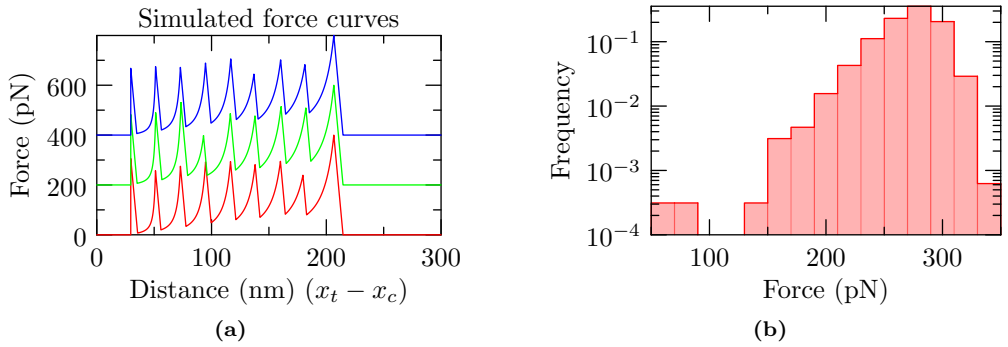


Figure 3.7: (a) Three simulated force curves from pulling a polymer of eight identical protein molecules. The simulation was carried out using the parameters: pulling speed $v = 1$ μm/s, cantilever spring constant $\kappa_c = 50$ pN/nm, temperature $T = 300$ K, persistence length of unfolded proteins $p_u = 0.40$ nm, $\Delta x_u = 0.225$ nm, and $k_{u0} = 5 \cdot 10^{-5}$ s^{-1}. The contour length between the two linking points on a protein molecule is $L_{f1} = 3.7$ nm in the folded form and $L_{u1} = 28.1$ nm in the unfolded form. These parameters are those of ubiquitin molecules connected through the N-C termini[17,19]. Detachment from the tip or substrate is assumed to occur at a force of 400 pN. In experiments, detachments have been observed to occur at a variety of forces. For clarity, the green and blue curves are offset by 200 and 400 pN respectively. (b) The distribution of the unfolding forces from 400 simulated force curves (3200 data points) such as those shown in (a). The frequency is normalized by the total number of points, *i.e.*, the height of each bin is equal to the number of data points in that bin divided by the total number of data points.

Because the unfolding behaviors of an individual sawtooth curve is stochastic (Fig. 3.7a), we cannot directly compare single curves in our fit quality metric. Instead, we gather many experimental and simulated curves, and compare the aggregate properties. For velocity-clamp experiments, the usual aggregate property used for comparison is a histogram of unfolding forces[6] (Fig. 3.7b). Defining and extracting "unfolding force" is suprisingly complicated (Section 6.3), but basically it is the highest tension force achieved by the chain before an unfolding event (the drops in the sawtooth). The final drop is not an unfolding event, it is the entire chain breaking away from the cantilever tip, severing the connection between the substrate and the cantilever.

3.3.2 The supramolecular scaffold

Analysis of the mechanical unfolding data is complicated by the dependence of the average unfolding force on the unfolding order due to the serial linkage of the molecules. Under an external stretching force F, the probability of some domain unfolding in a polymer with N_f folded domains is $N_f P_1$

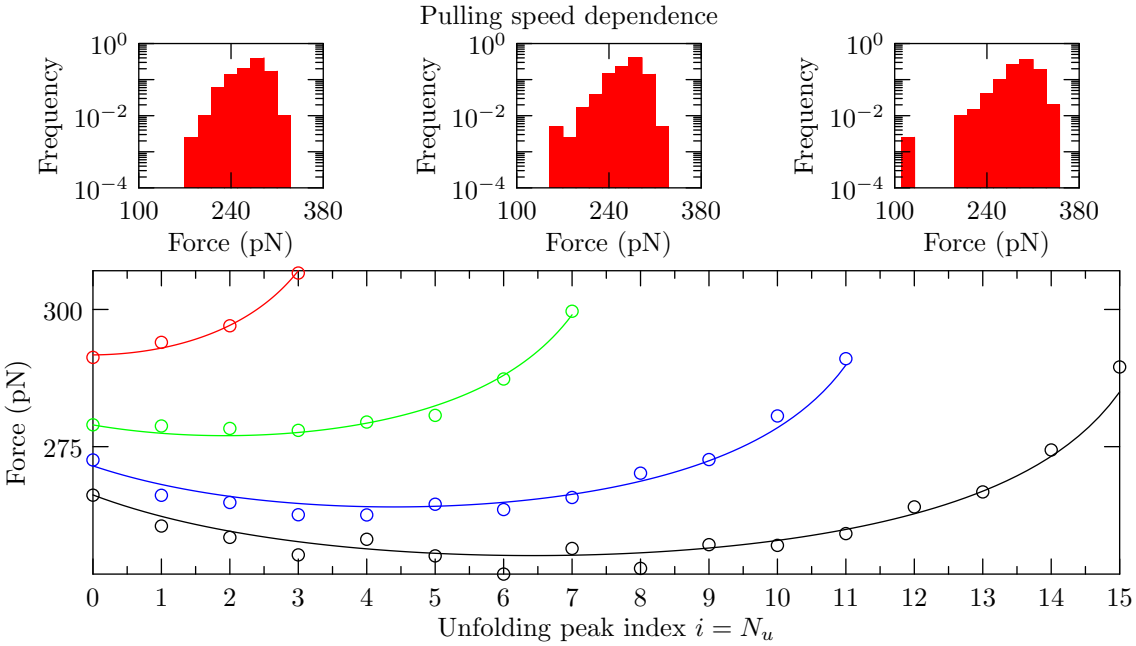


Figure 3.8: The dependence of the unfolding force on the temporal unfolding order for four polymers with 4, 8, 12, and 16 identical protein domains. Each point in the figure is the average of 400 data points. The first point in each curve represents the average of only the first peak in each of the 400 simulated force curves, the second point represents the average of only the second peak, and so on. The solid lines are fits of Eq. (3.21) to the simulated data, with best fit $\kappa_{\mathrm{WLC}} = 203$, 207, 161, and 157 pN/nm, respectively, for lengths 4 through 16. The insets show the force distributions of the first, fourth, and eighth peaks, left to right, for the polymer with eight protein domains. The parameters used for generating the data were the same as those used for Fig. 3.7a, except for the number of domains. The histogram insets were normalized in the same way as in Fig. 3.7b.

(Eq. (3.11)), which is higher than the unfolding probability for a single molecule P_1. Consequently, the average unfolding force is lower for the earlier unfolding events when N_f is larger, and the force should increase as more and more molecules become unfolded. However, there is a competing factor that opposes this trend. As the protein molecules unfold, the chain becomes softer and the force loading rate becomes lower when the pulling speed is constant. This lower loading rate leads to a decrease in the unfolding force (in the no-loading limit, all unfolding events occur at a tension of 0 N). The dependence of the average unfolding force on the unfolding order is the result of these two opposing effects. Figure 3.8 shows the dependence of the average unfolding force on the unfolding force peak order (the temporal order of unfolding events) for four polymers with 4, 8, 12, and 16 identical protein molecules. The effect of polymer chain softening dominates the initial unfolding events, and the average unfolding force decreases as more molecules unfold. After several molecules have unfolded, the softening for each additional unfolding event becomes less significant, the change in unfolding probability becomes dominant, and the unfolding force increases upon each subsequent unfolding event[82].

We validate this explanation by calculating the unfolding force probability distribution's dependence on the two competing factors. The rate of unfolding events with respect to force is

$$r_{uF} = -\frac{\mathrm{d}N_f}{\mathrm{d}F} = -\frac{\mathrm{d}N_f/\mathrm{d}t}{\mathrm{d}F/\mathrm{d}t} = \frac{N_f k_u}{\kappa v} \tag{3.13}$$

$$= \frac{N_f k_{u0}}{\kappa v} e^{\frac{F\Delta x_u}{k_B T}} = \frac{1}{\rho} e^{\frac{F-\alpha}{\rho}} , \tag{3.14}$$

where N_f is the number of folded domain, κ is the spring constant of the cantilever-polymer system, κv is the force loading rate[3], and k_u is the unfolding rate constant (Eq. (3.9)). In the last expression,

[3]

$$\frac{\mathrm{d}F}{\mathrm{d}t} = \frac{\mathrm{d}F}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}t} = \kappa v . \tag{3.15}$$

Alternatively,

$$F = \kappa x = \kappa v t \tag{3.16}$$

$$\frac{\mathrm{d}F}{\mathrm{d}t} = \kappa v . \tag{3.17}$$

See the text before Evans and Ritchie[86] Eq. (11) or Dudko et al.[89] Eq. (4) for similar explanations.

$\rho \equiv k_B T/\Delta x_u$, and $\alpha \equiv -\rho \ln(N_f k_{u0} \rho/\kappa v)$. We can approximate κ as a series of Hookean springs,

$$\kappa = \left(\frac{1}{\kappa_c} + \frac{N_u}{\kappa_{\mathrm{WLC}}} \right)^{-1} , \tag{3.18}$$

where κ_{WLC} is the effective spring constant of one unfolded domain, assumed constant for a particular polymer/cantilever combination.

The event probability density for events with an exponentially increasing likelihood function follows the Gumbel (minimum) probability density [113] with ρ and α being the scale and location parameters, respectively [88]

$$\mathcal{P}(F) = \frac{1}{\rho} e^{\frac{F-\alpha}{\rho} - e^{\frac{F-\alpha}{\rho}}} . \tag{3.19}$$

The distribution has a mode α, mean $\langle F \rangle = \alpha - \gamma_e \rho$, and a variance $\sigma^2 = \pi^2 \rho^2/6$, where $\gamma_e = 0.577\ldots$ is the Euler–Mascheroni constant [113]. Therefore, the unfolding force distribution has a variance

$$\sigma^2 = \frac{\left(\frac{\pi k_B T}{\Delta x_u}\right)^2}{6} , \tag{3.20}$$

and and average [67,88]

$$\langle F(i) \rangle = \frac{k_B T}{\Delta x_u} \left[\ln \left(\frac{\kappa v \Delta x_u}{N_f k_{u0} k_B T} \right) - \gamma_e \right] , \tag{3.21}$$

where N_f and κ depend on the domain index $i = N_u$. Curves based on this formula fit the simulated data remarkably well considering the effective WLC stiffness κ_{WLC} is the only fitted parameter, and that the actual WLC stiffness is not constant, as we have assumed here, but a non-linear function of F. Dudko et al. [114] derived a formula for the loading rate for a WLC, but as far as I know, nobody has found an analytical form for the unfolding force histograms produced under such a variable loading rate.

From Fig. 3.8, we see that the proper way to process data from mechanical unfolding experiments is to group the curves according to the length of the polymer and to perform statistical analysis separately for peaks with the same unfolding order. However, in most experiments, the tethering of the polymer to the AFM tip is by nonspecific adsorption; as a result, the polymers being stretched

between the tip and the substrate have various lengths[96]. In addition, the interactions between the tip and the surface often cause irregular features in the beginning of the force curve (Fig. 2.5b), making the identification of the first peak uncertain[64]. Furthermore, it is often difficult to acquire a large amount of data in single molecule experiments. These difficulties make the aforementioned data analysis approach unfeasible for many mechanical unfolding experiments. As a result, the values of all force peaks from polymers of different lengths are often pooled together for statistical analysis. To assess the errors caused by such pooling, simulation data were analyzed using different pooling methods and the results were compared. Figure 3.7b shows that, for a polymer with eight protein molecules, the average unfolding force is 281 pN with a standard deviation of 25 pN when all data is pooled. If only the first peaks in the force curves are analyzed, the average force is 279 pN with a standard deviation of 22 pN. While for the fourth and eighth peaks, the average force are 275 pN and 300 pN, respectively, and the standard deviations are 23 pN and 25 pN, respectively. As expected from the Gumbel distribution, the width of the unfolding force distribution (insets in Fig. 3.8) is only weakly effected by unfolding order, but the average unfolding force can be quite different for the same protein because of the differences in unfolding order and polymer length.

Benedetti et al.[115] have since proposed an alternative parameterization for Eq. (3.18), using

$$\kappa = \left(\frac{1}{\kappa_c} + \frac{N_f}{\kappa_f} + \frac{N_u}{\kappa_u} \right)^{-1} \equiv \frac{\kappa'}{1 - AN_f} \,, \tag{3.22}$$

where κ' is the spring constant of the completely unfolded chain and A is a correction term for the supramolecular scaffold. This is effectively a first order Taylor expansion for κ^{-1} about $N_f = 0$, but the remaining analysis is identical.

$$f(N_f) \equiv \kappa^{-1} = \frac{1}{\kappa_c} + \frac{N_f}{\kappa_f} + \frac{N - N_f}{\kappa_u} \tag{3.23}$$

$$= f(0) + \left. \frac{\mathrm{d}f}{\mathrm{d}N_f} \right|_{N_f = 0} N_f + \mathcal{O}\left(N_f^2\right) \tag{3.24}$$

$$\approx \left(\frac{1}{\kappa_c} + \frac{N}{\kappa_u} \right) + \left(\frac{1}{\kappa_f} - \frac{1}{\kappa_u} \right) N_f \tag{3.25}$$

In the case where the wormlike chain stiffnesses κ_f and κ_u are fairly constant over the unfolding region, there are no higher order terms and the first order expansion in Eq. (3.25) is exact. Comparing Eqs. (3.22) and (3.25), we see

$$\kappa' = \frac{1}{\kappa_c} + \frac{N}{\kappa_u} \tag{3.26}$$

$$-\kappa' A = \frac{1}{\kappa_f} - \frac{1}{\kappa_u} \tag{3.27}$$

$$A = \frac{\frac{1}{\kappa_u} - \frac{1}{\kappa_f}}{\frac{1}{\kappa_c} + \frac{N}{\kappa_u}} \tag{3.28}$$

By focusing on the $A = 0$ case (*i.e.* $\kappa_f = \kappa_u$), Benedetti et al.[115] avoid running Monte Carlo simulations when modeling unfolding histograms. This simplification does not hold for our simulated data (Fig. 3.8), but for some experimental analysis the loss of accuracy may be acceptable in return for the reduced computational cost.

3.3.3 The effect of cantilever force constant

In mechanical unfolding experiments, the ability to observe the unfolding of a single protein molecule depends on the tension drop after an unfolding event such that another molecule does not unfold immediately. The magnitude of this drop is determined by many factors, including the magnitude of the unfolding force, the contour and persistence lengths of the protein polymer, the contour length increase from unfolding, and the stiffness (force constant) of the cantilever. Among these, the effect of the cantilever force constant is particularly interesting because cantilevers with a wide range of force constants are available. In addition, different single molecule manipulation techniques, such as the AFM and laser tweezers, differ mainly in the range of the spring constants of their force transducers[109]. Figure 3.9 shows the simulated force curves from pulling an octamer of protein molecules using cantilevers with different force constants, while other parameters are identical. For this model protein, the appearance of the force curve does not change much until the force constant of the cantilever reaches a certain value ($\kappa_c = 50$ pN/nm). When κ_c is lower than this value, the individual unfolding events become less identifiable. In order to observe individual unfolding events, the cantilever needs to have a force constant high enough so that the bending at the maximum force

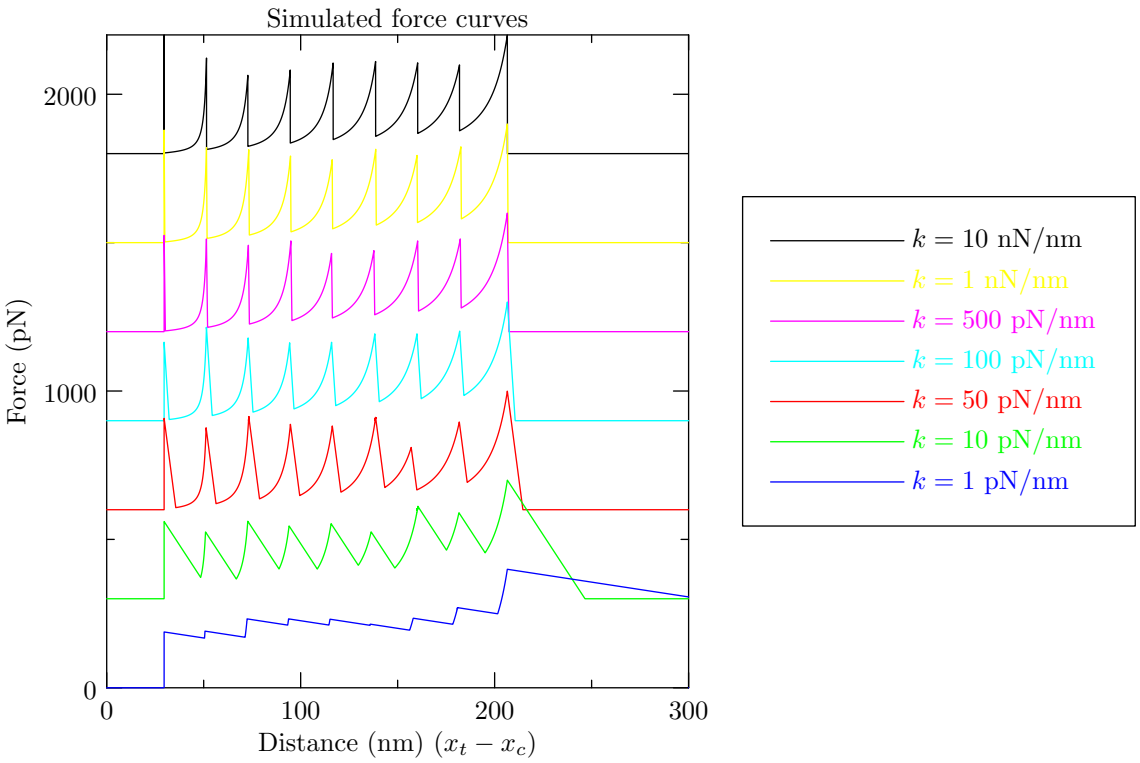


Figure 3.9: Simulated force curves obtained from pulling a polymer with eight protein molecules using cantilevers with different force constants κ_c. Parameters used in generating these curves are the same as those used in Fig. 3.7, except the cantilever force constant. Successive force curves are offset by 300 pN for clarity.

is small in comparison with the contour length increment from the unfolding of a single molecule. Figure 3.9 also shows that the back side of the force peaks becomes more tilted as the cantilever becomes softer. This is due to the fact that the extension (end-to-end distance) of the protein polymer has a large sudden increase as the tension rebalances after an unfolding event.

It should also be mentioned that the contour length increment from each unfolding event is not equal to the distance between adjacent peaks in the force curve because the chain is never fully stretched. This contour length increase can only be obtained by fitting the curve to WLC or other polymer models (Fig. 2.5b).

3.3.4 Determination of Δx_u and k_{u0}

As mentioned in Section 3.3.1, fitting experimental unfolding force histograms to simulated histograms allows you to extract best-fit parameters for your simulation model. For example, if you

have Bell model unfolding (Section 3.2.2), your two fitting parameters are the zero-force unfolding rate k_{u0} and the distance Δx_u from the native state to the transition state. Fig. 3.10a shows the dependence of the unfolding force on the pulling speed for different values of k_{u0} and Δx_u. As expected, the unfolding force increases linearly with the pulling speed in the linear-log plot [106]. While the magnitude of the unfolding forces is affected by both k_{u0} and Δx_u, the slope of speed dependence is primarily determined by Δx_u (Eq. (3.21)). Figure 3.10b shows that the width of the unfolding force distribution is very sensitive to Δx_u, as expected from the Gumbel distribution (Eq. (3.20)). To obtain the values of k_{u0} and Δx_u for the protein, the pulling speed dependence and the distribution of the unfolding forces from simulation, such as those shown in Fig. 3.10a and the insets of Fig. 3.10b, are compared with the experimentally measured results. The values of k_{u0} and Δx_u that provide the best match are designated as the parameters describing the protein under study. Since k_{u0} and Δx_u affect the unfolding forces differently, the values of both parameters can be determined simultaneously. The data used in plotting Fig. 3.10 includes all force peaks from the simulated force curves because most experimental data is analyzed that way.

In most published literature, k_{u0} and Δx_u were fit by carrying out simulations using a handful of possible unfolding parameters and selected the best fit by eye. This approach does not allow estimation of uncertainties in the fitting parameters, as shown by Best et al. [81]. A more rigorous approach involves quantifying the quality of fit between the experimental and simulated force distributions, allowing the use of a numerical minimization algorithm to pick the best fit parameters. We use the Jensen–Shannon divergence [116,117], a measure of the similarity between two probability distributions.

$$D_{\mathrm{JS}}(p_e, p_s) = D_{\mathrm{KL}}(p_e, p_m) + D_{\mathrm{KL}}(p_s, p_m) \ , \tag{3.29}$$

where $p_e(i)$ and $p_s(i)$ are the the values of the i^{th} bin in the experimental and simulated unfolding force histograms, respectively. D_{KL} is the Kullback–Leibler divergence

$$D_{\mathrm{KL}}(p_p, p_q) = \sum_i p_p(i) \log_2 \left(\frac{p_p(i)}{p_q(i)} \right) \ , \tag{3.30}$$

where the sum is over all unfolding force histogram bins. p_m is the symmetrized probability distribution

$$p_m(i) \equiv [p_e(i) + p_s(i)]/2 \; . \tag{3.31}$$

The major advantage of the Jensen–Shannon divergence is that D_{JS} is bounded ($0 \leq D_{\mathrm{JS}} \leq 1$) regardless of the experimental and simulated histograms. For comparison, Pearson's χ^2 test[118],

$$D_{\chi^2} = \sum_i \frac{(p_e(i) - p_s(i))^2}{p_s(i)} \; , \tag{3.32}$$

is infinite if there is a bin for which $p_e(i) > 0$ but $p_s(i) = 0$.

Figure 3.11 shows the Jensen–Shannon divergence calculated using Eq. (3.29) between an experimental data set and simulation results obtained using a range of values of k_{u0} and Δx_u. There is an order of magnitude range of k_{u0} that produce reasonable fits to experimental data (Fig. 3.11), which is consistent with the results Best et al.[81] obtained using a chi-square test. The values of k_{u0} and Δx_u can be determined to higher precision by using both the pulling speed dependent data and the unfolding force distribution, as well as any relevant information about the protein from other sources.

3.3.5 Features

sawsim is a great improvement over existing work in this field. Best et al.[81] are the only authors to mention such automatic simulation comparisons, and their χ^2 fit only compares mean unfolding forces over a range of speeds. They calculate $\langle F \rangle$ through an iterative method, and assume a standard deviation of 20 pN on their simulated $\langle F \rangle$. sawsim, by comparison, makes full use of your experimental histograms, which you specify in a plain-text histogram file:

```
#HISTOGRAM: -v 6e-7
#Force (N)      Unfolding events
1.4e-10 1
1.5e-10 0
...
3e-10   116
3.1e-10 18
```

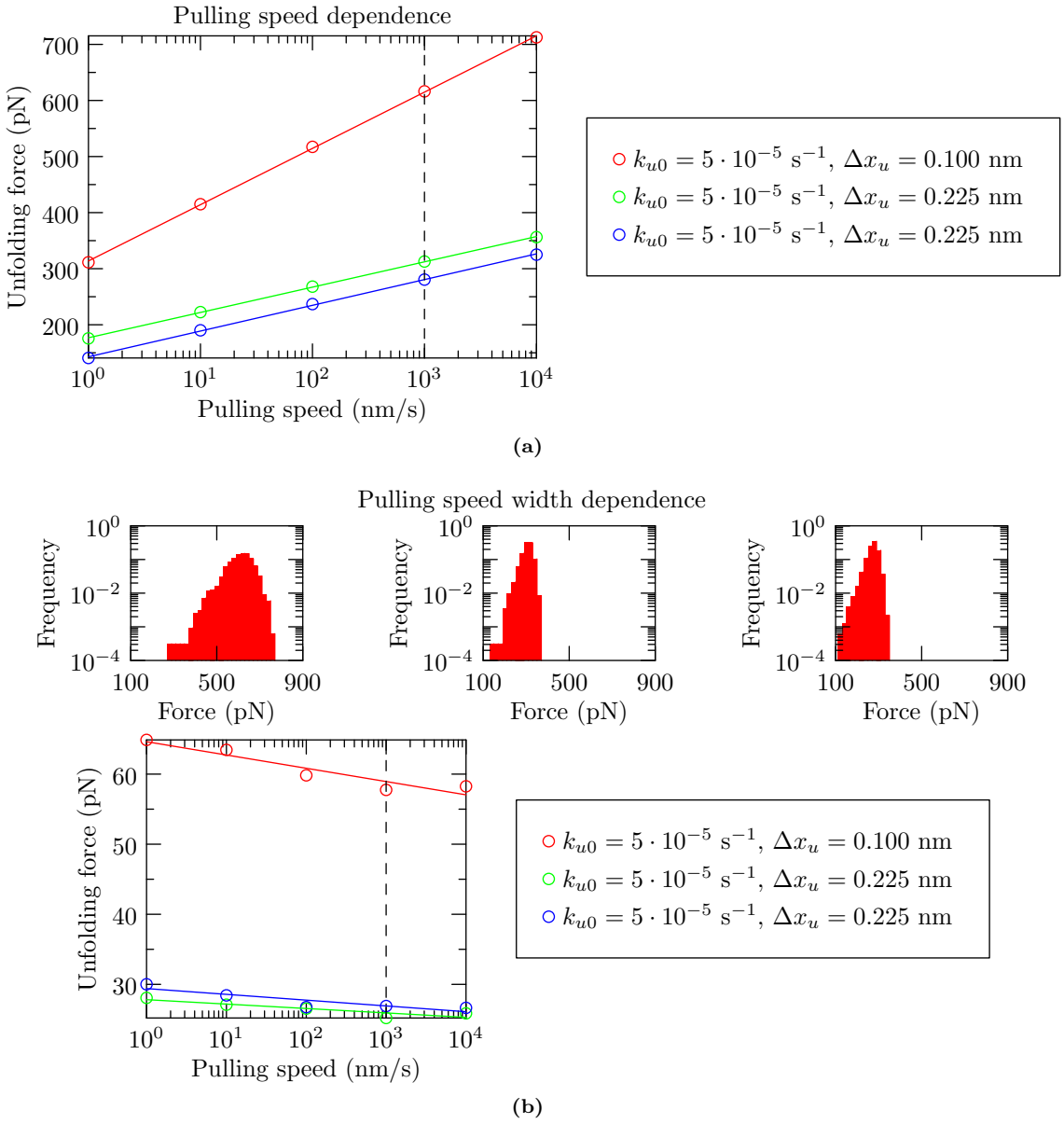


Figure 3.10: (a) The dependence of the unfolding forces on the pulling speed for three different model protein molecules characterized by the parameters k_{u0} and Δx_u. The polymer length is eight molecules, and each symbol is the average of 3200 data points. (b) The dependence of standard deviation of the unfolding force distribution on the pulling speed for the simulation data shown in (a), using the same symbols. The insets show the force distribution histograms for the three proteins at the pulling speed of 1 μm/s. The left, middle and right histograms are for the proteins represented by the top, middle, and bottom lines in (a), respectively.

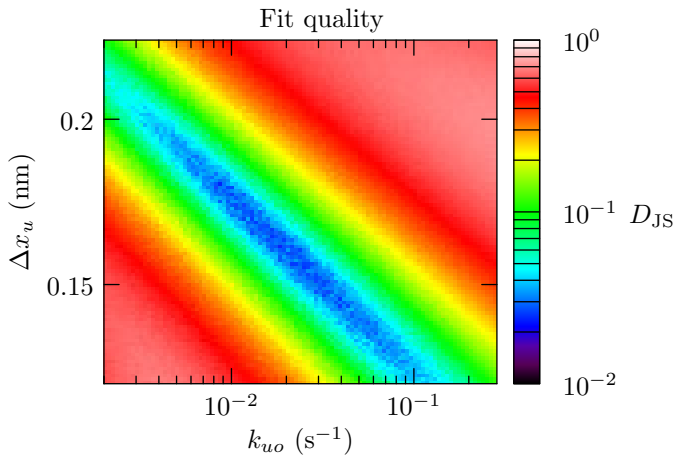


Figure 3.11: Fit quality between an experimental data set and simulated data sets obtained using various values of unfolding rate parameters k_{u0} and Δx_u. The experimental data are from octameric ubiquitin pulled at 1 μm/s[17], and the other model parameters are the same as those in Fig. 3.7. The best fit parameters are $\Delta x_u = 0.17$ nm and $k_{u0} = 1.2 \cdot 10^{-2}$ s^{-1}. The simulation histograms were built from 400 pulls at for each parameter pair.

```
3.2e-10 1

#HISTOGRAM: -v 8e-7
#Force (N)      Unfolding events
1.4e-10 0
1.5e-10 3
...
3.2e-10 50
3.3e-10 13

#HISTOGRAM: -v 1e-6
#Force (N)      Unfolding events
1.5e-10 2
1.6e-10 3
...
3.3e-10 24
3.4e-10 2
```

Each sawsim run simulates a single sawtooth curve, so you need to run many sawsim instances to generate your simulated histograms. To automate this task, sawsim comes with a Python wrapping library (`pysawsim`), which provides convenient programmatic and command line interfaces for generating and manipulating sawsim runs. For example, to compare the experimental histograms listed above with simulated data over a 50-by-50 grid of k_{u0} and Δx, you would use something like

```
$ sawsim_hist_scan.py -f '-s cantilever,hooke,0.05 -N1 -s folded,null -N8
>   -s "unfolded,wlc,{0.39e-9,28e-9}" -k "folded,unfolded,bell,{%g,x%g}"
>   -q folded' -r '[1e-5,1e-3,50],[0.1e-9,1e-9,50]' --logx histograms.txt
```

That's a bit of a mouthful, so let's break it down. Without the sawsim template (`-f ...`), we can focus on the comparison options:

```
$ sawsim_hist_scan.py ... -r '[1e-5,1e-3,50],[0.1e-9,1e-9,50]' --logx histograms.txt
```

This sets up a two-parameter sweep, with the first parameter going from $1 \cdot 10^{-5}$ to $1 \cdot 10^{-3}$ in 50 logarithmic steps, and the second going from $0.1 \cdot 10^{-9}$ to $1 \cdot 10^{-9}$ in 50 linear steps. The sawsim template defines the simulation model (Fig. 3.1 and Table 3.1), and `%g` marks the location where the swept parameters will be inserted.

Behind the scenes, `pysawsim` is spawning several concurrent sawsim processes to take advantage of any parallel processing facilities you may have access to (e.g. multiple cores, MPI, PBS, ...). A 50-by-50 grid with 400 runs per pixel at about one second per sawsim pull would take arount 12 days of serial execution. Moving the simulation to the departments' 16 core file server cuts that execution time down to 18 hours, which will easily complete over a quiet weekend. Using MPI on the departments' 15 box, dual core computer lab, the simulation would finish overnight.

3.3.6 Testing

Once a body of code reaches a certain level of complication, it becomes difficult to convince others (or yourself) that it's actually working correctly. In order to test sawsim, I've developed a test suite (distributed with sawsim) that compares simulated unfolding force histograms with analytical histograms for a number of situations where solving for the analytical histogram is possible. In the following subsection, I'll work out the theoretical unfolding force distribution for a number of tractable cases. The sawsim test suite generates simulated unfolding curves for these tractable cases (e.g. single domain Bell model unfolding with a constant loading rate), and compares the simulated unfolding force histograms with the expected theoretical distribution. The simulated histograms match the theoretical distributions for each combination of models regardless of the parameters you

feed into the models, so we can be confident that sawsim correctly implements at those models.

The instantaneous likelyhood of a protein unfolding is given by $\frac{\mathrm{d}N_u}{\mathrm{d}F}$, and the unfolding histogram is merely this function discretized over a bin of width W[4].

$$h(F) \equiv \frac{\mathrm{d}F}{\mathrm{dbin}} = \frac{\mathrm{d}N_u}{\mathrm{d}F} \cdot \frac{\mathrm{d}F}{\mathrm{dbin}} = W\frac{\mathrm{d}N_u}{\mathrm{d}F} = -W\frac{\mathrm{d}N_f}{\mathrm{d}F} = -W\frac{\mathrm{d}N_f}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}F} = \frac{W}{\kappa v}N_f k_u \tag{3.33}$$

Solving for theoretical histograms is merely a question of taking your chosen k_u, solving for $N_f(F)$, and plugging into Eq. (3.33). We can also make a bit of progress solving for N_f in terms of k_u as follows:

$$k_u \equiv -\frac{1}{N_f}\frac{\mathrm{d}N_f}{\mathrm{d}t} \tag{3.34}$$

$$-k_u \mathrm{d}t \cdot \frac{\mathrm{d}F}{\mathrm{d}t} = \frac{\mathrm{d}N_f}{N_f} \tag{3.35}$$

$$\frac{-1}{\kappa v}\int_0^F k_0(F')\mathrm{d}F' = \ln(N_f(F'))|_0^F = \ln\left(\frac{N_f(F)}{N_f(0)}\right) = \ln\left(\frac{N_f(F)}{N}\right) \tag{3.36}$$

$$N_f(F) = Ne^{\frac{-1}{\kappa v}\int_0^F k_u(F')\mathrm{d}F'}, \tag{3.37}$$

where $N_f(0) = N$ because all the domains are initially folded.

Constant unfolding rate

In the extremely weak tension regime, the protein's unfolding rate is independent of tension, so we can simplify Eq. (3.37) and plug into Eq. (3.33).

$$N_f = Ne^{\frac{-1}{\kappa v}\int_0^F k_u(F')\mathrm{d}F'} = Ne^{\frac{-k_{u0}}{\kappa v}\int_0^F \mathrm{d}F'} = Ne^{\frac{-k_{u0}F}{\kappa v}} \tag{3.38}$$

$$h(F) = \frac{W}{\kappa v}N_f k_u = \frac{Wk_{u0}N}{\kappa v}e^{\frac{-k_{u0}F}{\kappa v}}. \tag{3.39}$$

A constant unfolding-rate/hazard-function gives exponential decay. This is not an earth shattering result, but it's a comforting first step, and it does show explicitly the dependence in terms of the various unfolding-specific parameters.

[4] This is similar to Dudko et al.[89] Eq. (2), remembering that $\dot{F} = \kappa v$, that their probability density is not a histogram ($W = 1$), and that their probability density function is normalized to $N = 1$

Bell model

Stepping up the intensity a bit, we come to Bell's model for unfolding (Section 3.2.2). We can simplify the following calculation by parametrizing with the characteristic force ρ defined in Section 3.3.2 and the similar single-domain mode $\alpha' \equiv -\rho \ln(k_{u0}\rho/\kappa v)$. With these substitutions, Eq. (3.9) becomes

$$k_u = k_{u0} e^{\frac{F}{\rho}} \ . \tag{3.40}$$

The unfolding histogram is then given via Eqs. (3.33) and (3.37).

$$N_f = N e^{\frac{-1}{\kappa v}\int_0^F k_u \mathrm{d}F'} = N e^{\frac{-1}{\kappa v}\int_0^F k_{u0} e^{\frac{F'}{\rho}} \mathrm{d}F'} = N e^{\frac{-k_{u0}}{\kappa v}\int_0^F e^{\frac{F'}{\rho}} \mathrm{d}F'} = N e^{\frac{-k_{u0}\rho}{\kappa v}\left(e^{\frac{F}{\rho}}-1\right)} \tag{3.41}$$

$$= N e^{\frac{k_{u0}\rho}{\kappa v}\left(1-e^{\frac{F}{\rho}}\right)} = N e^{e^{\frac{-\alpha'}{\rho}}\left(1-e^{\frac{F}{\rho}}\right)} = N e^{e^{\frac{-\alpha'}{\rho}} - e^{\frac{F-\alpha'}{\rho}}} \tag{3.42}$$

$$h(F) = \frac{W}{\kappa v} N_f k_u = \frac{W}{\kappa v} N e^{e^{\frac{-\alpha'}{\rho}} - e^{\frac{F-\alpha'}{\rho}}} k_{u0} e^{\frac{F}{\rho}} = \frac{WNk_{u0}}{\kappa v} e^{\frac{F}{\rho} - e^{\frac{F-\alpha'}{\rho}} + e^{\frac{-\alpha'}{\rho}}} \tag{3.43}$$

$$= \frac{WN}{\rho} e^{\frac{F-\alpha'}{\rho} - e^{\frac{F-\alpha'}{\rho}} + e^{\frac{-\alpha'}{\rho}}} = \frac{WN}{\rho} e^{\frac{F-\alpha'}{\rho} - e^{\frac{F-\alpha'}{\rho}}} e^{e^{\frac{-\alpha'}{\rho}}} \tag{3.44}$$

$$= \frac{WN e^{e^{\frac{-\alpha'}{\rho}}}}{\rho} e^{\frac{F-\alpha'}{\rho} - e^{\frac{F-\alpha'}{\rho}}} \tag{3.45}$$

which matches Eq. (3.19) except for a constant prefactor due to the range[5].

Saddle-point Kramers' model

For the saddle-point approximation for Kramers' model for unfolding (Evans and Ritchie [86] Eqn. 3, Hänggi et al. [91] Eqn. 4.56c, van Kampen [112] Eqn. XIII.2.2).

$$k_u = \frac{D}{l_b l_{ts}} \cdot e^{\frac{-U_b(F)}{k_B T}} \ , \tag{3.46}$$

where $U_b(F)$ is the barrier height under an external force F, D is the diffusion constant of the protein conformation along the reaction coordinate, l_b is the characteristic length of the bound state $l_b \equiv 1/\rho_b$, ρ_b is the density of states in the bound state, and l_{ts} is the characteristic length of the

[5] The Gumbel distribution in Eq. (3.19) is normalized for the range $-\infty < F < \infty$, but Eq. (3.45) is normalized for the range $0 \leq F < \infty$. This distinction will alter the analytical mean and variance listed after Eq. (3.19), but with the experimental unfolding histograms showing few zero-force unfolding events, the effective difference will be negligible.

transition state.

Evans and Ritchie [86] solved this unfolding rate for both inverse power law potentials and cusp potentials.

3.4 Conclusions

We have described the method of performing Monte Carlo simulations based on a simple two-state model for the mechanical unfolding of protein molecules and discussed the complications involved in the simulation procedure. Besides its use in this thesis, sawsim analysis has been used in Roman [119] Fig. 75. In addition to the extraction of kinetic properties of the protein from mechanical unfolding data, such simulations can help to elucidate the effects of various experimental parameters on the appearance of force curves and to estimate the errors associated with data pooling. To date, the force-induced unfolding approach has been used to investigate several different types of proteins. As the technique is used to study a wider range of proteins, this simple simulation method will be useful for data analysis, experimental design, and artifact identification.

Chapter 4: Experiment control software, pyafm, and related packages

> Civilization advances by extending the number of important operations which we can perform without thinking about them. Operations of thought are like cavalry charges in a battle—they are strictly limited in number, they require fresh horses, and must only be made at decisive moments.
>
> Alfred North Whitehead [120]

Velocity clamp experiments have been carried out since the initial work by Rief et al. [30], so I was somewhat surprised that there weren't already community-driven packages for carrying out and analyzing these experiments [121–124]. When I joined Prof. Yang's lab, we were using experiment control software written in LabVIEW and analysis software written in IGOR Pro, both developed in-house. The existing software was not designed to control sample temperature or for easy extension, so I proceeded to write my own control and analysis stack to add these capabilities.

For those of you thinking, "Why is he calling this thing a stack?", software is rarely developed as a single monolithic program. Instead, developers write software as a series of modular components, with each layer in the stack using lower level features from the layers below it to supply higher level features to the layers above it. New high-level programs will contain logic for the new idea (perform velocity-clamp unfolding experiments) and leverage pre-existing packages for all the old ideas that you need to get the job done (open a file, Fourier transform an array, ...). A well structured suite of software breaks up the task at hand into many sub-components, with a distinct package handling each component.

Whitehead [120] introduces his claim about civilization and subconscious operations to motivate the utility of symbolism in subconcious reasoning. By encapsulating already established ideas in a compact form, we can focus on the crux of an issue without being distracted by the peripheral boilerplate.

In this chapter, I will discuss the earlier frameworks and abortive attempts that lead me towards

my current architecture (Sections 4.1 and 4.2). I will also discuss some auxiliary packages I developed to support the main stack (Section 4.3). I'll wrap up by comparing my stack with Prof. Yang's earlier framework and summarizing lessons I've learned along the way (Sections 4.4 and 4.5).

4.1 Analog input output frameworks

For many users, computers are fairly self-contained systems. Users read and write files, using a variety of editors, and share information between computers via networked connections. Using computers to control and monitor arbitrary physical processes is common in the scientific community and industry, but less so in the general consumer market. This means that interfaces between the digital and analog worlds haven't seen the focused development in the open source community that more mainstream problem areas have received. In this section I'll discuss a few possible options—both open and proprietary—in the context of my experiment control stack.

4.1.1 LabVIEW

National Instruments[127] is a major player in the experiment control and data aquisition market. On the hardware side, they produce a wide range of DAQ cards. On the software side, they produce LabVIEW[125], a graphical programming language designed to make writing control and aquisition experiments straightforward. Both LabVIEW and NI-DAQmx cards are ubiquitous in scientific computing; in the four research labs I've worked in over my career, every lab has used both. By the time I joined Prof. Yang's lab, I'd been using LabVIEW for years, and had become familiar with its two major limitations: name based linking and a binary file format.

Programming in a graphical language is quite similar to programming in a textual language. In both, you reduce complexity by encapsulating functional subroutines of your process, and then assembling those subroutines in other, higher-level subroutines[128–131]. This means that the application level code can focus on application-level task (approach the surface, wait for binding, ...) without getting bogged down in the details (increment analog output channel zero in 5 bit steps until analog input channel exceeds 39322 bits). In textual languages like C or Python, you can use functions and libraries to package the functional subroutines. In LabVIEW, you package the

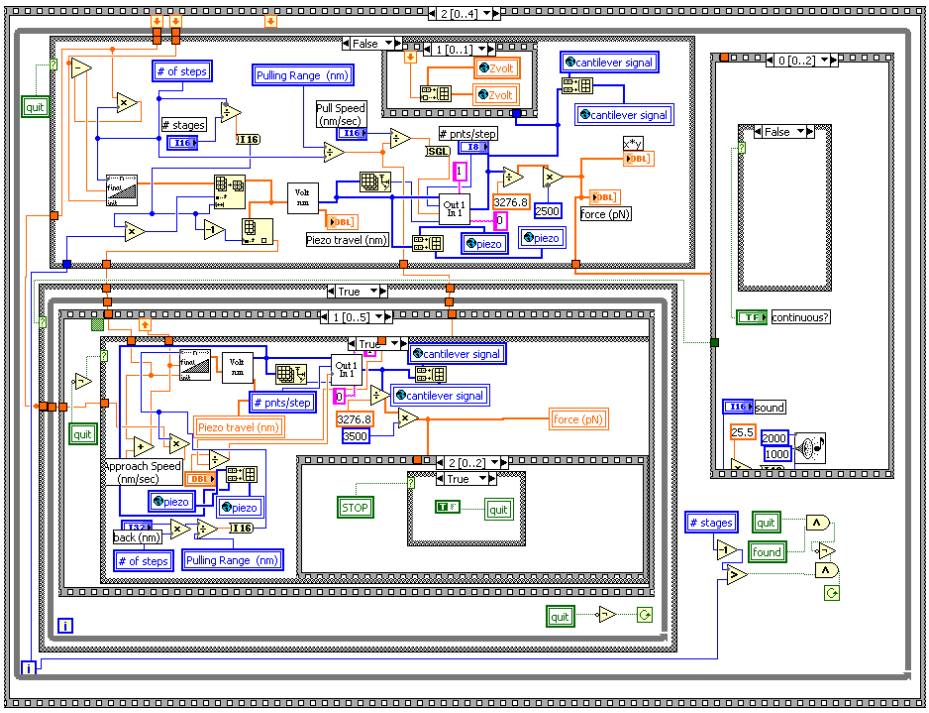


Figure 4.1: An excerpt from the main frame of the LabVIEW stack. This frame codes for the velocity-clamped pull phase of a push–bind–pull experiment.

subroutines in *virtual instruments* (VIs).

The problem comes when you want to update one of your subroutines. LabVIEW VIs are linked dynamically by VI name[132], so there was no easy way to swap a new version of the VI into your application for testing without renaming the subroutine. With the Project Explorer (new in LabVIEW 8.0[132], released 2005), these renames became easier. However, throughout my time in the Yang lab, the Windows machines all ran LabVIEW 7.1 (released in 2004).

Because of difficulties with name-based VI linking and the relative inexperience of many scientists in the maintenance benefits of modular programming[133,134], LabVIEW code often ends up without a clean separation between high-level and low-level tasks (Fig. 4.1). This lack of structure makes it difficult to reuse existing code to address similar tasks.

The second obstacle to maintaining LabVIEW code is the binary file format for VIs. The established method for recording software history is to use a version control system (VCS), which records versions of the project in a repository. Each change to the project is committed to the

repository with some associated metadata (timestamp, committer name, explanatory message, ...). Users can access this database to recover earlier versions of the project. For example, if you find a bug in your package, you can use your VCS to determine if that bug affected the data you gathered six months ago.

There are a number of open source version control systems in common use (Git, Mercurial, and Subversion, ...), but in order to track and merge *changes*, they need a way to calculate the difference between two versions of a given file. For textual programming languages, the line-based textual differences used by VCSs work extremely well, but for binary file formats, performance decreases drastically. There are third-party merge tools[135] for LabVIEW, but the tools are not officially supported.

While National Instruments seems to put a reasonable amount of effort into maintaining backwards compatibility, long term archival of binary formats is still a difficult problem. For example, our legacy LabVIEW 7.1 installation is no longer compatible with recent LabVIEW releases. Support for the releases is so low, that without access to the old LabVIEW release, you may not even be able to determine which version of LabVIEW your VI corresponds to. One officially suggested method for extracting the version from an older VI is[136]:

> Open the VI in the earliest version on your computer. If an error occurs, the VI is saved in a later version. Close the VI and repeat this process for the next version of LabVIEW. The first version that opens a VI without any error is the version in which the VI is compiled.

This does not inspire confidence in an ability to extract experiment control software from VIs after a decade of archival[137].

4.1.2 NI-DAQmx

After deciding to avoid LabVIEW, my first attempt at writing an experiment control framework involved calling National Instrument's DAQmx library from C[138] (Fig. 4.2). I spent most of 2007 working this framework, using Cygwin as the development environment. Inspired by EPICS, I built

```
static int set_digital_output_data(DIGITAL_OUTPUT *d, unsigned int data)
{
  d->data = (uInt32) data;
  DAQmxErrChk_struct( Write_WriteDigPort(d->taskHandle, d->data) );
 Error:
  if (d->error != 0) {
    CHK( close_digital_output(d) );
    M_EXIT(FAILURE, "Error in NIDAQ stepper output\n");
  }
  CHK( nsleep(100) );
  PING(1);
  return SUCCESS;
}
```

Figure 4.2: An excerpt from the digital output module of my experiment server stack. Most of the C code is error checking and tracing macros. The hardcoded delay time and stepper-specific error message are symptoms of my previously poor programming practices. `Write_WriteDigPort` is a simplifying wrapper around `DAQmxWriteDigitalU32` from the examples bundled with NI-DAQmx.

a message passing server with experiment control and hardware interface modules connected via sockets.

As the experiment server evolved, I started running into problems. The overhead of sending all the data through sockets to generic hardware interface modules was larger than I had naïvely expected. I also had trouble with multithreaded socket code on Cygwin, and decided to drop Microsoft Windows altogether in favor of an open source operating system.

4.1.3 Comedi

After transitioning to Linux-based systems, I could no longer use NI-DAQmx (which only supported Microsoft Windows). Luckily, the Comedi project already provided open source driver code for our DAQ card (an NI-PCI-6052E). Comedi (from "Control and Measurement Device Interface") is a general purpose library for interacting with DAQ devices, and supports a wide range of hardware. When I moved to Comedi, it was a stand-alone kernel module, but since November 2008 it has been included in the Linux source as a staging driver.

Comedi development goes back to 2000, so by the time I arrived things were already pretty stable. I submitted a small patch to support simultaneous analog input/output triggering on National

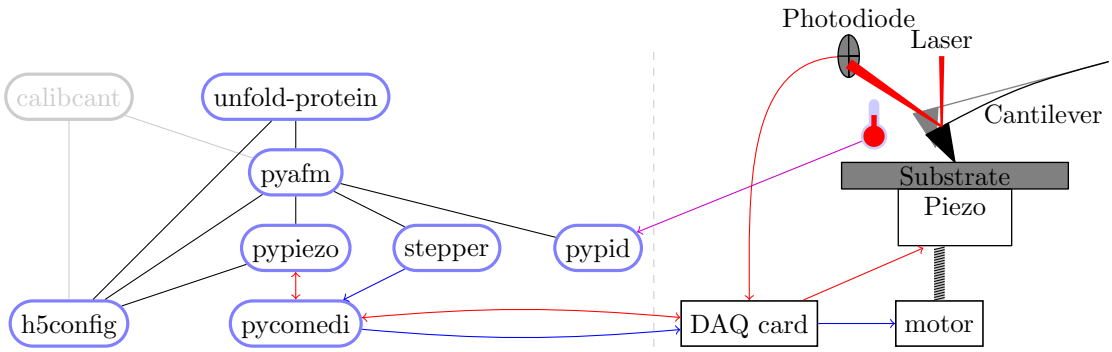


Figure 4.3: Dependency graph for my modular experiment control stack. The unfold-protein package controls the experiment, but the same stack is used by calibcant for cantilever calibration (Fig. 5.1). The dashed line ( ) separates the software components (on the left) from their associated hardware (on the right). The data flow between components is shown with arrows. For example, the stepper package calls pycomedi, which talks to the DAQ card, to write digital output that controls the stepper motor (, Section 4.3.2). The pypiezo package, on the other hand, uses two-way communication with the DAQ card ( ), writing driving voltages to position the piezo and recording photodiode voltages to monitor the cantilever deflection (Section 4.2.2). The pypid package measures the buffer temperature using a thermocouple inserted in the fluid cell (, Section 4.3.3). I represent the thermocouple with a thermometer icon ( ), because I expect it is more recognizable than a more realistic .

Instruments cards, and started building my stack.

4.2 The pyafm stack

In order to reduce future maintenance costs, I have based my stack as much as possible on existing open source software, and split my stack into reusable components where such components might appeal to a wider audience. From the bottom up, pycomedi wraps the Comedi device driver for generic input/output, pypiezo builds generic piezo-control logic on top of pycomedi, pyafm combines a pypiezo-controlled piezo with a stepper-controlled stepper motor and pypid-controlled temperature controller, and unfold-protein adds experiment logic to pyafm to carry out velocity-clamp force spectroscopy (Fig. 4.3).

4.2.1 Pycomedi

After my experience with C (Section 4.1.2), I knew I wanted a higher level language for the bulk of my experiments. Comedi already had SWIG-generated Python bindings, so I set to work creating pycomedi, an object-oriented interface around the SWIG bindings. The first generation pycomedi

interface was much easier to use than the raw SWIG bindings, especially for simultaneous analog input/output, which I needed to monitor cantilever deflection during piezo-sweeping velocity-clamp pulls.

The SWIG-based interface to Comedi provided a solid base for my experiment control stack, but as the stack matured, I started bumping up against problems due to both my poor design choices[1] and general awkwardness with the thin SWIG bindings. In 2011 I ripped out most of this layer and used Cython to bind directly to Comedi's userspace library. This lead to a much more Pythonic interface, and removed a number of previously sticky workarounds required by earlier versions of pycomedi.

As a generic Python interface to Comedi, pycomedi has a wider user base than the rest of my experiment control stack (there are more folks writing Python code for DAQ cards on Linux than there are writing velocity-clamp AFM controllers on Linux). I've had a number of people contact me directly with pycomedi questions, including a neuroscientist, a radiologist, and an automotive electrician. Éric Piel even contributed a few patches for software-calibrated devices.

Comparing the NI-DAQmx implementation of digital writes (Fig. 4.2) with a more complete pycomedi implementation (Fig. 4.4), the pycomedi implementation reads much more naturally. The main difference is that Python's object-oriented structure allows us to bundle complex Comedi subdevice handling into a series of intuitive methods. We also benefit from Python's exception handling. While C requires you to actively check for exceptions where they might occur ("hey, this write failed"), Python exceptions bubble up the call stack so you can deal with them at a more appropriate level ("hey, the stepper motor failed"). This allows us to centralize error handling in higher level code, making the low level code much cleaner.

4.2.2 Pypiezo

The piezo controlling code builds on the framework established by pycomedi to define an interface for controlling the peizo-mounted surface (Fig. 2.1b). This involves code to sweep the piezo in a hardware-timed ramp as well as code for discrete jumps. To carry out these tasks, pypiezoalso

[1] Brooks [142] says "plan to throw one away," although I'm more optimistic about the feasibility of long-term maintenance than he is.

```
from pycomedi.device import Device
from pycomedi.channel import DigitalChannel
from pycomedi.constant import SUBDEVICE_TYPE, IO_DIRECTION

device = Device('/dev/comedi0')
device.open()

subdevice = device.find_subdevice_by_type(SUBDEVICE_TYPE.dio)
channels = [subdevice.channel(i, factory=DigitalChannel)
            for i in (0, 1, 2, 3)]
for chan in channels:
    chan.dio_config(IO_DIRECTION.output)

def write(value):
    subdevice.dio_bitfield(bits=value, write_mask=2**4-1)
```

Figure 4.4: A four-channel digital output example in pycomedi (from the stepper doctest). Compare this with the much more verbose Fig. 4.2, which is analogous to the `subdevice.dio_bitfield()` call.

contains code to convert piezo motion (in meters) to DAC output voltages (in bits), an h5config-based framework for automatically configuring pycomedi channels and piezo axes, and surface detection logic.

Because of the tight coupling needed between piezo motion and cantilever deflection detection for synchronized ramps, the basic `Piezo` class can be configured with generic pycomedi input channels. In practice, only the cantilever deflection is monitored, but if other pypiezo users want to measure other analog inputs, the functionality is already built in.

The surface detection logic is somewhat heuristic, although it has proven quite robust in practice. Given a particular piezo axis, target deflection, number of steps, and an allowed piezo range, the procedure is:

1. Ramp the piezo from its current position away to its maximum separation $z_{\max}$.
2. Step the piezo in towards its minimum separation, checking the deflection after each step to see if the target deflection threshold has been crossed. This is the high-contact piezo position $z_{\min}$.
3. Ramp the piezo away to its maximum separation $z_{\max}$. Because of protein on the surface, the

detachment region between the contact region and non-contact region may have additional forces (Fig. 2.5b). This can make the determination of the contact kink difficult. The full retraction breaks any protein-based contacts between the cantilever and the surface.

4. Ramp the piezo in to the high-contact position z_{min}. Because any protein-based contacts were broken in the previous step, the contact kink at z_{kink} should be clear and crisp.

5. Ramp the piezo back to the original position.

The deflection data $d(z)$ from 4, which should clearly show the contact kink, is fit to a bilinear model (a linear non-contact region and a linear contact region, which meet at the the contact kink). The fitting is carried out by minimizing the residual difference between the approach data and bilinear model with SciPy's `leastsq` optimizer, a wrapper around MINPACK's `lmdif` and `lmder` algorithms[145,146].

$$d(z) = \begin{cases} d_{\text{kink}} + \sigma_{p,\text{c}}(z - z_{\text{kink}}) & z \leq z_{\text{kink}} \\ d_{\text{kink}} + \sigma_{p,\text{nc}}(z - z_{\text{kink}}) & z \geq z_{\text{kink}} \end{cases} \tag{4.1}$$

Initial parameters for the fit are:

$$d_{\text{kink}} = d(z_{\text{max}}) \tag{4.2}$$

$$z_{\text{kink}} = \frac{z_{\text{max}} - z_{\text{min}}}{2} \tag{4.3}$$

$$\sigma_{p,\text{c}} = 2 \cdot \frac{d(z_{\text{max}}) - d(z_{\text{min}})}{z_{\text{max}} - z_{\text{min}}} \tag{4.4}$$

$$\sigma_{p,\text{nc}} = 0 \tag{4.5}$$

The fitted d_{kink} is accepted unless:

- the fitted slope ratio $|\sigma_{p,\text{c}}/\sigma_{p,\text{nc}}|$ is less than a minimum threshold (which defaults to 10), or
- the fitted kink position z_{kink} is within an excluded z_{window} of the boundaries (z_{window} defaults to 2% of the total range $z_{\text{max}} - z_{\text{min}}$).

The default slope ratios work well for the cantilevers I generally use, but softer cantilevers may have enough drift that the minumum slope ratio threshold needs to be reduced. I have not yet run into problems with the default kink window. Although these are low level parameters, appropriate values may depend on details of the particular experimental setup. The h5config-based configuration structure makes it easy to configure (and record) the values of many similar heuristic parameters like these involved in robust experiment control.

In the event that the surface detection is not acceptable, pypiezo raises an exception which bubbles up the call stack until it is handled in unfold-protein (Section 4.2.4).

4.2.3 Pyafm

Sweeping piezos around and measuring the resulting cantilever deflection is the core of velocity-clamp force spectroscopy. However, our experimental apparatus contains some additional supporting hardware: a stepper motor for coarse positioning and a peltier/thermocouple module for temperature control. The pyafm module builds on pypiezo, stepper (Section 4.3.2), and pypid (Section 4.3.3) to provide an easy to use (and easy to configure) `AFM` class for controlling the whole package.

While the piezo tube is able to move the surface relative to the cantilever tip (Fig. 2.1), it only has a limited range (on the order of microns). Achieving such a small separation by hand when assembling the microscope is unlikely, so a stepper motor controlling a fine pitch screw is used for course positioning. Generic stepper control is handled by the stepper package, and pyafm builds on this to add h5config-based configuration (time delay between steps, approximate step size, approximate stepper backlash, digital control port, ...) and a few course positioning methods:

`AFM.move_away_from_surface` provides a safety mechanism that higher level applications can use to bail out. For example, when unfold-protein can not locate the surface (for example, a bubble in the fluid cell may be blocking the laser beam), it uses this method to put some distance between the cantilever and the surface, to avoid crashing the tip and breaking the cantilever. Less drastically, this method is also used by calibcant (Chapter 5) when it changes from the surface bump stage (Section 5.4.1) to the thermal vibration stage (Section 5.4.3).

```
class Unfolder (object):
    # ...
    def run(self):
        """Approach-bind-unfold-save[-plot] cycle.
        """
        ret = {}
        ret['timestamp'] = _email_utils.formatdate(localtime=True)
        ret['temperature'] = self.afm.get_temperature()
        ret['approach'] = self._approach()
        self._bind()
        ret['unfold'] = self._unfold()
        self._save(**ret)
        if _package_config['matplotlib']:
            self._plot(**ret)
        return ret
```

Figure 4.5: The main unfolding loop in unfold-protein. Compare this with the much more opaque pull phase in Fig. 4.1.

`AFM.stepper_approach` quickly positions the surface within piezo-range of the cantilever tip by stepping in (with the stepper motor) until the cantilever deflection crosses a target threshold. The piezo extension is kept constant during the approach, but a single stepper step only moves the surface ~ 170 nm, and our cantilevers can safely absorb deflections on that scale.

`AFM.move_just_onto_surface` is a more refined version of `AFM.stepper_approach`. This method uses pypiezo's surface detection algorithm to locate the surface kink position z_{kink}, and adjusts the stepper in single steps until the measured kink is within two stepper steps (~ 340 nm) of the centered piezo position. Then it shifts the piezo to position the cantilever tip at an exact offset from the measured kink. This precise positioning is used for running calibcant's bumps (Section 5.4.1), but the per-step piezo manipulation makes long distance approaches much slower than `AFM.stepper_approach`.

4.2.4 Unfold-protein

Capping the experimental control stack, unfold-protein adds the actual experiment logic to the lower level control software. The abstractions provided by the lower level code make for clean, easily adaptable code (Figs. 4.5 and 4.6).

```
class UnfoldScanner (object):
    # ...
    def run(self, stepper_tweaks=True):
        self._stop = False
        _signal.signal(_signal.SIGTERM, self._handle_stop_signal)
        self.unfolder.afm.move_away_from_surface()
        self.stepper_approach()
        for i in range(self.config['velocity']['num loops']):
            _LOG.info('on loop {} of {}'.format(
                    i, self.config['velocity']['num loops']))
            for velocity in self.config['velocity']['unfolding velocities']:
                if self._stop:
                    return
                self.unfolder.config['unfold']['velocity'] = velocity
                try:
                    self.unfolder.run()
                except _ExceptionTooFar:
                    if stepper_tweaks:
                        self.stepper_approach()
                    else:
                        raise
                except _ExceptionTooClose:
                    if stepper_tweaks:
                        self.afm.move_away_from_surface()
                        self.stepper_approach()
                    else:
                        raise
                else:
                    self.position_scan_step()
```

Figure 4.6: The scanning loop unfold-protein. Unfolding pulls are carried out with repeated calls to `self.unfolder.run()` (Fig. 4.5), looping over the configured range of velocities for a configured number of cycles. If `stepper_tweaks` is `True`, the scanner adjusts the stepper position to keep the surface within the piezo's range. After a successful pull, `self.position_scan_step()` shifts the piezo in the x direction, so the next pull will not hit the same surface location.

```
import h5config.config as _config

class AxisConfig (_config.Config):
    "Configure a single piezo axis"
    settings = [
        _config.FloatSetting(
            name='gain',
            help=(
                'Volts applied at piezo per volt output from the DAQ card '
                '(e.g. if your DAQ output is amplified before driving the '
                'piezo),')),
        _config.FloatSetting(
            name='sensitivity',
            help='Meters of piezo deflection per volt applied to the piezo.'),
        # ...
        _config.ConfigSetting(
            name='channel',
            help='Configure the underlying DAC channel.',
            config_class=OutputChannelConfig,
            default=None),
        # ...
        ]
```

Figure 4.7: Portions of the configuration class for a single piezo axis (from pypiezo, Section 4.2.2). The more generic analog output channel configuration is nested under the `channel` setting.

4.3 Auxiliary packages

The previous section covered the core of the experiment stack (Section 4.2), but skipped over some of the more peripheral packages.

4.3.1 h5config

The h5config package makes it easy to save and load configuration classes from disk. After populating base configuration classes with parameters (Fig. 4.7), h5config automatically generates HDF5 and YAML backends for saving and loading that class.

Basic configuration types include booleans, integers, floating point numbers, enumerated choices, and freeform text. There is also support for lists of these basic types (e.g. lists of integers). The key feature is nesting configuration classes. This means that your higher level tools can have their own configuration settings and also include the configuration settings for their lower level components.

For example, the piezo axis configuration given in Fig. 4.7 contains configuration settings specific to piezo axes, and it also contains a reference to the configuration settings for a generic analog output channel. The piezo axis code doesn't need to know what the analog output channel configuration settings are, those are defined somewhere else.

The nesting continues all the way up the stack, to the unfold-protein configuration. This means that a single file (`~/.config/unfold_protein.yaml`) contains every configurable setting required for the whole experiment in an easy-to-edit text format. Adding additional configuration settings at any level of the experiment stack is just a matter of adjusting a single `h5config.config.Config` subclass the corresponding entry in the configuration file. There is no need to adjust the higher level code, the new setting is passed down the stack to its point of use automatically.

Besides making it easy to configure your experiment, h5config also makes it easy to save the configuration alongside your data. The section of unfold-protein that writes the whole configuration stack into the per-pull HDF5 file is only four lines long. This makes post-processing much easier, because almost every setting needed to analyze the data is already stored in the data file (the only missing values are those that you did not need during the experiment control phase).

4.3.2 stepper

Because of thermal drift and mechanical instability, the distance between the tip and the surface changes significantly over time. When the distance change exceeds the range of the piezo scanner, the stepper motor must be engaged to reposition the AFM tip relative to the sample. The earlier LabVIEW software (Section 4.1.1) lacked the ability to control the motor on its own, so it would pause roughly every half hour and prompt the operator to make the necessary manual adjustments. Automatic motor control allows the system to run longer without interrupts, facilitating the collection of large data sets.

The stepper package provides Python control of stepper motors[149]. The package is mostly concerned with the maintenance of internal motor state:

position is a half-step counter that records the current motor position.

full step selects full or half stepping.

logic selects active high or active low operation.

delay sets the time delay between steps in seconds, in case the motor response is slower than the digital output driver.

step size approximates the step size in meters.

backlash estimates the drive chain backlash in half-steps.

Actualizing the motor control signal is left up to the caller, in this case pyafm.

We verified the stability and reproducibility of the microscopic movement of the motor by making several approach-retreat cycles from the surface of ∼70 steps which resulted in the data displayed in Fig. 4.8a. We also measured the distance the surface moved with every step by determining the change in deflection voltage as a function of peizo position as we stepped the AFM tip closer the the surface. Our stepsize data is displayed in Fig. 4.8b.

The motor is very consistent when approaching the surface, which indicates that our control software is operating correctly. However, the motor exhibits some hysteretic behavior on a scale of ∼46 steps, which is almost certainly due to *backlash*, or slack in the motor–surface coupling machinery. The first 46 steps in a new direction take the slack out of the coupling, and further steps move the tip relative to the surface. The problem can be avoided entirely by simply replacing "backwards motion by one step" with "backwards motion by 60 steps and forward motion by 59 steps".

One issue raised by backlash is that it might be the source of some of our surface drift, as the drive-chain relaxes towards some central value and pulls the surface with it. By oscillating into our eventual position, we could perhaps settle the system at the beginning, reducing the need for adjustments later on. While this is not a problem for the current unfolding experiments, it could be an issue for longer unfolding-refolding experiments.

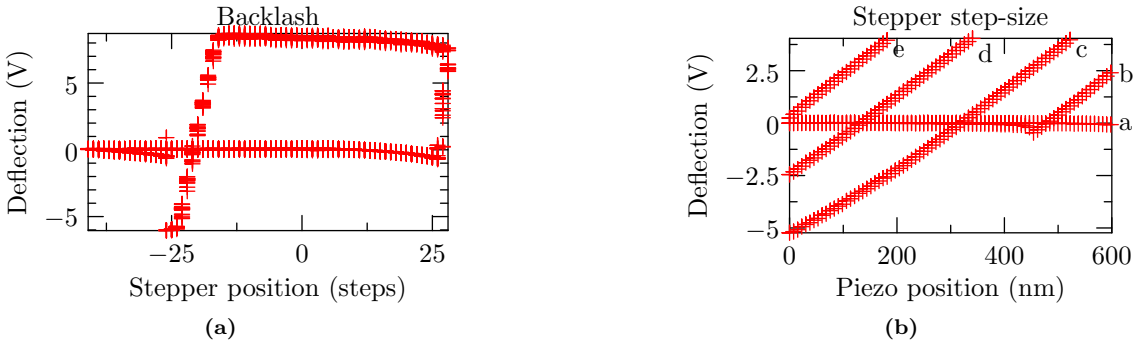


Figure 4.8: (a) Stepper motor reproducibility, stability, and backlash. The data are from a single continuous counterclockwise trace of 14 approach-retreat cycles. The jump from about $(18, -6)$ to $(17, -0.4)$ is the snap-off effect, where short-range attractive interactions between the tip and the sample—due to surface wetting in air—require the tip to be actively pulled off surface. Signal noise is comparable to that expected by drift. (b) Motor step size calibration. The stepper gradually stepped closer to the surface, feeling forward with the piezo after each step. Successive motor positions yield traces *a*, *b*, *c*, *d*, and *e*. As the motor moves the sample closer, less piezo movement is required to approach to same deflection level. The average spacing between the traces is roughly 170 nm. Traces *c* and *d* have regions of negative deflection because the tip no longer retracts far enough from the surface to break free of the snap-off effect. There is no backlash because the data were taken during a single approach.

4.3.3 pypid

The final component of the experiment control stack is pypid, which uses pymodbus to communicate with a Melcor Series MTCA Thermoelectric Cooler Controller[151] over a serial line. The controller monitors the fluid cell temperature with a thermocouple, and reading temperatures from the controller is fairly straightforward (Fig. 4.5). Temperature control is via a Peltier device mounted underneath the sample surface (Fig. 4.9).

The controller tries to keep the measured temperature at the setpoint temperature via a modified proportional-integral-derivative (PID) feedback algorithm. PID systems have been around for a while[152], but finding appropriate feedback terms for sensitive systems is not trivial. There are a number of tuning procedures which characterize the system by evaluating its response under simpler driving conditions. The pypid package implements Ziegler–Nichols' step response[152], bang-bang response, and ultimate cycle response[152] tuning rules, as well as Cohen–Coon's[153] and Wang–Juang–Chan's[154] step response tuning rules[155].

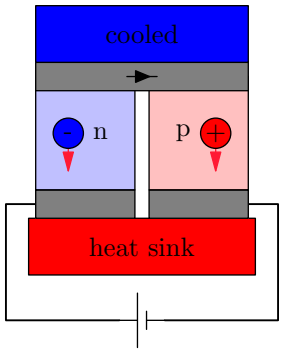


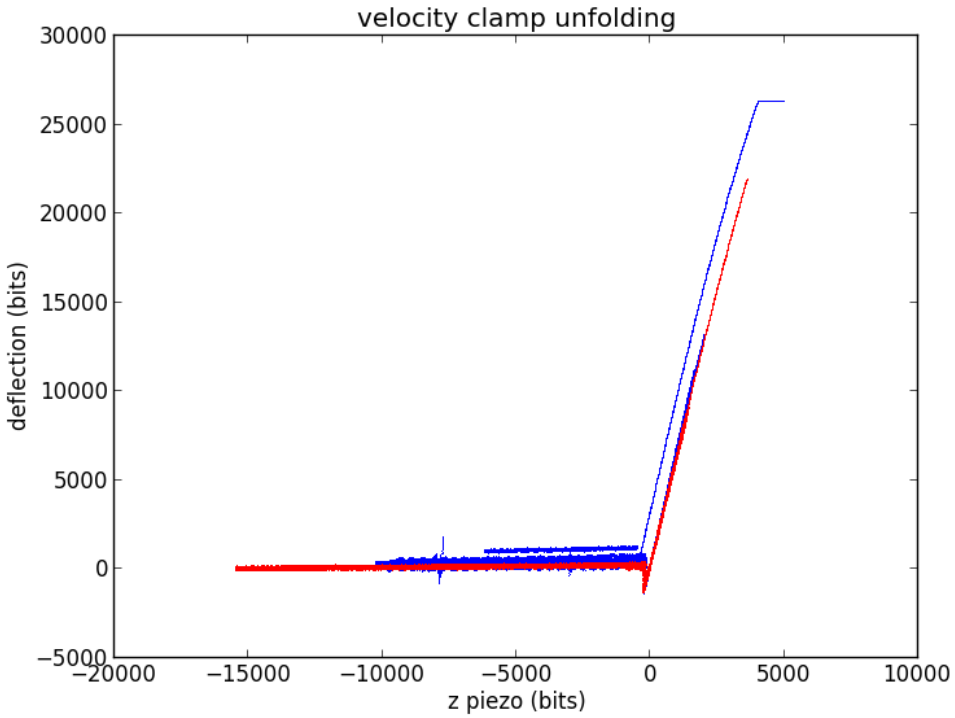
Figure 4.9: A Peltier functions by applying a voltage to regions of p- and n-type semiconductor in series. Conduction in n-type semiconductors is mainly through thermally excited electrons and in p-type semiconductors is mainly through thermally excited holes. Applying a positive voltage as shown in this figure cools the sample by constantly pumping hot conductors in both semiconductors towards heat sink, which radiates the heat into the environment. Reversing the applied voltage heats the surface.

4.4 Discussion

With the radical shift from LabVIEW and Microsoft Windows over to Comedi and Linux, it is a good idea to compare my new experiment control software with the earlier stack. Because the fundamental procedure in my experiments is the velocity-clamp pull (Section 2.4), I used both approaches in quick succession to collect pulls. Because the stacks diverge after the PCI DAQ card, I was able to collect several pulls using my setup, power down the Linux computer, swap the PCI card into the Windows computer, power up, and collect several pulls using the Windows stack on top of the exact same hardware.

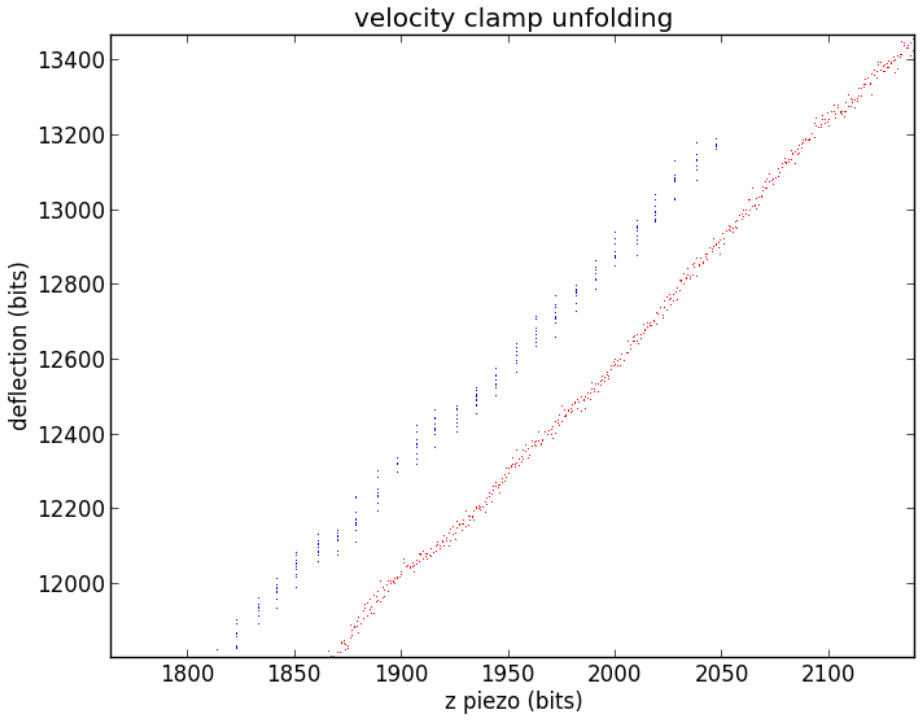
Because the goal of these experiments was to compare the two software stacks, the comparison was carried out using our standard AFM cantilevers and gold surface, but distilled water was used instead of PBS and no protein was bound to the surface. This gives a simpler system with fewer distracting features. As shown in Figs. 4.10 and 4.11, large-scale features are identical, with similar contact slopes and non-contact noise.

Although grossly similar, the two stacks do have some statistically significant differences. The slope of the contact region for the LabVIEW/Windows stack (excluding the out-of-deflection-range outlier) is 6.59 ± 0.13, while the Comedi/Linux stack slope is 6.17 ± 0.03 (both in deflection bits per



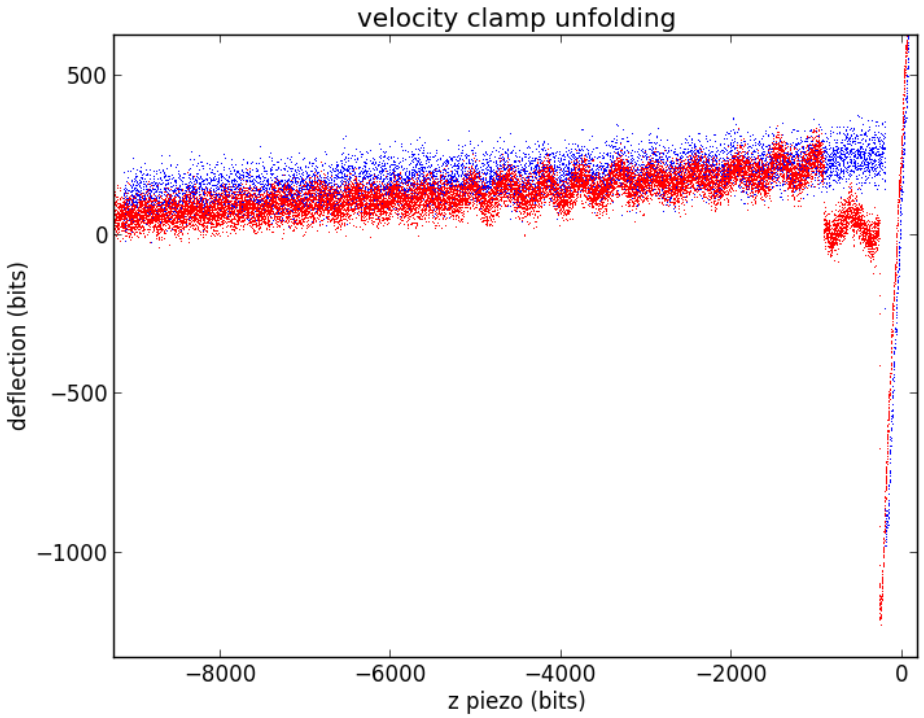
(a)

Figure 4.10: (a)Several velocity clamp pulls using both the LabVIEW/Windows stack (blue) and the Comedi/Linux stack (red). The contact voltage and pulling distance were not synchronized between the two experiments, and the raw data has been shifted to locate the contact point at the origin. One LabVIEW/Windows curve has a flat deflection in the high-contact region, where the laser was deflected beyond the photodiode's working range. This is probably due to a high approach setpoint, followed by surface drift during the binding phase, but is not relevant to the stack comparison.



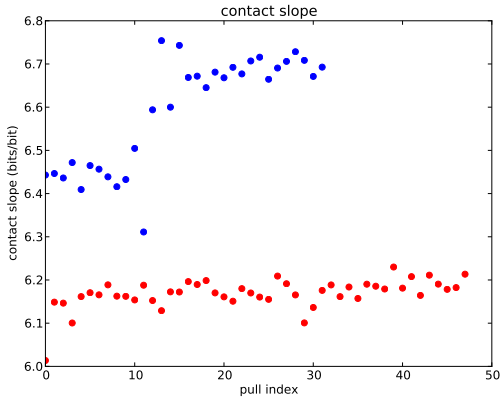
(b)

Figure 4.10: (b)The contact region from a single pull from (a). The LabVIEW/Windows stack (blue) takes 0.5 nm piezo steps with ten deflection reads at each step. The Comedi/Linux stack (red) makes a single read per step, but can take as many small steps as possible within DAQ card's memory buffer, frequency, and precision limitations. For 1 μm/s pulls, a stepping/sampling frequency of 50 kHz generated steps that were less than one DAC bit wide.

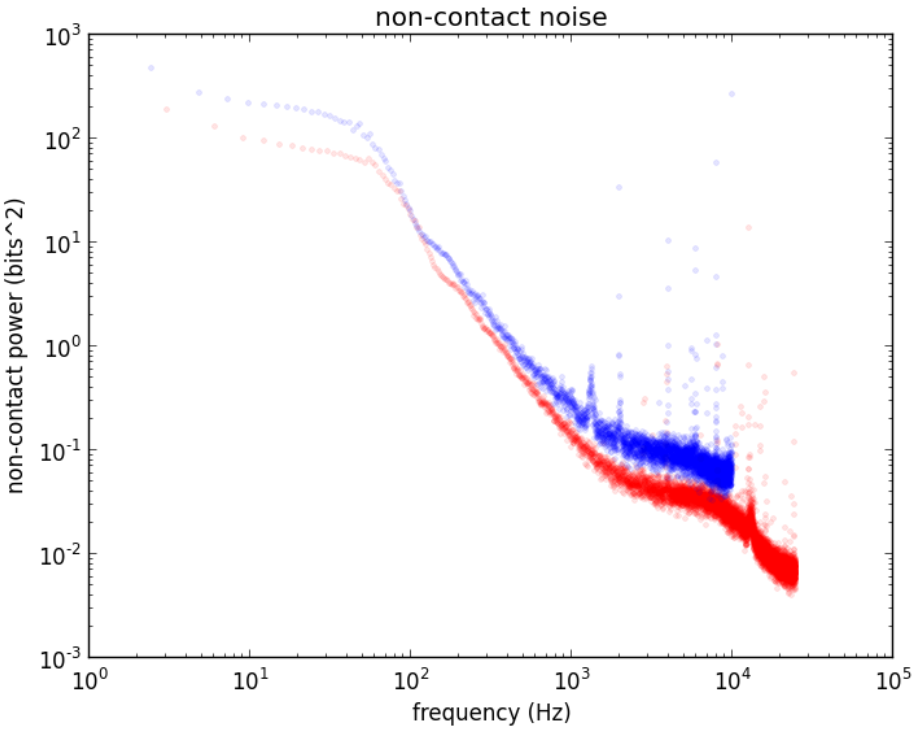


(c)

Figure 4.10: (c)The non-contact region from a single pull from (a) using both the LabVIEW/Windows stack (blue) and the Comedi/Linux stack (red). The signal oscillates because the AFM is sitting directly on the lab bench, our usual isolation mechanisms being unavailable when these curves were recorded. All protein unfolding experiments were carried out with isolation, so the vibration was not a problem in those cases.



(a)



(b)

Figure 4.11: (a)Contact slope for the pull in Fig. 4.10 using both the LabVIEW/Windows stack (blue) and the Comedi/Linux stack (red). The low-slope outlier is from the pull with out-of-range deflection. (b)Power spectral densities (PSDs) of the non-contact noise for the pulls from 4.10a. To produce this image, the PSD of the non-contact data extracted from 4.10a was averaged for each software stack. The number of points in the non-contact region truncated to the nearest power of two (for efficient fast Fourier transformation), which has the convenient side effect of aligning the frequency axis for easy cross-pull averaging.

z-piezo bit, Fig. 4.11a). While small, the 7% difference is significant, both statistically (3σ for the LabVIEW/Windows standard deviation) and practically, because contact slope plays a key role in cantilever calibration (Section 5.4.1).

Due to the table vibration, comparisons of non-contact noise between the two stacks are less conclusive. However, rough comparisons of the noise spectra show that noise in the Comedi/Linux data is generally a factor of two to three less than noise in the LabVIEW/Windows data across a range of frequencies (Fig. 4.11b). While the noise difference may be small, it does highlight the importance (and difficulty) of characterizing your apparatus and controlling software.

4.5 Conclusions

Developing an open software stack for controlling single molecule force spectroscopy is hard work, especially for scientists who lack experience designing or managing moderately large software projects. Coming into this project, I already had several years of experience working with LabVIEW, but I had very little experience in other languages and no formal training in project maintenance. I spent the first two years of my research project acquiring enough experience to start making progress on a sustainable stack[2], and I've spent the remaining time tuning this stack (and the analysis software) while running experiments.

An open source stack allows collaborative development so that this development cost can be shared between labs, as well as lowering the barrier to entry for new labs entering the field. Besides benefiting SMFS groups, lower-level packages in the stack will be useful to a wider audience (who can share the maintenance cost). My existing stack and future distributed maintenance will allow researchers to focus on generating new science, instead of generating new software.

Besides development efficiencies, a common stack could provide a benchmark for comparative analysis between experiments carried out by different labs. With every lab using in-house software and in-house hardware, it's hard to judge the reliability or accuracy of the lab's published research. A common stack should include methods like those used in Section 4.4 to characterize and validate

[2]Since last summer I've been helping the Software Carpentry project [134] reach out to scientists (mostly graduate students) to provide boot camp introductions to software development and version control. It's a chance to tell other folks what I wish I'd been told when I was starting out.

your apparatus. A common stack also provides a common file format for experimental data, which makes it easier to share data and analysis tools.

While there have been other attempts SMFS control which claim to be open source, they have either been based on closed source tools[156] or have (critically) not actually published their source[157]. Both of these limitations make it hard to realize the benefits of communal, open source development, and the pyafm stack suffers from neither.

Chapter 5: Cantilever spring constant calibration

The most common method for calibrating cantilevers for atomic force microscopes is via thermal vibration[27]. In this chapter, I'll derive the theory behind this procedure and introduce my calibcant package for performing this calibration automatically.

We know the energy of the cantilever's thermal vibration from the equipartion theorem (Eq. (2.2) and Section 2.5). Solving the equipartition theorem for the spring contant κ yields

$$\kappa = \frac{k_B T}{\langle x^2 \rangle} \,, \tag{5.1}$$

so we need to measure (or estimate) the temperature T and variance of the cantilever position $\langle x^2 \rangle$ in order to estimate κ.

We don't measure x directly, though. We reflect a laser off the back of the cantilever and measure the position of the deflected beam with a photodiode (Fig. 2.1a). In order to convert the photodiode signal V_p to a tip displacement x, we scale V_p by a linear photodiode sensitivity σ_p.

$$x(t) = \frac{V_p(t)}{\sigma_p} \,. \tag{5.2}$$

We measure σ_p by pushing the tip against the substrate surface and measuring the slope (deflection volts per piezo meter) of the resulting contact-deflection trace (Section 5.4.1). By keeping V_p and σ_p separate in our calculation of κ, we can gauge the relative importance errors in each parameter and calculate the uncertainty in our estimated κ (Section 5.5.2).

In order to filter out noise in the measured value of $\langle V_p^2 \rangle$ we fit the measured cantilever deflection to the expected theoretical power spectral density (PSD_f) of a damped harmonic oscillator exposed to thermal noise

$$\mathrm{PSD}_f(V_p, f) = \frac{G_{1f}}{(f_0^2 - f^2)^2 + \beta_f^2 f^2} \,. \tag{5.3}$$

In terms of the fit parameters G_{1f}, f_0, and β_f, the expectation value for V_p^2 is given by

$$\left\langle V_p(t)^2 \right\rangle = \frac{\pi G_{1f}}{2\beta_f f_0^2} \ . \tag{5.4}$$

Combining Eqs. (5.1), (5.2) and (5.4), we have

$$\kappa = \frac{\sigma_p^2 k_B T}{\langle V_p(t)^2 \rangle} = \frac{2\beta_f f_0^2 \sigma_p^2 k_B T}{\pi G_{1f}} \ . \tag{5.5}$$

A calibration run consists of bumping the surface with the cantilever tip to measure σ_p (Section 5.4.1), measuring the buffer temperature T with a thermocouple (Section 5.4.2), and measuring thermal vibration when the tip is far from the surface to extract the fit parameters G_{1f}, f_0, and β_f (Section 5.4.3).

Although this theory should be well established (Section 5.1), there is continued confusion about the details of the fitting (Section 5.2). To avoid further ambiguity, I'll derive the power spectral density mentioned above (Eqs. (5.3) and (5.4)) in Section 5.3. In Section 5.4, I'll introduce my calibcant package for automatically calibrating cantilevers. I'll clear up a few remaining points in Section 5.5 before wrapping up with Section 5.6.

5.1 Related work

In reality, the cantilever motion is more complicated than a pure simple harmonic oscillator. Various corrections taking into acount higher order vibrational modes [158,159] and cantilever tilt [160] have been proposed and reviewed [27,54,161], but we will focus here on the derivation of noise in damped simple harmonic oscillators that underlies all frequency-space methods for improving the basic $\kappa \left\langle x^2 \right\rangle = k_B T$ method.

Roters and Johannsmann [162] derive the PSD with a similar Fourier transform, but they use the fluctuation–dissipation theorem to extract the PSD from the susceptibility (see their Eq. (4)). Benedetti [163] has independently developed a Parseval's approach similar to mine (in his Section 8.2.1),

although he glosses over some of the integrals. Berg-Sørensen and Flyvbjerg [164] has an extensive treatment of the extremely overdamped case and laser tweezer calibration, which they revisit a year later during a discussion of noise color [165]. Gittes and Schmidt [166] derive some related results in the extremely overdamped case, such the fact that the signal to thermal noise ratio is independent of trap stiffness κ. Despite this earlier work, I think it is worth explicitly deriving the PSD of a damped harmonic oscillator here, as I have been unable to find a reference that I feel treats the problem with sufficient rigor. An explicit derivation may also help clear up the confusion about the proper PSD form discussed in the next section.

5.2 Fitting with a Lorentzian

It is popular to refer to the thermal power spectral density as a "Lorentzian" [27,28,54,162,167], but there is dissagreement on what this means. The classic Lorentzian function is [168]

$$L(x) = \frac{1}{\pi} \frac{\frac{1}{2}\Gamma}{(x - x_0)^2 + \left(\frac{1}{2}\Gamma\right)^2} , \tag{5.6}$$

where x_0 sets the center and Γ sets the width of the curve. However, the correct PSD for a damped harmonic oscillator in a white noise bath is given by Eq. (5.3) [26,163].

These formulas are fundamentally different.

For example, the slope of Eq. (5.3) is zero at $f = 0$, as we can see by using the chain rule

repeatedly,

$$\frac{\mathrm{dPSD}_f}{\mathrm{d}f} = \frac{\mathrm{d}}{\mathrm{d}f}\left(\frac{G_{1f}}{(f_0^2-f^2)^2+\beta_f^2 f^2}\right) = \frac{-G_{1f}}{\left((f_0^2-f^2)^2+\beta_f^2 f^2\right)^2}\frac{\mathrm{d}}{\mathrm{d}f}\left((f_0^2-f^2)^2+\beta_f^2 f^2\right) \tag{5.7}$$

$$= \frac{-G_{1f}}{\left((f_0^2-f^2)^2+\beta_f^2 f^2\right)^2}\left(2(f_0^2-f^2)\frac{\mathrm{d}}{\mathrm{d}f}(f_0^2-f^2)+2\beta_f^2 f\right) \tag{5.8}$$

$$= \frac{-G_{1f}}{\left((f_0^2-f^2)^2+\beta_f^2 f^2\right)^2}\left(-4f(f_0^2-f^2)+2\beta_f^2 f\right) \tag{5.9}$$

$$= \frac{2G_{1f}f}{\left((f_0^2-f^2)^2+\beta_f^2 f^2\right)^2}\left(2(f_0^2-f^2)-\beta_f^2\right) \tag{5.10}$$

$$\left.\frac{\mathrm{dPSD}_f}{\mathrm{d}f}\right|_{f=0} = 0\,. \tag{5.11}$$

On the other hand, the slope of Eq. (5.6) is only zero at the peak (where $x = x_0$).

$$\frac{\mathrm{d}L(x)}{\mathrm{d}x} = \frac{1}{\pi}\frac{\frac{-1}{2}\Gamma}{\left((x-x_0)^2+\left(\frac{1}{2}\Gamma\right)^2\right)^2}\cdot\frac{\mathrm{d}}{\mathrm{d}x}\left((x-x_0)^2+\left(\frac{1}{2}\Gamma\right)^2\right) \tag{5.12}$$

$$= \frac{1}{\pi}\frac{\frac{-1}{2}\Gamma}{\left((x-x_0)^2+\left(\frac{1}{2}\Gamma\right)^2\right)^2}\cdot 2(x-x_0) \tag{5.13}$$

$$= \frac{1}{\pi}\frac{-\Gamma(x-x_0)}{\left((x-x_0)^2+\left(\frac{1}{2}\Gamma\right)^2\right)^2} \tag{5.14}$$

It is unclear whether the "Lorentzian" references are due to uncertainty about the definition of the Lorentzian or to the fact that the two equations have similar behavior near the peak. Florin et al.[27] likely *are* using Eq. (5.6), as the slope of the fitted PSD in their Fig. 2, has a slope at $f = 0$. If they were using Eq. (5.3), the derivative would have been zero (Eq. (5.11)).

We have at least two models in use, one likely the "Lorentzian" (Eq. (5.6)) and one that's not. Perhaps researchers claiming to use the "Lorentzian" are consistently using Eq. (5.6)? There are at least two counterexamples—Roters and Johannsmann[162], Benedetti[163]—with solid derivations of Eq. (5.45) which they then refer to as the "Lorentzian". Which formula are the remaining "Lorentzian" fitters using? What about groups that only reference their method as "thermal calibration" without specifying a PSD model? In order to avoid any uncertainty, we leave Eq. (5.3)

unnamed. I encourage future researchers to explicitly list the model they use, ideally by citing their associated open source calibration package.

5.3 Power spectra of damped harmonic oscillators

As discussed in Section 5.2, the power spectral density for a Hookean cantilever is surprisingly ambiguous. In this section, I'll derive the frequency-space power spectra of the deflection voltage (Eqs. (5.4) and (5.38)), modeling the cantilever as a damped harmonic oscillator.

$$m\ddot{x} + \gamma\dot{x} + \kappa x = F(t) \ , \tag{5.15}$$

where x is the displacement from equilibrium, m is the effective mass, γ is the effective drag coefficient, κ is the spring constant, and $F(t)$ is the external driving force. During the non-contact phase of calibration, $F(t)$ comes from random thermal noise.

In the following analysis, we use the unitary, angular frequency Fourier transform normalization

$$\mathcal{F}\left\{x(t)\right\} \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \mathrm{d}t \ , \tag{5.16}$$

where ω is the angular frequency and $i \equiv \sqrt{-1}$ is the imaginary unit.

We also use the following theorems (proved elsewhere):

$$\cos\left(\frac{\theta}{2}\right) = \pm\sqrt{\frac{1}{2}[1+\cos(\theta)]} \ , \tag{5.17}$$

[169]

$$\mathcal{F}\left\{\frac{\mathrm{d}^n x(t)}{\mathrm{d}t^n}\right\} = (i\omega)^n x(\omega) \ , \tag{5.18}$$

[170]

$$\int_{-\infty}^{\infty} |x(t)|^2 \mathrm{d}t = \int_{-\infty}^{\infty} |x(w)|^2 \mathrm{d}\omega \ . \tag{5.19}$$

(Parseval's) [171]

As a corollary to Parseval's theorem, we note that the one sided power spectral density per unit time (PSD) defined by

$$\mathrm{PSD}(x,\omega) \equiv \lim_{t_T \to \infty} \frac{1}{t_T} \, 2|x(\omega)|^2 \tag{5.20}$$

[172]

relates to the variance by

$$\left\langle x(t)^2 \right\rangle = \lim_{t_T \to \infty} \frac{1}{t_T} \int_{-t_T/2}^{t_T/2} |x(t)|^2 \mathrm{d}t = \lim_{t_T \to \infty} \frac{1}{t_T} \int_{-\infty}^{\infty} |x(\omega)|^2 \mathrm{d}\omega = \int_0^{\infty} \mathrm{PSD}(x,\omega)\mathrm{d}\omega \ , \tag{5.21}$$

where t_T is the total time over which data has been aquired.

We also use the Wiener–Khinchin theorem, which relates the two sided power spectral density $S_{xx}(\omega)$ to the autocorrelation function $r_{xx}(t)$ via

$$S_{xx}(\omega) = \mathcal{F}\left\{r_{xx}(t)\right\} \ , \qquad \text{(Wiener–Khinchin)}\,^{173} \tag{5.22}$$

where $r_{xx}(t)$ is defined in terms of the expectation value

$$r_{xx}(t) \equiv \langle x(\tau)\overline{x}(\tau - t)\rangle \ , \qquad ^{174} \tag{5.23}$$

and $\overline{x}$ represents the complex conjugate of x.

5.3.1 Highly damped case

For highly damped systems, the inertial term in Eq. (5.15) becomes insignificant ($m \to 0$). This model is commonly used for optically trapped beads[175]. Because it is simpler and solutions are more easily available[26,175,176], it will serve to outline the general approach before we dive into the general case.

Fourier transforming Eq. (5.15) with $m = 0$ and applying Eq. (5.18) we have

$$(i\gamma\omega + \kappa)x(\omega) = F(\omega) \tag{5.24}$$

$$|x(\omega)|^2 = \frac{|F(\omega)|^2}{\kappa^2 + \gamma^2\omega^2} \ . \tag{5.25}$$

We compute the PSD by plugging Eq. (5.25) into Eq. (5.20)

$$\mathrm{PSD}(x,\omega) = \lim_{t_T \to \infty} \frac{1}{t_T} \frac{2\,|F(\omega)|^2}{\kappa^2 + \gamma^2\omega^2} \ . \tag{5.26}$$

Because thermal noise is white (not autocorrelated + Wiener–Khinchin Theorem), we can write the one sided thermal power spectral density per unit time as

$$G_0 \equiv \mathrm{PSD}(F,\omega) = \lim_{t_T\to\infty} \frac{1}{t_T}\, 2\left|F(\omega)\right|^2 \ . \tag{5.27}$$

Plugging Eq. (5.27) into Eq. (5.26) we have

$$\mathrm{PSD}(x,\omega) = \frac{G_0}{\kappa^2+\gamma^2\omega^2} \ . \tag{5.28}$$

This is the formula we would use to fit our measured PSD, but let us go a bit farther to find the expected PSD and thermal noise given γ and κ.

Integrating over positive ω to find the total power per unit time yields

$$\int_0^\infty \mathrm{PSD}(x,\omega)\mathrm{d}\omega = \int_0^\infty \frac{G_0}{\kappa^2+\gamma^2\omega^2}\mathrm{d}\omega = \frac{G_0}{\gamma}\int_0^\infty \frac{1}{\kappa^2+z^2}\mathrm{d}z = \frac{\pi G_0}{2\gamma\kappa} \ , \tag{5.29}$$

where we made the simplifying replacement $z \equiv \gamma\omega$, so $\mathrm{d}\omega = \mathrm{d}z/\gamma$. The integral is solved in Appendix A.2.1.

Plugging into our corollary to Parseval's theorem (Eq. (5.21)),

$$\left\langle x(t)^2 \right\rangle = \frac{\pi G_0}{2\gamma\kappa} \ . \tag{5.30}$$

Plugging Eq. (5.30) into Eq. (2.2) we have

$$\kappa\frac{\pi G_0}{2\gamma\kappa} = k_B T \tag{5.31}$$

$$G_0 = \frac{2\gamma k_B T}{\pi} \ . \tag{5.32}$$

Combining Eqs. (5.28) and (5.32), we expect $x(t)$ to have a power spectral density per unit time

given by[1]

$$\mathrm{PSD}(x,\omega) = \frac{2}{\pi} \cdot \frac{\gamma k_B T}{\kappa^2 + \gamma^2 \omega^2} \ . \tag{5.33}$$

5.3.2 General form

The procedure here is exactly the same as the previous section. The integral normalizing G_0, however, becomes a little more complicated.

Fourier transforming Eq. (5.15) and applying Eq. (5.18) we have

$$(-m\omega^2 + i\gamma\omega + \kappa)x(\omega) = F(\omega) \tag{5.34}$$

$$(\omega_0^2 - \omega^2 + i\beta\omega)x(\omega) = \frac{F(\omega)}{m} \tag{5.35}$$

$$|x(\omega)|^2 = \frac{|F(\omega)|^2/m^2}{(\omega_0^2 - \omega^2)^2 + \beta^2\omega^2} \ , \tag{5.36}$$

where $\omega_0 \equiv \sqrt{\kappa/m}$ is the resonant angular frequency and $\beta \equiv \gamma/m$ is the drag-acceleration coefficient.

We compute the PSD by plugging Eq. (5.36) into Eq. (5.20)

$$\mathrm{PSD}(x,\omega) = \lim_{t_T \to \infty} \frac{1}{t_T} \ \frac{2|F(\omega)|^2/m^2}{(\omega_0^2 - \omega^2)^2 + \beta^2\omega^2} \ . \tag{5.37}$$

Plugging Eq. (5.27) into Eq. (5.37) we have[2]

$$\mathrm{PSD}(x,\omega) = \frac{G_0/m^2}{(\omega_0^2 - \omega^2)^2 + \beta^2\omega^2} \ . \tag{5.38}$$

[1] Eq. (5.33) is Bechhoefer and Wilson [175] Eq. (A12) (who's $\tau_0 \equiv \gamma/\kappa$), except that they're missing a factor of $1/\pi$. Eq. (5.33) is also Burnham et al. [26] Eq. (8), where their damping coefficient b is equivalent to our γ, their frequency ν is equivalent to our $f = \omega/2\pi$, and their roll off frequency $\nu_R \equiv k/2\pi b$ is equivalent to our $\kappa/2\pi\gamma$.

[2] Equation (5.38) is Roters and Johannsmann [162] Eq. (4)

Integrating over positive ω to find the total power per unit time yields

$$\int_0^\infty \mathrm{PSD}(x,\omega)\mathrm{d}\omega = \frac{G_0}{2m^2}\int_{-\infty}^{\infty}\frac{1}{(\omega_0^2-\omega^2)^2+\beta^2\omega^2}\mathrm{d}\omega = \frac{G_0}{2m^2}\cdot\frac{\pi}{\beta\omega_0^2} = \frac{\pi G_0}{2m^2\beta\omega_0^2} \tag{5.39}$$

where the integration is solved in Appendix A.2.2[3]. By the corollary to Parseval's theorem (Eq. (5.21)), we have

$$\left\langle x(t)^2\right\rangle = \frac{\pi G_0}{2m^2\beta\omega_0^2}\,. \tag{5.42}$$

Plugging Eq. (5.42) into the equipartition theorem (Eq. (2.2)) we can reproduce Eq. (5.32).

$$\kappa\frac{\pi G_0}{2m^2\beta\omega_0^2} = k_BT \tag{5.43}$$

$$G_0 = \frac{2m^2\beta\omega_0^2k_BT}{\pi\kappa} = \frac{2m^2\beta\frac{\kappa}{m}k_BT}{\pi\kappa} = \frac{2m\beta k_BT}{\pi} = \frac{2m\frac{\gamma}{m}k_BT}{\pi} = \frac{2\gamma k_BT}{\pi}\,. \tag{5.44}$$

Combining Eqs. (5.38) and (5.44), we expect $x(t)$ to have a power spectral density per unit time given by[4]

$$\mathrm{PSD}(x,\omega) = \frac{2k_BT\beta}{\pi m\left[(\omega_0^2-\omega^2)^2+\beta^2\omega^2\right]}\,. \tag{5.45}$$

As expected, we can recover the extremely overdamped form Eq. (5.33) from the general form Eq. (5.45). Plugging in for $\beta \equiv \gamma/m$ and $\omega_0 \equiv \sqrt{\kappa/m}$,

$$\lim_{m\to 0}\mathrm{PSD}(x,\omega) = \lim_{m\to 0}\frac{2k_BT\gamma}{\pi m^2\left[\left(\frac{\kappa}{m}-\omega^2\right)^2+\frac{\gamma^2}{m^2}\omega^2\right]} = \lim_{m\to 0}\frac{2k_BT\gamma}{\pi\left[(\kappa-m\omega^2)^2+\gamma^2\omega^2\right]} \tag{5.46}$$

$$= \frac{2}{\pi}\cdot\frac{\gamma k_BT}{\kappa^2+\gamma^2\omega^2}\,. \tag{5.47}$$

[3] Comparing Eqs. (5.29) and (5.39), we see

$$\frac{\pi G_0}{2m^2\beta\omega_0^2} = \frac{\pi G_0}{2m^2\frac{\gamma}{m}\frac{k}{m}} = \frac{\pi G_0}{2\gamma\kappa}\,. \tag{5.40}$$

This is not a coincidence. Both spectra satisfy the equipartion theorem, so

$$\int_0^\infty \mathrm{PSD}(x,\omega)\mathrm{d}\omega = \left\langle x(t)^2\right\rangle = \frac{k_BT}{\kappa}\,, \tag{5.41}$$

which is the same for both cases.

[4] Eq. (5.45) is Benedetti [163] Eq. (8.11).

5.3.3 Fitting deflection voltage directly

In order to keep our errors in measuring σ_p separate from other errors in measuring $\langle x(t)^2 \rangle$, we can fit the voltage spectrum before converting to distance. Plugging Eq. (5.2) into Eq. (5.15),

$$\frac{\ddot{V_p}}{\sigma_p} + \beta \frac{\dot{V_p}}{\sigma_p} + \omega_0^2 \frac{V_p}{\sigma_p} = F(t) \tag{5.48}$$

$$\ddot{V_p} + \beta \dot{V_p} + \omega_0^2 V_p = \sigma_p \frac{F(t)}{m} \tag{5.49}$$

$$\ddot{V_p} + \beta \dot{V_p} + \omega_0^2 V_p = \frac{F_p(t)}{m} \ , \tag{5.50}$$

where $F_p(t) \equiv \sigma_p F(t)$. This has the same form as Eq. (5.15), which can be rearranged to:

$$\ddot{x} + \frac{\gamma}{m}\dot{x} + \frac{\kappa}{m}x = \frac{F(t)}{m} \tag{5.51}$$

$$\ddot{x} + \beta \dot{x} + \omega_0^2 x = \frac{F(t)}{m} \ , \tag{5.52}$$

so the PSD of $V_p(t)$ will be the same as the PSD of $x(t)$, after the replacements $x \to V_p(t)$, $F \to F_p$, and (because of Eq. (5.27)) $G_0 \to \sigma_p^2 G_0$. Making these replacements in Eqs. (5.38) and (5.42), we have

$$\mathrm{PSD}(V_p, \omega) = \frac{\sigma_p^2 G_0 / m^2}{(\omega_0^2 - \omega^2)^2 + \beta^2 \omega^2} \tag{5.53}$$

$$\langle V_p(t)^2 \rangle = \frac{\pi \sigma_p^2 G_0}{2 m^2 \beta \omega_0^2} = \sigma_p^2 \langle x(t)^2 \rangle \ . \tag{5.54}$$

The scaling parameters—G_0 and m—cannot be fit independently, so we condense the power spectrum of the right hand side of Eq. (5.49) into a single

$$G_1 \equiv \frac{\sigma_p^2 G_0}{m^2} \ . \tag{5.55}$$

This gives

$$\mathrm{PSD}(V_p, \omega) = \frac{G_1}{(\omega_0^2 - \omega^2)^2 + \beta^2\omega^2} \tag{5.56}$$

$$\left\langle V_p(t)^2 \right\rangle = \frac{\pi G_1}{2\beta\omega_0^2} = \sigma_p^2 \left\langle x(t)^2 \right\rangle \ . \tag{5.57}$$

Plugging into the equipartition theorem (Eq. (5.1)) yields

$$\kappa = \frac{\sigma_p^2 k_B T}{\langle V_p(t)^2 \rangle} = \frac{2\beta\omega_0^2\sigma_p^2 k_B T}{\pi G_1} \ . \tag{5.58}$$

Shifting this around, we can find the expected value of G_1.

$$G_1 = \frac{2\beta\omega_0^2\sigma_p^2 k_B T}{\pi\kappa} = \frac{2\beta\frac{\kappa}{m}\sigma_p^2 k_B T}{\pi\kappa} = \frac{2\beta\sigma_p^2 k_B T}{\pi m} \tag{5.59}$$

5.3.4 Fitting deflection voltage in frequency space

As another alternative, you could fit in frequency $f \equiv \omega/2\pi$ instead of angular frequency ω. The analysis will be the same, but we must be careful with normalization due to the different scales. Comparing the angular frequency and normal frequency unitary Fourier transforms

$$\mathcal{F}\left\{x(t)\right\}(\omega) \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \mathrm{d}t \tag{5.60}$$

$$\mathcal{F}_f\left\{x(t)\right\}(f) \equiv \int_{-\infty}^{\infty} x(t) e^{-2\pi i f t} \mathrm{d}t = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \mathrm{d}t = \sqrt{2\pi} \cdot \mathcal{F}\left\{x(t)\right\}(\omega = 2\pi f) \ , \tag{5.61}$$

from which we can translate the PSD

$$\mathrm{PSD}(x, \omega) \equiv \lim_{t_T \to \infty} \frac{1}{t_T} \ 2\left|\mathcal{F}\left\{x(t)\right\}(\omega)\right|^2 \tag{5.62}$$

$$\begin{aligned} \mathrm{PSD}_f(x, f) &\equiv \lim_{t_T \to \infty} \frac{1}{t_T} \ 2\left|\mathcal{F}_f\left\{x(t)\right\}(f)\right|^2 = 2\pi \cdot \lim_{t_T \to \infty} \frac{1}{t_T} \ 2\left|\mathcal{F}\left\{x(t)\right\}(\omega = 2\pi f)\right|^2 \\ &= 2\pi \, \mathrm{PSD}(x, \omega = 2\pi f) \ . \end{aligned} \tag{5.63}$$

The variance of the function $x(t)$ is then given by plugging into Eq. (5.21) (our corollary to Parseval's theorem)

$$\left\langle x(t)^2 \right\rangle = \int_0^\infty \mathrm{PSD}(x,\omega)\mathrm{d}\omega = \int_0^\infty \frac{1}{2\pi}\,\mathrm{PSD}_f(x,f)2\pi\cdot\mathrm{d}f = \int_0^\infty \mathrm{PSD}_f(x,f)\mathrm{d}f\ . \tag{5.64}$$

We can now extract Eqs. (5.3) and (5.4) from Eqs. (5.56) and (5.57).

$$\begin{aligned}\mathrm{PSD}_f(V_p,f) &= 2\pi\,\mathrm{PSD}(V_p,\omega) = \frac{2\pi G_1}{(4\pi^2 f_0^2 - 4\pi^2 f^2)^2 + \beta^2 4\pi^2 f^2} = \frac{2\pi G_1}{16\pi^4(f_0^2 - f^2)^2 + \beta^2 4\pi^2 f^2} \\ &= \frac{G_1/8\pi^3}{(f_0^2 - f^2)^2 + \frac{\beta^2 f^2}{4\pi^2}} = \frac{G_{1f}}{(f_0^2 - f^2)^2 + \beta_f^2 f^2}\end{aligned} \tag{5.65}$$

$$\left\langle V_p(t)^2 \right\rangle = \frac{\pi\frac{G_1}{(2\pi)^3}}{2\frac{\beta}{2\pi}\left(\frac{\omega_0}{2\pi}\right)^2} = \frac{\pi G_{1f}}{2\beta_f f_0^2}\ . \tag{5.66}$$

where $f_0 \equiv \omega_0/2\pi$, $\beta_f \equiv \beta/2\pi$, and $G_{1f} \equiv G_1/8\pi^3$. Finally, we can generate Eq. (5.5) from Eqs. (5.1) and (5.2).

$$\kappa = \frac{\sigma_p^2 k_B T}{\langle V_p(t)^2\rangle} = \frac{2\beta_f f_0^2 \sigma_p^2 k_B T}{\pi G_{1f}}\ . \tag{5.67}$$

Shifting this around, we can find the expected value of G_{1f}.

$$G_{1f} = \frac{2\beta_f f_0^2 \sigma_p^2 k_B T}{\pi\kappa} = \frac{2\beta_f \frac{\kappa}{4\pi^2 m}\sigma_p^2 k_B T}{\pi\kappa} = \frac{\beta_f \sigma_p^2 k_B T}{2\pi^3 m} \tag{5.68}$$

Plugging Eq. (5.68) into Eq. (5.3), we have

$$\mathrm{PSD}_f(V_p,f) = \frac{\sigma_p^2 k_B T \beta_f}{2\pi^3 m}\cdot\frac{1}{(f_0^2 - f^2)^2 + \beta_f^2 f^2} \tag{5.69}$$

From which we can recover Burnham et al.[26] Eq. (6).

$$\mathrm{PSD}_f(x,f) = \frac{\mathrm{PSD}_f(V_p,f)}{\sigma_p^2} = \frac{k_B T \beta_f}{2\pi^3 m} \cdot \frac{1}{(f_0^2 - f^2)^2 + \beta_f^2 f^2} \tag{5.70}$$

$$= \frac{k_B T f_0}{2\pi^3 m Q} \cdot \frac{1}{(f_0^2 - f^2)^2 + \frac{f_0^2 f^2}{Q^2}} = \frac{k_B T}{2\pi^3 m Q f_0^3} \cdot \frac{1}{(1 - \frac{f^2}{f_0^2})^2 + \frac{f^2}{f_0^2 Q^2}} \tag{5.71}$$

$$= \frac{k_B T}{2\pi^3 m Q \left(\frac{\omega_0}{2\pi}\right)^3} \cdot \frac{1}{(1 - \frac{f^2}{f_0^2})^2 + \frac{f^2}{f_0^2 Q^2}} = \frac{4 k_B T}{m Q \omega_0 \omega_0^2} \cdot \frac{1}{(1 - \frac{f^2}{f_0^2})^2 + \frac{f^2}{f_0^2 Q^2}} \tag{5.72}$$

$$= \frac{4 k_B T}{m Q \omega_0 \frac{\kappa}{m}} \cdot \frac{1}{(1 - \frac{f^2}{f_0^2})^2 + \frac{f^2}{f_0^2 Q^2}} = \frac{4 k_B T}{\omega_0 Q \kappa} \frac{1}{(1 - \frac{f^2}{f_0^2})^2 + \frac{f^2}{f_0^2 Q^2}} , \tag{5.73}$$

where Q is the quality factor[26]

$$Q \equiv \frac{\sqrt{\kappa m}}{\gamma} = \sqrt{\frac{\kappa}{m}} \frac{m}{\gamma} = \frac{\omega_0}{\beta} = \frac{2\pi f_0}{2\pi \beta_f} = \frac{f_0}{\beta_f} . \tag{5.74}$$

5.4 Calibcant

A calibration run based on Eq. (5.5) consists of bumping the surface with the cantilever tip to measure σ_p, measuring the buffer temperature T with a thermocouple, and measuring thermal vibration when the tip is far from the surface to extract the fit parameters G_{1f}, f_0, and β_f. I've written the calibcant package to carry out this calibration procedure, building on packages in the pyafm stack (Fig. 5.1).

5.4.1 Photodiode calibration

To calculate the photodiode sensitivity σ_p, we need surface bumps with a clearly delimited contact slope. The calibcant package uses `AFM.move_just_onto_surface` (Section 4.2.3) to position the cantilever tip a configurable distance off the surface (Section 4.3.1). Then calibcant uses the pypiezo component (Section 4.2.2) to ramp the tip towards the surface a configurable distance before returning the tip to its original position. The cantilever deflection during this approach–retract cycle is analyzed to measure σ_p (Fig. 5.2).

The retraction data is analyzed using a similar approach to pypiezo's surface detection algorithm

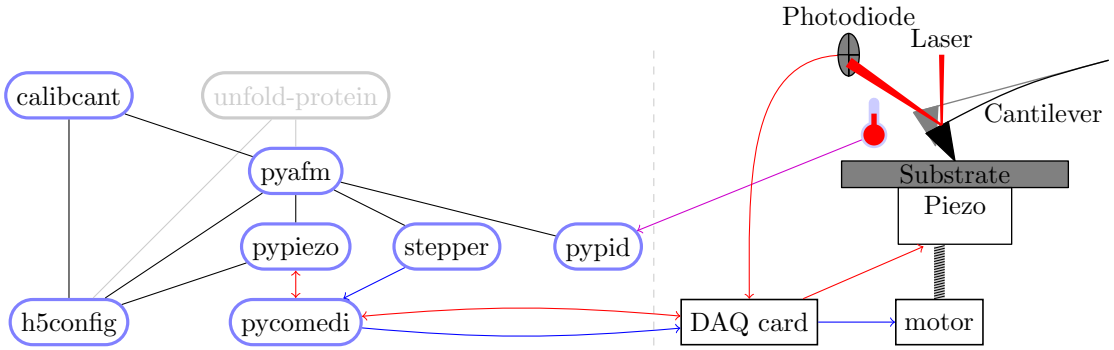


Figure 5.1: Dependency graph for calibcant, which shares the pyafm stack with unfold-protein(Fig. 4.3). The only difference is that the "brain" module controlling the stack has changed from unfold-protein to calibcant.

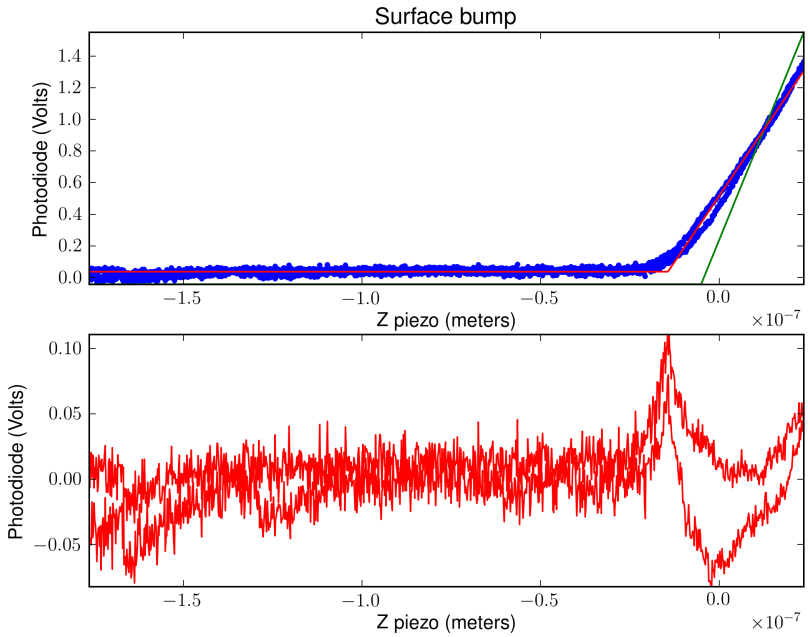


Figure 5.2: Measuring the photodiode sensitivity σ_p by bumping the cantilever tip on the substrate surface. In the first panel, the blue dots are experimental data, the green line is the heuristic guess at initial fitting parameters, and the red line is the optimized fit. The second panel shows the residual (measured data minus modeled data) for the bump. In both panels, there are two deflection measurements at each position, one taken during the approach phase and another taken during the retraction phase. This is the first bump from the 2013-02-07T08-20-46 calibration.

to extract the slope of the contact region. Where pypiezo uses a bilinear model (Eq. (4.1)), calibcant uses a limited linear model:

$$d(z) = \begin{cases} d_{\text{rail}} & z \leq z_{\text{rail}} \\ d_{\text{kink}} + \sigma_p(z - z_{\text{kink}}) & z_{\text{rail}} \leq z \leq z_{\text{kink}} \\ d_{\text{kink}} & z \geq z_{\text{kink}} \end{cases} \tag{5.75}$$

The fitted parameters are the surface contact point $(z_{\text{kink}}, d_{\text{kink}})$ and the contact slope σ_p. ADCs can only digitize voltages between the rails of their power supply, and the clipping deflection d_{rail} is the deflection ADC's maximum measureable voltage (2^{16} bits for our 16-bit ADCs).

By explicitly modeling the clipping deflection, we avoid the need for manual intervention when the configured approach distance is too large for the cantilever geometry and a bump pushes too hard. With short cantilevers, even small tip deflection distances can generate large laser deflection angles (Fig. 2.1a), leading to unmeasurable deflection voltages. One of the unfolding pulls in Fig. 4.10a exhibits this effect, although it was recorded using a different stack.

An alternative approach using sinusoidal piezo oscillation in the contact region has been proposed by Materassi et al. [157], on the grounds that it is more reliable and easily automated than an explicit bump and manual analysis. While I agree that *any* automated method is likely better than manual analysis, I feel that the difference between using an automated bump with a linear contact fit and using an automated oscillation with a linear contact fit is likely negligible.

5.4.2 Temperature measurements

After a series of surface bumps have been made to measure σ_p, the stepper motor is used to move the cantilever a configurable distance from the surface (generally ~ 30 μm). While the cantilever settles down after the jarring stepper motion, we measure the buffer temperature using pypid (Section 4.3.3), a Melcor Series MTCA Thermoelectric Cooler Controller [151], and a type E thermocouple[5]. The thermocouple is inserted through one of the ports in the AFM fluid cell, so the thermocouple tip is

[5]Part number 5TC-TT-E-30-72 from OMEGA Engineering Inc. [177]. Breaking down the product number, it's a five pack of thermocouples (5TC) with perfluoroalkoxy insulation (TT), type E metals (chromel–constantan), number 30 AWG wires (0.255 mm diameter), in a 72 inch length.

in the buffer less than 3 mm from the cantilever tip.

Equation (5.5) depends on the *absolute* temperature, so labs without easy access to a thermocouple can probably get away with estimating the buffer temperature. Errors of 5 K from an actual temperature of around 300 K will be within 1.7% of the actual value. The effect of this error on κ will be modest, but see Section 5.5.2 for a full discussion.

5.4.3 Thermal vibration

After the temperature measurements are complete, we measure the cantilever's thermal vibration without moving the piezo (Fig. 5.3). The parameters controlling these vibrations are configurable (with h5config, Section 4.3.1), but the default values are:

frequency The sampling frequency, which defaults to 50 kHz. This value gives a Nyquist frequency of 25 kHz, which is well above our resonant cantilever frequencies ($\sim$ 5 kHz in the buffer).

sample time The acquisition time in seconds. This is rounded up as required so the number of samples will be an integer power of two for efficient Fourier transformation. It defaults to 1 s.

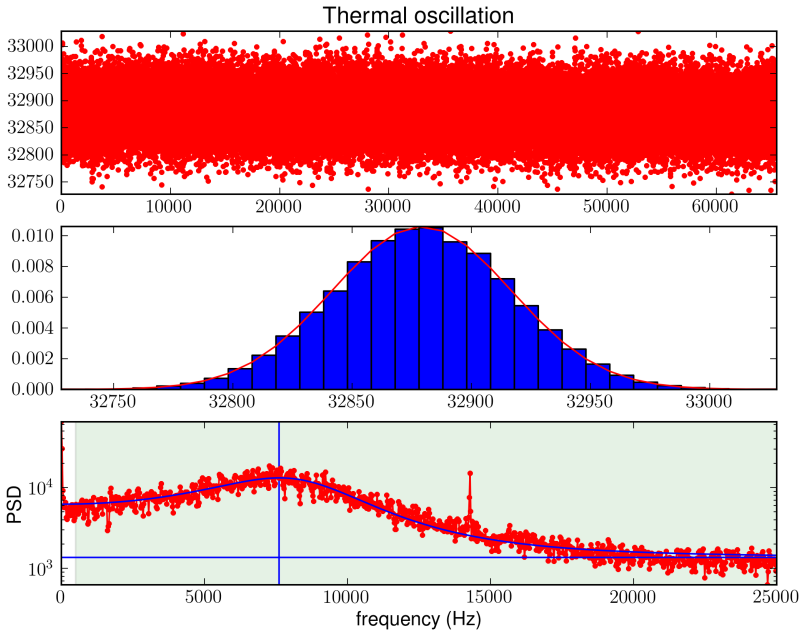
model The vibration model. This selects the fitting method for extracting the variance $\left\langle V_p(t)^2 \right\rangle$. By default, Eq. (5.3) is used, but you can add the constant offset (discussed below) or use the naïve $\left\langle V_p(t)^2 \right\rangle = \sum(V_p^2)/N$.

minimum fit frequency The low-frequency end of the PSD usually has a good deal of noise due to detector drift or background (non-cantilever) vibrations. This parameter allows you to select a window of the PSD for fitting that excludes the troublesome low-frequency region. It defaults to 500 Hz.

maximum fit frequency For completeness, you can also set a high-frequency cutoff, although I've never had to use this parameter.

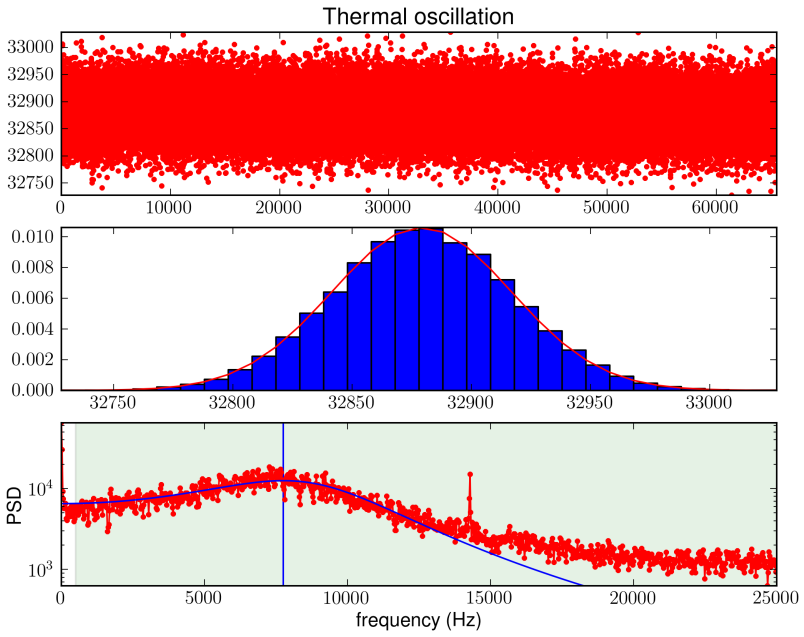
chunk-size, overlap, ... Assorted parameters for Fourier transforms used to compute the PSD.

Equation (5.3) decreases for large frequencies, but the measured PSD levels out (Fig. 5.3b). I attribute this to background white noise in the measurement circuit, and not due to cantilever



(a)

Figure 5.3: (a)Measuring the cantilever's thermal vibration. The top panel shows the raw time series data in bins, the middle panel shows the distribution of bin values with a Gaussian fit, and the bottom panel shows the $\mathrm{PSD}_f(V_p, f)$ with a fit following Eq. (5.76). The constant offset P_{0f}, drawn as the horizontal line in the third panel, accounts for white noise in the measurement circuit[26]. The vertical line marks the peak frequency $f_{\max}$ (Eq. (5.80)). Only data in the blue region was used when computing the best fit. This is the first vibration from the 2013-02-07T08-20-46 calibration, yielding a fitted variance $\left\langle V_p(t)^2 \right\rangle = 96.90 \pm 0.99\ \mathrm{mV}^2$. The narrow spike around 14.3 kHz is not due to the cantilever's thermal vibration, and rejecting noise like this is the reason we use a frequency-space fit to calculate the thermal deflection variance.



(b)

Figure 5.3: (b)This is the same data as in (a) fit with Eq. (5.3), yielding a fitted variance $\langle V_p(t)^2 \rangle = 120.92 \pm 0.90$ mV^2. The third panel is very similar to figure 2 in Florin et al. [27], but they do not go into further detail on the method or model. They may be fitting their data to Eq. (5.6), see Section 5.2. Another similar figure is in Hutter and Bechhoefer [28].

oscillation. To avoid artificially inflating the estimated $\langle V_p(t)^2 \rangle$, I created an alternative model for $\mathrm{PSD}_f(V_p, f)$ that adds a frequency-independent offset P_{0f} [26].

$$\mathrm{PSD}_f(V_p, f) = \frac{G_{1f}}{(f_0^2 - f^2)^2 + \beta_f^2 f^2} + P_{0f} \ . \tag{5.76}$$

Plots of Eq. (5.76) fits look better than Eq. (5.3) fits (Fig. 5.3), but the significance on the variance calculated with Eq. (5.4) depends on the amount of background noise in the vibration data. With over an order of magnitude difference between the power of the damped harmonic oscillator peak and the background noise, the effect of P_{0f} will be small. With noisier setups, removing the white background noise can lead to a significant difference. The fitted variance $\langle V_p(t)^2 \rangle$ of my 2013-02-07T08-20-46 data shifts from 120.92±0.90 mV^2 using Eq. (5.3) to 96.90±0.99 mV^2 using Eq. (5.76), a 20% decrease. The calculated spring constant increases from 43.3 ± 2.1 to 54.1 ± 2.7 mN/m, a 25% increase. Changes of this magnitude are important for accurate unfolding force calibration.

5.5 Discussion

5.5.1 Peak frequency

Since we went through the trouble of calculating the derivative of PSD_f in Eq. (5.10), it's useful to also calculate the frequency of the resonant peak.

$$0 = \frac{\mathrm{dPSD}_f}{\mathrm{d}f} = \frac{2G_{1f} f_{\max}}{\left((f_0^2 - f_{\max}^2)^2 + \beta_f^2 f_{\max}^2\right)^2} \left(2(f_0^2 - f_{\max}^2) - \beta_f^2\right) \tag{5.77}$$

$$= 2(f_0^2 - f_{\max}^2) - \beta_f^2 \tag{5.78}$$

$$f_{\max}^2 = f_0^2 - \frac{\beta_f^2}{2} \tag{5.79}$$

$$f_{\max} = \sqrt{f_0^2 - \frac{\beta_f^2}{2}} \ , \tag{5.80}$$

where we used $f \neq 0$ during the large simplifying multiplication. We see that the peak frequency is shifted from f_0 depending on the damping term β_f. For overdamped cantilevers with large values of β, the peak frequency will not have a real solution.

5.5.2 Propagation of errors

Extracting cantilever spring constants with Eq. (5.5) is great, but the number you get is not much good if you can't estimate its accuracy. We can find the effect of measurement and fitting errors on the calculated κ using Taylor expansions[178]. To the first order,

$$f(\mathbf{x}) \approx f_0 + \sum_i^n \left(\frac{\mathrm{d}f}{\mathrm{d}x_i}(x_i - x_{i0}) \right) . \tag{5.81}$$

To applying this to Eq. (5.5), we need the derivatives

$$\frac{\mathrm{d}\kappa}{\mathrm{d}\sigma_p} = \frac{\mathrm{d}}{\mathrm{d}\sigma_p}\left(\frac{\sigma_p^2 k_B T}{\langle V_p(t)^2 \rangle} \right) = \frac{2\kappa}{\sigma_p} \tag{5.82}$$

$$\frac{\mathrm{d}\kappa}{\mathrm{d}T} = \frac{\kappa}{T} \tag{5.83}$$

$$\frac{\mathrm{d}\kappa}{\mathrm{d}\langle V_p(t)^2 \rangle} = \frac{-\kappa}{\langle V_p(t)^2 \rangle} , \tag{5.84}$$

where I have used $\langle V_p(t)^2 \rangle$ directly to support alternative variance extraction models (Section 5.4.3).

Our measurements of σ_p, T, and $\langle V_p(t)^2 \rangle$ are independent and therefore uncorrelated. This lets us estimate standard deviation of κ from the standard deviation of the measured parameters[178].

$$\sigma_\kappa \approx \sqrt{\left(\frac{\mathrm{d}\kappa}{\mathrm{d}\sigma_p} \right)^2 \sigma_{\sigma_p}^2 + \left(\frac{\mathrm{d}\kappa}{\mathrm{d}T} \right)^2 \sigma_T^2 + \left(\frac{\mathrm{d}\kappa}{\mathrm{d}\langle V_p(t)^2 \rangle} \right)^2 \sigma_{\langle V_p(t)^2 \rangle}^2} \tag{5.85}$$

$$\approx \sqrt{\frac{4\kappa^2}{\sigma_p^2}\sigma_{\sigma_p}^2 + \frac{\kappa^2}{T^2}\sigma_T^2 + \frac{\kappa^2}{\langle V_p(t)^2 \rangle}\sigma_{\langle V_p(t)^2 \rangle}^2} \tag{5.86}$$

$$\frac{\sigma_\kappa}{\kappa} \approx \sqrt{4\left(\frac{\sigma_{\sigma_p}}{\sigma_p} \right)^2 + \left(\frac{\sigma_T}{T} \right)^2 + \left(\frac{\sigma_{\langle V_p(t)^2 \rangle}^2}{\langle V_p(t)^2 \rangle}^2 \right)} \tag{5.87}$$

By repeating each experiment (surface bumps, temperature readings, and thermal vibrations) several times, we can estimate the statistical uncertainty in each parameter (Fig. 5.4). Values for σ_p and $\langle V_p^2 \rangle$ are quite sensitive to the location of the laser spot on the cantilever, so they can vary over large time scales as the microscope alignment drifts (e.g. due to thermal expansion as the room

warms up). However, the calculated value for κ should not vary significantly.

For example, on a recent calibration run[6] I measured $\sigma_p = 35.68 \pm 0.87$ V/μm, $T = 298.151 \pm 0.033$ K, and $\left\langle V_p^2 \right\rangle = 96.90 \pm 0.99$ mV^2, which gives $\kappa = 54.1 \pm 2.7$ mN/m. These numbers are very similar to those obtained with a different cantilever from the same batch measured a month later (Table 5.1). The uncertainty contributions from each term are

$$4\left(\frac{\sigma_{\sigma_p}}{\sigma_p}\right)^2 = 2.38 \cdot 10^{-3}\ \mathrm{N}^2/\mathrm{m}^2 \tag{5.88}$$

$$\left(\frac{\sigma_T}{T}\right)^2 = 1.29 \cdot 10^{-8}\ \mathrm{N}^2/\mathrm{m}^2 \tag{5.89}$$

$$\left(\frac{\sigma^2_{\langle V_p(t)^2\rangle}}{\langle V_p(t)^2\rangle}^2\right) = 1.04 \cdot 10^{-4}\ \mathrm{N}^2/\mathrm{m}^2 \tag{5.90}$$

The size of the thermal vibration is $\sqrt{\left\langle V_p^2 \right\rangle}/\sigma_p \approx 2.8$ Å with forces on the order of $\kappa\sqrt{\left\langle V_p^2 \right\rangle}/\sigma_p \approx 15$ pN.

In this particular run, most of the uncertainty in κ comes from σ_{σ_p}, with some from $\sigma_{\langle V_p(t)^2\rangle}$. To add uncertainty comparable to the photodiode sensitivity contribution, the temperature variance would have to be

$$\sigma_T = \frac{\sigma_{\sigma_p}}{\sigma_p} \cdot T \approx \frac{2 \cdot 0.87}{35.68} \cdot 298.151\ \mathrm{T} \approx 15\ \mathrm{K}\ . \tag{5.91}$$

This is a large enough window that simply using room temperature (or even a hard-coded 300 K) should not introduce excessive error in the calculated κ.

5.5.3 Archiving experimental data

Scientific data is not thrown away after analysis. Organizations may have standards for archival, and many journals require supporting data to be available on request after publication or archived in public databases[179,180]. Both the raw data and the experimental parameters used to collect need to be preserved, but managing this manually is tedious and error prone. Lab notebooks rarely contain *all* of the parameters used to collect and analyze a particular calibration. Data collected with calibcant is saved in HDF5 with the full configuration (Section 4.3.1), bundling all of the

[6] 2013-02-07T08-20-46

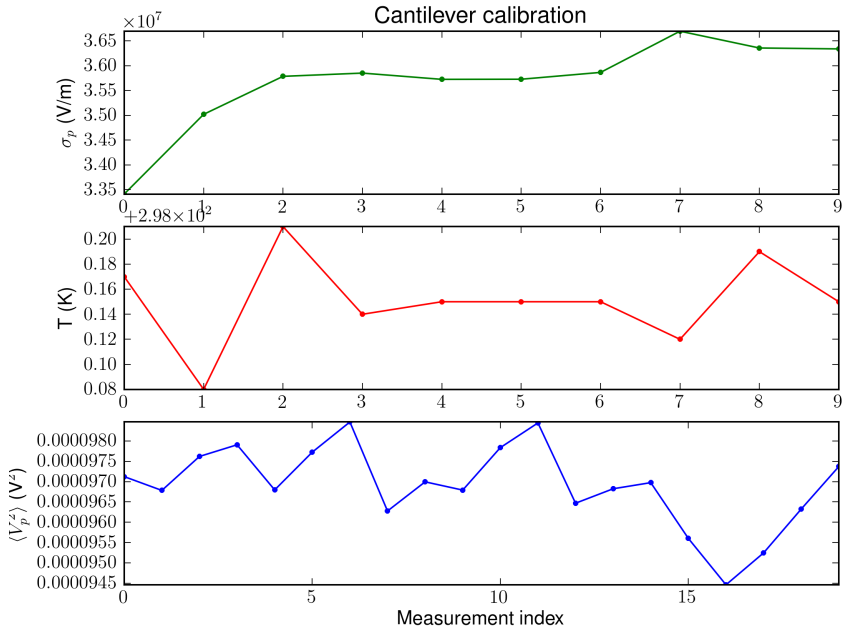


Figure 5.4: Estimating the statistical uncertainty of the calculated cantilever spring constant κ through repeated measurements.

Table 5.1: Measured spring constant calibration parameters (mean and standard deviation) for a single cantilever on two consecutive days. The measured parameters have changes slightly because the laser alignment and buffer temperature drift over time, but the measured κ are not significantly different ($p = 0.9$, as measured with a two-tailed Welch's t-test[7,8]).

Timestamp:		2013-03-03T16-37-12		2013-03-04T12-21-54	
Quantity	Units	Mean	Std. Dev.	Mean	Std. Dev.
σ_p	V/μm	46.22	0.76	41.30	0.21
T	K	296.302	0.021	294.272	0.022
$\langle V_p^2 \rangle$	mV^2	108.3	1.1	105.5	2.16
κ	mN/m	67.3	2.5	65.6	1.5

information together in a single file.

One minor drawback to this approach is that configuration information (which is not likely to change often) is duplicated between calibration runs. While this uses some extra disk space, the overhead is small. The full calibration datafile weighs in at 3.4 MB, while the calibration section alone is just 37 kB (1% of the total).

Besides the benefit of having a self contained file, HDF5 provides efficient support for large arrays of typed data (such as the unsigned 16-bit values from our DACs and ADCs), which is not possible with many other open file formats. The HDF libraries are supported by the non-profit HDF Group with a 20 year development history[181] and many users[182]. This suggests that HDF will be around for the long haul, and if it is eventually phased out, that there will be a number of well funded organizations interested in developing migration plans.

5.6 Conclusions

Thermal cantilever calibration has been common practice for many years now[27,28], starting with estimates of thermal noise[183]. However, discussion and explicit derivations have been limited. While this is likely done because the underlying theory is "obvious", it makes it more likely that corner cases slip by the notice of calibration experts[184] or incorrect formulæ are used during the fitting (Section 5.2). By centralizing calibration procedures in an open package, calibcant should both reduce the effort needed to calibrate AFM cantilevers, improve the quality of the calibration, and ease data sharing and archival.

Chapter 6: Data analysis with Hooke

Individual unfolding force curves acquired with the pyafm stack (Chapter 4) are hard to compare with other experimental data. One way to analyze them in bulk is to extract unfolding force histograms. These experimental histograms can then be used for fitting simulated models with sawsim (Chapter 3), and the model fitting parameters will summarize the kinetic behavior. In this chapter, I'll discuss the histogram extraction procedure and the Hooke package that facilitates it.

6.1 History

Hooke was originally developed by Sandal et al.[5] working at the University of Bologna. It was actively developed until the paper came out, after which development became more sporadic. This was partly because Hooke worked well enough for the original authors and partly because some of the developers had graduated and moved on to other fields[1].

Before discovering Hookein 2010, I had been using a series of fairly site-specific scripts to post-process my unfolding data. Excited by the existence of a published, open source post-processing framework, I dropped my scripts and started working on Hooke. Other open-source tools for post-processing SMFS data exist, but they are based on closed sorce tools[156,185] and some are no longer being developed[185]. There are also some completely closed source tools such as PUNIAS[187] and JPK's ForceRobot[189]. Other work along this line exists, but source code is unavailable[190].

Hooke supports a wide range of input file formats via *drivers* (Section 6.2), but when I began working on the project, there wasn't a clear interface between the drivers and processing *plugins* (Section 6.3). I cleaned up this interface as part of a general refactoring, fixing a number of plugins that relied on obscure internals in particular driver code. My refactoring removed these leaky

[1] Developer turnover may seem like a good reason to avoid open source software. Why use something when its developers may not stay around to support it? This argument may make sense if you're comparing open source and commercial packages, but it makes less sense if you're comparing existing open source packages to hypothetical in-house software. Why *not* use something, if it's free and already exists? Figuring out someone else's software is often much more efficient than writing your own tool from scratch[2].

[2] ... says the person who threw out the existing implementation and rewrote the control stack from scratch ;). I'm ok with starting over if the existing project is not maintainable, but realize that you're probably biting off a lot of work.

abstractions, and now every analysis plugin works with every data driver.

Hooke has been designed with an eye to modular flexibility, with a command line interface (CLI) and graphical user interface (GUI). However, as with drivers and plugins, the initial abstractions were leaky. I added a generic argument interface for analysis plugins, and taught the interfaces how to handle the generic argument types (Section 6.4). With this framework, new analysis plugins are automatically supported by both the CLI and the GUI. As a side effect of this reorganization, the GUI (which had previously been a display for the CLI) became truly interactive. The interactive portions of the GUI owe a large debt to earlier work by Rolf Schmidt et al. from the Centre for NanoScience Research at Concordia University.

6.2 Drivers for loading unfolding pull data

Hooke supports a number of different SMFS data formats, including Hemingway[157], JPK's ForceRobot, Asylum's MFP3D[191], Bruker's PicoForce, and my unfold-protein (Section 4.2.4) formats. The drivers are responsible for loading curve data into a standardized format so that plugins can work with data from any source. Drivers can determine if they are capable of reading a particular file, so if you need to analyze a directory full of curve files in a number of formats, you can just point Hooke at the directory and it will pick the appropriate driver for each curve.

After loading and parsing the data, drivers return a list of scaled *blocks* and a dictionary[3] of metadata. Each block corresponds to a different phase of the experiment; standard unfolding experiments have an approach block and a retraction block. The piezo position and cantilever deflection data in each block is scaled by the driver into meters, but further processing (e.g. the conversion of cantilever position to a chain tension in newtons) is carried out by plugins. The metadata dictionary includes standard keys for information that is required for the analysis (e.g. the calibrated spring constant in N/m). If the driver can parse any additional metadata from the file, it adds it to the dictionary using non-standard keys. You can use this auxilliary metadata to perform subsequent analysis (e.g. "give me all the curves that were recorded in `PBS + 0.5M CaCl2`").

[3] Python dictionaries are hash tables, which allow you to easily access arbitrary data if you know the key under which it was stored.

6.3 Plugins for analysis

Plugins are groups of related commands for processing curves. Curves can be stored in playlists, and there are builtin plugins for administrative tasks like managing curves (getting curve metadata, exporting blocks, ...) and playlists (moving to the next curve, globbing curves to the playlist, ...). There are also analysis plugins with commands for doing science. The `vclamp` plugin for velocity clamp analysis has commands for finding the surface contact point, scaling the cantilever deflection, removing the cantilever deflection from the total extension (Fig. 2.5), and flattening polynomial drift in the non-contact region. The `flatfilt` plugin has commands for identifying peaks based on spikes in the deflection derivative and for filtering curves from a playlist that only have more than a minimum threshold of such peaks[193]. The `polymer_fit` plugin has commands for fitting polymer models to the loading peaks (Section 3.2.1), which may have been identified using the `flatfilt` plugin or with any other peak-marking plugin. For other available plugins, see the Hooke documentation.

6.4 The user interface

Hookecommands are written with abstract argument definitions (using the cleverly named `Argument` class). This makes it easier to add new user interfaces (UIs), because the user interface is fundamentally about selecting commands and arguments to pass to them. In my work on Hooke, I borrowed from the graphical user interface (GUI) version from Concordia with the original partially-command-line-version from Bologna to produce two independent interfaces: a command line interface (CLI) and the GUI. You can do exactly the same things in either interface; choosing whichever is most convenient for the task at hand (Figs. 6.1 and 6.2). I usually use the CLI for scripting and routine tasks and reproducible analysis, but fire up the GUI when I'm exploring new data.

6.5 Conclusions

My Hooke work builds on previous open source development from other groups to provide a solid, extensible framework for processing raw force spectroscopy data. With this framework, it's easy to add analysis plugins to support new approaches to data analysis. It's also easy to add drivers to

```
$ hk.py
Hooke version 1.0.0.alpha (Ninken)
...
----
hooke> new_playlist --output_playlist mylist
<FilePlaylist mylist>
Success

hooke> glob_curves_to_playlist *.jpk
<Curve 2009.04.23-15.15.47.jpk>
<Curve 2009.04.23-15.21.39.jpk>
Success

hooke> curve_info
name: 2009.04.23-15.15.47.jpk
path: /.../hooke/test/data/vclamp_jpk/2009.04.23-15.15.47.jpk
driver: <hooke.driver.jpk.JPKDriver object at 0x28f9710>
note: None
command stack: []
blocks: 2
block names: ['approach', 'retract']
block sizes: [(4096, 6), (4096, 4)]
Success
```

Figure 6.1: Creating a playlist with two JPK files in the Hooke command line interface.

support new data file formats.

In it's current state, Hooke is used by a handful of people to analyze single molecule velocity clamp unfolding experiments. Even with Hooke helping out, the process is not automatic. Because of the scarcity of clean curves, it's hard to set agressive margins for automatically filtering clean unfolding sawteeth from curves that contain other interactions. Manually identifying clean curves from the rest is time consuming and subjective. By writing software to objectively identify unfolding events, we can improve the quality of the resulting science while at the same time increasing throughput. Real world data is messy, so developing objective filtering procedures is difficult. With Hooke's standardized framework and broad support for existing analysis procedures, it will be easy to apply new objective filtering procedures to old data, and see how well the new procedures match up with their partially automated predecessors.

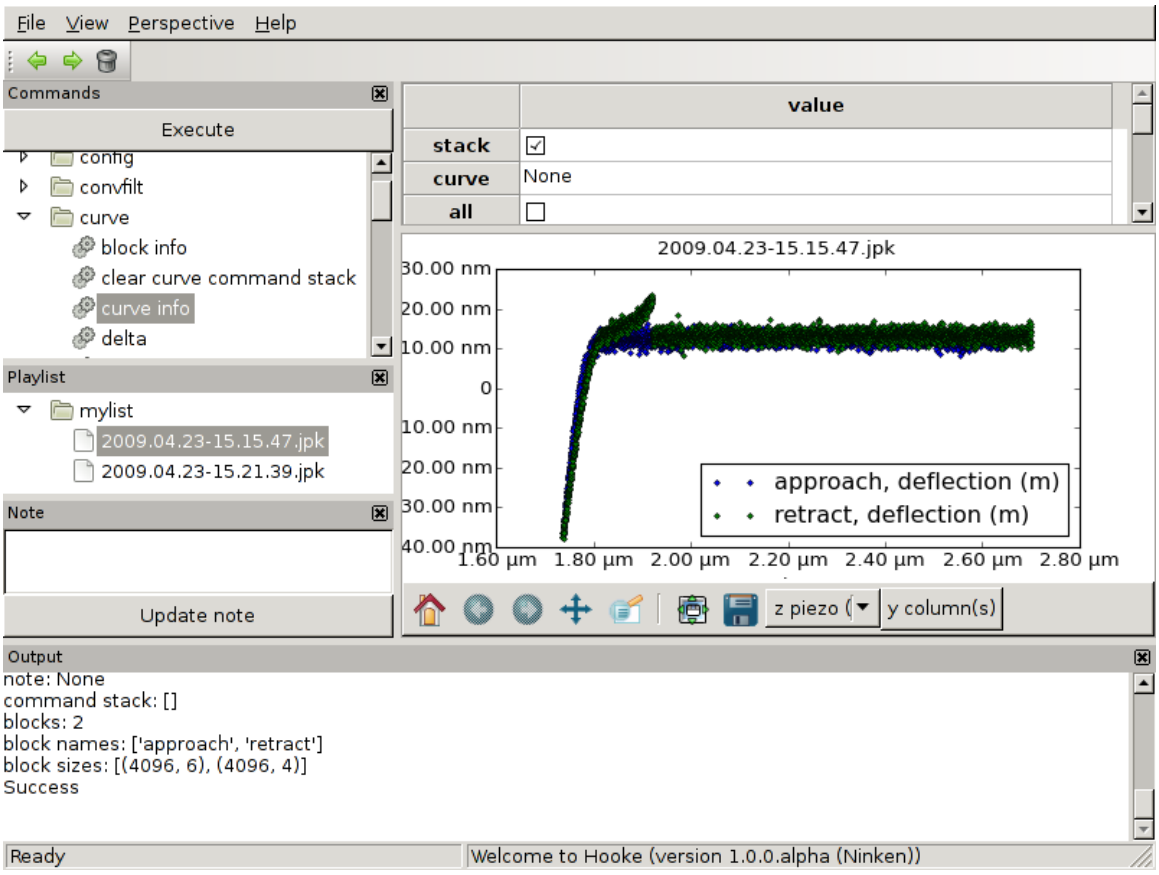


Figure 6.2: Creating a playlist with two JPK files in the graphical Hooke interface. You can see the output of the last *curve info* call, which matches the output from the command line version (Fig. 6.1). This screenshot is a bit cramped (to fit on a printed page), but the wxWidgets GUI toolkit provides automatic support for interactively rearranging and resizing panels. The tree of commands is in the upper left corner. After you select a command, the table of argument in the upper right corner is populated with default values, which you can adjust as you see fit. The *Playlist* panel provides an easier interface for navigating to different playlists and curves than the using *jump to playlist* and *jump to curve* commands.

Chapter 7: The effect of ions on unfolding force

With the tools in place, it's time to do some science! As a simple experiment to demonstrate the utility of the new stack, I ran a series of velocity clamp unfolding experiments on I27 octomers in PBS (Section 2.3). After a series of pulls in the standard buffer, a buffer with additional calcium (from $CaCl_2$[194]) was flushed into the fluid cell. After the new buffer equilibrated, unfolding experiments were continued.

Since Hofmeister[195], there has been a lot of research on the effect of ions on protein solubility. However, previous research on the effect of ions on unfolding forces is small, but we do have some material to work with. Earlier experimental work on amyloid β shows decreased fibrillation with even small Ca^{2+} concentrations[196,197]. The mechanism behind this weaker bonding is unclear[196,198], although aspartic and glutamic acid groups tend to have a strong affinity for cations while arginine has a strong affinity for the Cl^- anion[199]. Of these amino acids, only glutamic acid occurs in the key hydrogen bond regions responsible for I27 unfolding[29] (Fig. 7.1). NaCl has also been shown to decrease internal hydrogen bonding in the protein[200]. In molecular dynamics studies of amalyoid β fragments, the destabilizing effect of $CaCl_2$ is much greater than the destabilizing effects of $MgCl_2$, NaCl, and KCl[201].

We added 0.5 M $CaCl_2$ to our standard PBS (Section 2.3), which is much larger than extracellular Ca^{2+} levels on the order of 2 mM[197,202]. After mixing, both buffers were adjusted with drops of HCl and NaOH as needed to reach a pH around 7.5 (7.42 for the PBS, and 7.60 for the Ca^{2+}-enhanced PBS).

Unfolding experiments carried out on 2013-03-04 using our usual procedure (Section 2.4) yielded an unusual density of clean unfolding curves[1], with 105 successful pulls concentrated in a two hour window. Of these pulls, 37 were in the standard PBS and 68 were in the enhanced Ca^{2+} buffer. Histograms of unfolding forces show decreased unfolding forces in the Ca^{2+} buffer (Fig. 7.2), which

[1] Experiments carried out using the same procedures throughout February were much less successful.

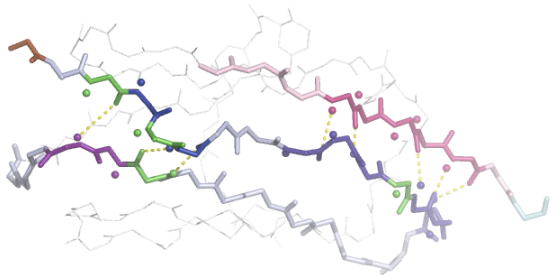


Figure 7.1: The backbone of I27 showing the eight key hydrogen bonds responsible for the critical unfolding force. Glutamic acids are highlighted in green. Based on Lu and Schulten [29] Fig. 1b. For a ribbon diagram of I27 showing the β-sheets, see Fig. 2.3. This figure was also generated with PyMol.

is what we expect due to destabilized hydrogen bonding.

Modeling I27 as a Bell-model unfolder, we can use sawsim to find the Bell parameters that best fit these experimental unfolding histograms (Sections 3.2.2 and 3.3.4). The results in Fig. 7.3 show that the best fit for standard PBS was with $\Delta x_u = 0.132$ nm and $k_{u0} = 0.222$ s^{-1}. In the Ca^{2+} buffer, the best fit was with $\Delta x_u = 0.123$ nm and $k_{u0} = 0.450$ s^{-1}.

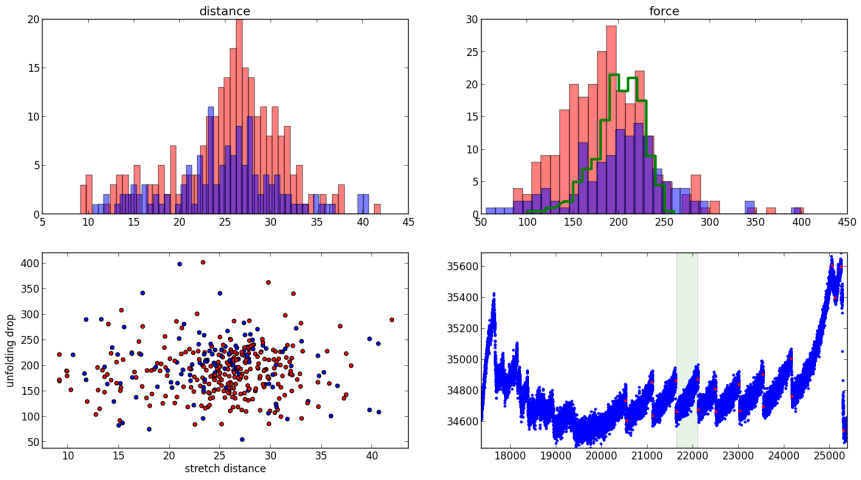


Figure 7.2: I27 runs from 2013-03-04 with (red) and without (blue) an extra 0.5 M Ca^{2+}. Clockwise from the upper left, we have the distance (in nm) between peaks, the unfolding force (in pN), and example force curve, and a scatter plot of unfolding force (in pN) versus the distance between peaks. All of the pulls were taken with the same Olympus TR400-PSA cantilever with a pulling speed of 1 μm/s. The green histogram drawn over the unfolding force histograms is I27 unfolding data in PBS with 5 mM DTT from Carrion-Vazquez et al. [6], rescaled by a factor of $\frac{1}{2}$ because they had more unfolding events.

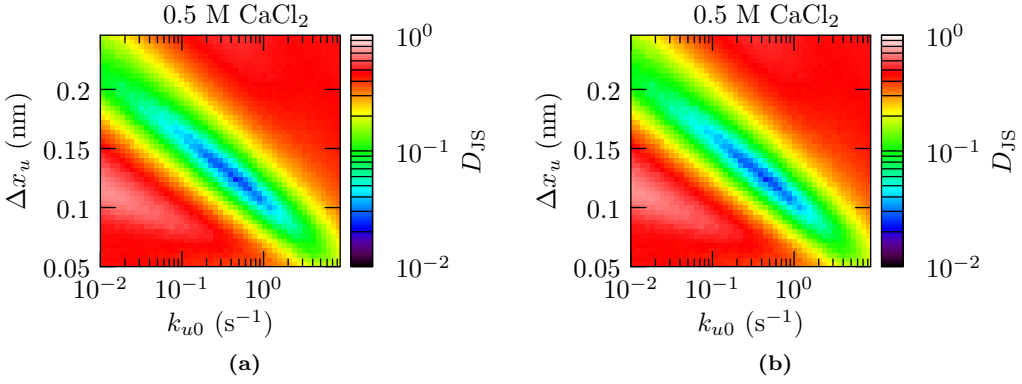


Figure 7.3: (a) Model fit quality for the standard PBS unfolding histogram data shown in Fig. 7.2. (b) Model fit quality for the Ca^{2+}-enhanced PBS unfolding histogram data. The best fit parameters occur when the Jensen–Shannon divergence is minimized (at the bottom of these valleys, Section 3.3.4).

Chapter 8: Conclusions and future work

Single molecule force spectroscopy (SMFS) provides an experimental window on the mechanics and kinetics of individual molecules and domains. Single molecule measurements extend bulk measurements, by offering insight into the variations within a population of molecules in addition to insight into the aggregate behavior of bulk solutions. They also bridge the gap between chemists experimenting at the bulk level and theorists simulating at the amino-acid level. By providing data for comparison, SMFS lays the ground work for improving and validating all-atom protein simulations. These simulations can then be used in predictive biological applications such as high throughput drug screening. By developing open source experiment control and analysis software, I have made it easier for new labs to get started in this field and for existing labs to collaborate on critical tools. I validate my experiment and analysis suite by carring out new experiments showing that increased $CaCl_2$ concentration significantly decreases the stability of folded I27.

8.1 Salt

As expected[196,197,200], increasing the ionic strength of the buffer did significantly decrease the unfolding force (folding stability) of I27. For labs with strong gene-splicing capability, it would be interesting to replace the glutamic acids involved in the major bonding (Fig. 7.1) with alternative groups to gauge the specificity of the effect. For example, glutamine is identical to glutamic acid, except that it has a hydroxyl group (OH) in the side-chain where glutamic acid has an amine group (NH_2). This gives glutamine and glutamic acid similar steric properties, but very different chemistry.

While the statistics are strong for the two concentrations we tested (standard PBS and PBS with an additional 0.5 M $CaCl_2$), it would be useful to study destabilization scaling over a range of concentrations. Carrying out these experiments over a range of pulling speeds with additional force clamp experiments would also increase confidence in the kinetic models used to summarize the data. SMFS is a low-throughput technique, so such an exponential increase in assembled data would be

much easier with more reliable hardware.

8.2 Hardware

Very few approach–bind–retract cycles actually pick up a protein and produce a clean unfolding curve. This makes it hard to gather sufficient unfolding statistics unless you can run experiments continuously over a long time span. While our modified MultiMode[203] gets the job done, some hardware upgrades would allow futher automation, increasing throughput through longer run times.

MultiModes measure the position of the cantilever by monitoring the reflected laser with a four-segment photodiode. While the vertical and horizontal signals are accessible in the cable connecting the MultiMode to its controlling NanoScope, the total photodiode signal is not. The loss of laser signal—which can occur when bubbles in the fluid cell obstruct the laser—results in low voltage deflection that is independent of piezo position. This flat-line deflection also occurs when mechanical drift moves the surface out of range for the piezo positioner. In the drift case, we would like to use the stepper motor to reduce the tip–surface separation. In the loss-of-signal case, we would like to *increase* the tip–surface separation to avoid accidentally crashing the tip into a surface we can no longer detect. By exposing the total photodiode signal to the control software, we could unambiguously distinguish these two cases. This would allow for longer runs, aggressively using the stepper motor to mitigate mechanical drift.

We could also reduce deflection signal noise—which is especially important for accurate cantilever calibration (Chapter 5)—by automating photodiode positioning. The four-segment photodiode has the least signal noise when the deflected laser lands near the point between all four sections. However, mechanical drift in microscope alignment causes the spot location to vary with time. We currently use manual thumbscrews to re-zero the photodiode as needed, but unmonitored overnight runs would require computer-controlled positioning. Similar automatic positioning would be useful for automatically aligning the incoming laser with the cantilever. While laser–cantilever alignment seems to be less sensitive than cantilever–photodiode alignment, automatic laser alignment would also open the door to automatic piezo calibration through measurements of laser interference patterns[204].

Finally, our current hardware does not address potential piezo hysteresis, nonlinearity, or drift.

Newer piezos often use capacitive feedback, adjusting the driving voltage as needed to maintain the target extension. Besides making existing distance measurements more accurate, the increased stability opens the door to slower pulling speeds needed to monitor proteins with less stable folded positions.

8.3 Software

Open source experiment control is possible! Even for a small lab, with a single novice developer[1], building reasonable software on top of existing pieces is possible. After a significant investment in developing sawsim (Chapter 3), the pyafm stack (Chapters 4 and 5), and Hooke (Chapter 6), we have a complete experiment control and analysis suite for single molecule force spectroscopy. All of the software in the stack—including the existing libraries and systems layer dependencies—is open source, so other labs are free to use, improve, and republish it as they see fit.

As the body of existing science increases, new researchers must become at the same time more specialized and more interdisciplinary than their fore-bearers. With a relatively fixed undergraduate curriculum, new researchers cannot afford to spend time becoming experts in every field that bears on their research project. By pooling resources between labs, individual researchers can reduce time spent on generic tooling and increase time spent on their particular project. Experiment control, analysis, and simulation software is particularly amenable to community development, because the cost of sharing software between labs is minimal.

Besides the low cost of transferring the data itself, the rise of distributed version control systems such as Git have reduced the administrative overhead of maintaining a project with many far-flung contributors. Researchers can automatically fetch and merge changes made by other groups, incorperating remote improvements. They can also commit and push local improvements, which are then available for remote researchers to incorperate. The version control systems and workflows that facilitate this cooperation scale well, from small projects with a single user, to huge projects like the Linux kernel with thousands of developers contributing to each release.

[1] I started this project with a bit of LabVIEW and Matlab experience, but only a few days of Python from the physics department's "Welcome to Drexel" boot camp. I stumbled across version control on my own, after a year of maintaining a directory full of version-stampted tarballs.

Once the software used in a lab has been published, it is also easier to audit by others who may be skeptical of the summary published in a journal article. For example, resolving the confusion about the "Lorentzian" (Section 5.2) would be trivial if Florin et al.[27] had also published their explicit procudure for generating their figure. Do you think I'm not calibrating my cantilevers correctly? Feel free to dig through my code. Let me know if you find something wrong (or fix it and send me a patch!). Science is built on reproducible experiments and analysis, and open source software allows you to explicitly specify your methods. With well organized code, the specification should be clear from high-level, experiment-design choices down through low-level bit manipulation.

Many researchers have not received formal training in software development best practices, so how do we bootstrap this transition to open source science? There is a wealth of documentation available online for self-teaching, and scientists have lots of experience reading technical writing in their own field. For those who are overwhelmed by the amount of available resources, organizations such as Software Carpentry are actively reaching out to scientists with short boot camps to lay the ground work. Mastery of any subject takes a significant investment, but gaining a working level of knowledge in distributed version control should only take a few days[2]. The difficulty for the uninitiated is often not mastering the new tool or workflow, but learning that it exists at all. There are a number of papers highlighting best practices and tools that are good surveys for guiding future learning[124,134,206,207].

[2] Software Carpentry allocates half a day to take students from "What is version control?" to being functioning Git users.

Bibliography

[1] Isaac Newton. *The correspondence of Isaac Newton*, volume 1. Published for the Royal Society at the University Press, 1959. URL `http://books.google.com/books?id=pr8WAQAAMAAJ`. The "Giants" quote is on page 416, in a letter to Robert Hooke dated February 5, 1676.

[2] William Trevor King, Meihong Su, and Guoliang Yang. Monte Carlo simulation of mechanical unfolding of proteins based on a simple two-state model. *International Journal of Biological Macromolecules*, 46(2):159–166, March 2010. ISSN 0141-8130. doi: 10.1016/j.ijbiomac.2009.12.001. URL `http://dx.doi.org/10.1016/j.ijbiomac.2009.12.001`. Sawsim is available at `http://blog.tremily.us/posts/sawsim/`.

[3] Nathan J. Mlot, Craig A. Tovey, and David L. Hu. Fire ants self-assemble into waterproof rafts to survive floods. *Proceedings of the National Academy of Sciences of the United States of America*, 108(19):7669–7673, May 2011. ISSN 1091-6490. doi: 10.1073/pnas.1016658108. URL `http://www.ncbi.nlm.nih.gov/pubmed/21518911`. Higher resolution pictures are available at `http://antlab.gatech.edu/antlab/The_Ant_Raft.html`.

[4] William Trevor King. Pyafm, 2013. URL `http://pypi.python.org/pypi/pyafm/`. Version 0.4+. I used commit 32bfbf9 (`afm: Optionally return stepper_approach data with 'record_data'`, 2013-01-18).

[5] Massimo Sandal, Fabrizio Benedetti, Marco Brucale, Alberto Gomez-Casado, and Bruno Samorì. Hooke: An open software platform for force spectroscopy. *Bioinformatics (Oxford, England)*, 25(11):1428–1430, June 2009. ISSN 1367-4811. doi: 10.1093/bioinformatics/btp180. URL `http://www.ncbi.nlm.nih.gov/pubmed/19336443`.

[6] Mariano Carrion-Vazquez, Andres F. Oberhauser, Susan B. Fowler, Piotr E. Marszalek, Sheldon E. Broedel, Jane Clarke, and Julio M. Fernandez. Mechanical and chemical unfolding of a single protein: A comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 96(7):3694–3699, 1999. doi: 10.1073/pnas.96.7.3694. URL `http://www.pnas.org/cgi/content/abstract/96/7/3694`.

[7] Bernard Lewis Welch. The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29(3-4):350–362, February 1938. ISSN 0006-3444. URL `http://www.jstor.org/stable/2332010`.

[8] Bernard Lewis Welch. The generalization of Student's problems when several different population variances are involved. *Biometrika*, 34(1-2):28–35, January 1947. ISSN 0006-3444. URL `http://www.ncbi.nlm.nih.gov/pubmed/20287819`.

[9] S. Freitag, I. Le Trong, L. Klumb, P. S. Stayton, and R. E. Stenkamp. Structural studies of the streptavidin binding loop. *Protein Science*, 6(6):1157–1166, June 1997. ISSN 0961-8368. doi: 10.1002/pro.5560060604. PDB ID: 1SWE, DOI: 10.2210/pdb1swe/pdb.

[10] Ronen Alon, Edward A. Bayer, and Meir Wilchek. Streptavidin contains an RYD sequence which mimics the RGD receptor domain of fibronectin. *Biochemical and Biophysical Research Communications*, 170(3):1236–1241, August 1990. ISSN 0006-291X. doi: 10.1016/0006-291X(90)90526-S. URL `http://dx.doi.org/10.1016/0006-291X(90)90526-S`. Biological role of streptavidin.

[11] W. L. DeLano. The PyMOL molecular graphics system, 2002. URL `http://www.pymol.org`. Version 1.2.1.

[12] Sabrina Bédard, Mallela M. G. Krishna, Leland Mayne, and S. Walter Englander. Protein folding: Independent unrelated pathways or predetermined pathway with optional errors. *Proceedings of the National Academy of Sciences of the United States of America*, 105(20):7182–7187, May 2008. ISSN 1091-6490. doi: 10.1073/pnas.0801864105. URL `http://www.pnas.org/content/105/20/7182.full`.

[13] K. A. Dill and H. S. Chan. From levinthal to pathways to funnels. *Nature Structural Biology*, 4(1):10–19, January 1997. ISSN 1072-8368. doi: 10.1038/nsb0197-10. URL `http://www.nature.com/nsmb/journal/v4/n1/abs/nsb0197-10.html`. Pretty folding funnel figures.

[14] Wikipedia. Skeletal muscle — Wikipedia, the free encyclopedia, 2012. URL `http://en.wikipedia.org/wiki/File:Skeletal_muscle.jpg`.

[15] S. Improta, A. S. Politou, and A. Pastore. Immunoglobulin-like modules from titin I-band: Extensible components of muscle elasticity. *Structure*, 4(3):323–337, March 1996. ISSN 0969-2126. doi: 10.1016/S0969-2126(96)00036-6. PDB ID: 1TIT, DOI: 10.2210/pdb1tit/pdb.

[16] Thomas Kempe, Stephen B. H. Kent, Flora Chow, Susan M. Peterson, Wesley I. Sundquist, James J. L'Italien, Douglas Harbrecht, David Plunkett, and William J. DeLorbe. Multiple-copy genes: Production and modification of monomeric peptides from large multimeric fusion proteins. *Gene*, 39(2-3):239–245, 1985. ISSN 0378-1119. URL `http://www.ncbi.nlm.nih.gov/pubmed/2419204`.

[17] Chia-Lin Chyan, Fan-Chi Lin, Haibo Peng, Jian-Min Yuan, Chung-Hung Chang, Sheng-Hsien Lin, and Guoliang Yang. Reversible mechanical unfolding of single ubiquitin molecules. *Biophysical Journal*, 87(6):3995–4006, December 2004. ISSN 0006-3495. doi: 10.1529/biophysj.104.042754. URL `http://www.cell.com/biophysj/abstract/S0006-3495(04)73864-3`. includes pH effects.

[18] Nicholas Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, 15: 125–130, 1987. URL `http://library.lanl.gov/cgi-bin/getfile?15-12.pdf`.

[19] Mariano Carrion-Vazquez, Hongbin Li, Hui Lu, Piotr E. Marszalek, Andres F. Oberhauser, and Julio M. Fernandez. The mechanical stability of ubiquitin is linkage dependent. *Nature Structural Biology*, 10(9):738–743, September 2003. ISSN 1072-8368. doi: 10.1038/nsb965. URL `http://www.nature.com/nsmb/journal/v10/n9/abs/nsb965.html`.

[20] William Trevor King. Unfold-protein, 2013. URL `http://pypi.python.org/pypi/unfold-protein/`. Version 0.2.

[21] William Trevor King. Calibcant, 2013. URL `http://pypi.python.org/pypi/calibcant/`. Version 0.8+. I used commit d9d9ac6 (`calibcant-plot.py: Plot the stepper approach curve`, 2013-01-18).

[22] William Trevor King. Stepper, 2012. URL `http://pypi.python.org/pypi/stepper/`. Version 0.4.

[23] William Trevor King. Pycomedi, 2013. URL `http://pypi.python.org/pypi/pycomedi/`. Version 0.6+. I used commit d07922f (`README: Double-backtick code for proper ReST markup`, 2013-01-17).

[24] William Trevor King. Pypiezo, 2012. URL `http://pypi.python.org/pypi/pyafm/`. Version 0.7+. I used commit 38c97d9 (`Don't plot missing surface detection regions in analyze_surface_position_data`, 2012-08-19.

[25] William Trevor King. Pypiezo, 2012. URL `http://pypi.python.org/pypi/pyafm/`. Version 0.4+. I used commit f9c020d (`.update-copyright.conf: update to pipe separators`, 2012-10-20).

[26] N. A. Burnham, Xinyong Chen, C. S. Hodges, G. A. Matei, E. J. Thoreson, C. J. Roberts, M. C. Davies, and S. J. B. Tendler. Comparison of calibration methods for atomic-force microscopy cantilevers. *Nanotechnology*, 14(1):1–6, January 2003. URL `http://stacks.iop.org/0957-4484/14/i=1/a=301`. Contains both the overdamped (Eq. (6)) and general (Eq. (8)) power spectral densities used in thermal cantilever calibration, but punts to textbooks for the derivation.

[27] Ernst-Ludwig Florin, Matthias Rief, H. Lehmann, Markus Ludwig, C. Dornmair, Vincent T. Moy, and Hermann E. Gaub. Sensing specific molecular interactions with the atomic force microscope. *Biosensors and Bioelectronics*, 10(9–10):895–901, 1995. ISSN 0956-5663. doi: 10.1016/0956-5663(95)99227-C. URL `http://dx.doi.org/10.1016/0956-5663(95)99227-C`. Good review of calibration to 1995, with experimental comparison between resonance-shift, reference-spring, and thermal methods. They incorrectly cite Hutter and Bechhoefer [28] as being published in 1994.

[28] Jeffrey L. Hutter and John Bechhoefer. Calibration of atomic-force microscope tips. *Review of Scientific Instruments*, 64(7):1868–1873, 1993. doi: 10.1063/1.1143970. URL `http://link.aip.org/link/?RSI/64/1868/1`. Original equipartition-based calibration method (thermal calibration), after the brief mention in Howard and Hudspeth [167]. This is the first paper I've found that works out the theory in detail, although they punt to page 431 of Heer [208] instead of listing a formula for their "Lorentzian". The experimental data uses high-Q cantilevers in air, and their figure 2 shows clear water-layer snap-off. There is a published erratum [184].

[29] Hui Lu and Klaus Schulten. The key event in force-induced unfolding of titin's immunoglobulin domains. *Biophysical Journal*, 79(1):51–65, July 2000. ISSN 0006-3495. doi: 10.1016/S0006-3495(00)76273-4. URL `http://www.cell.com/biophysj/abstract/S0006-3495%2800%2976273-4`.

[30] Matthias Rief, Mathias Gautel, Filipp Oesterhelt, Julio M. Fernandez, and Hermann E. Gaub. Reversible unfolding of individual titin immunoglobulin domains by AFM. *Science*, 276(5315):1109–1112, 1997. doi: 10.1126/science.276.5315.1109. URL `http://www.sciencemag.org/cgi/content/abstract/276/5315/1109`. Seminal paper for force spectroscopy on Titin. Cited by Dietz and Rief [209] (ref 9) as an example of how unfolding large proteins is easily interpreted (vs. confusing unfolding in bulk), but Titin is a rather simple example of that, because of its globular-chain structure.

[31] David Schleef, Frank Mori Hess, Herman Bruyninckx, Bernd Porr, and Ian Abbott. The control and measurement device interface handbook for comedilib 0.10.0, 2012. URL `http://www.comedi.org/doc/`.

[32] T. G. Wolfsberg, J. McEntyre, and G. D. Schuler. Guide to the draft human genome. *Nature*, 409(6822):824–826, February 2001. ISSN 0028-0836. doi: 10.1038/35057000. URL `http://www.nature.com/nature/journal/v409/n6822/full/409824a0.html`.

[33] J. D. McPherson, M. Marra, L. Hillier, R. H. Waterston, A. Chinwalla, J. Wallis, M. Sekhon, K. Wylie, E. R. Mardis, R. K. Wilson, R. Fulton, T. A. Kucaba, C. Wagner-McPherson, W. B. Barbazuk, S. G. Gregory, S. J. Humphray, L. French, R. S. Evans, G. Bethel, A. Whittaker, J. L. Holden, O. T. McCann, A. Dunham, C. Soderlund, C. E. Scott, D. R. Bentley, G. Schuler, H. C. Chen, W. Jang, E. D. Green, J. R. Idol, V. V. Maduro, K. T. Montgomery, E. Lee, A. Miller, S. Emerling, Kucherlapati, R. Gibbs, S. Scherer, J. H. Gorrell, E. Sodergren, K. Clerc-Blankenburg, P. Tabor, S. Naylor, D. Garcia, P. J. de Jong, J. J. Catanese, N. Nowak, K. Osoegawa, S. Qin, L. Rowen, A. Madan, M. Dors, L. Hood, B. Trask, C. Friedman, H. Massa, V. G. Cheung, I. R. Kirsch, T. Reid, R. Yonescu, J. Weissenbach, T. Bruls, R. Heilig, E. Branscomb, A. Olsen, N. Doggett, J. F. Cheng, T. Hawkins, R. M. Myers, J. Shang, L. Ramirez, J. Schmutz, O. Velasquez, K. Dixon, N. E. Stone, D. R. Cox, D. Haussler, W. J. Kent, T. Furey, S. Rogic, S. Kennedy, S. Jones, A. Rosenthal, G. Wen, M. Schilhabel, G. Gloeckner, G. Nyakatura, R. Siebert, B. Schlegelberger, J. Korenberg, X. N.

Chen, A. Fujiyama, M. Hattori, A. Toyoda, T. Yada, H. S. Park, Y. Sakaki, N. Shimizu, S. Asakawa, K. Kawasaki, T. Sasaki, A. Shintani, A. Shimizu, K. Shibuya, J. Kudoh, S. Minoshima, J. Ramser, P. Seranski, C. Hoff, A. Poustka, R. Reinhardt, and H. Lehrach. A physical map of the human genome. *Nature*, 409(6822):934–941, February 2001. ISSN 0028-0836. doi: 10.1038/35057157. URL `http://www.nature.com/nature/journal/v409/n6822/full/409934a0.html`.

[34] Francis S. Collins, Michael Morgan, and Aristides Patrinos. The human genome project: Lessons from large-scale biology. *Science*, 300(5617):286–290, April 2003. ISSN 1095-9203. doi: 10.1126/science.1084564. URL `http://www.sciencemag.org/cgi/content/summary/300/5617/277`. See also: Landmark HPG Papers.

[35] J. M. Claverie. Gene number. what if there are only 30,000 human genes? *Science*, 291 (5507):1255–1257, February 2001. ISSN 0036-8075. URL `http://www.sciencemag.org/cgi/content/full/291/5507/1255`.

[36] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. Di Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R. R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Wang, A. Wang, X. Wang, J. Wang, M. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y. H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigó, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y. H. Chiang, M. Coyne, C. Dahlke, A. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen, D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu. The sequence of the human genome. *Science*, 291(5507):1304–1351, Feb 2001. ISSN 0036-8075. doi: 10.1126/science.1058040. URL `http://www.sciencemag.org/cgi/content/short/291/5507/1304`.

[37] Cyrus Levinthal. How to fold graciously. In P. Debrunner, J.C.M. Tsibris, and E. Münck, editors, *Mossbauer Spectroscopy in Biological Systems*, pages 22–24, Allerton House, Monti-

cello, IL, 1969. University of Illinois Press, Urbana. URL http://www-miller.ch.cam.ac.uk/levinthal/levinthal.html.

[38] Carlos Bustamante. In singulo biochemistry: When less is more. *Annual Review of Biochemistry*, 77:45–50, 2008. ISSN 0066-4154. doi: 10.1146/annurev.biochem.012108.120952. URL http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.biochem.012108.120952.

[39] Hui Lu, B. Isralewitz, Andre Krammer, Viola Vogel, and Klaus Schulten. Unfolding of titin immunoglobulin domains by steered molecular dynamics simulation. *Biophysical Journal*, 75(2):662–671, August 1998. ISSN 0006-3495. doi: 10.1016/S0006-3495(98)77556-3. URL http://www.cell.com/biophysj/abstract/S0006-3495(98)77556-3.

[40] Hui Lu and Klaus Schulten. Steered molecular dynamics simulations of force-induced protein domain unfolding. *Proteins*, 35(4):453–463, June 1999. ISSN 0887-3585. doi: 10.1002/(SICI)1097-0134(19990601)35:4⟨453::AID-PROT9⟩3.0.CO;2-M. URL http://www3.interscience.wiley.com/journal/65000328/abstract.

[41] Matthias Rief and Helmut Grubmüller. Force spectroscopy of single biomolecules. *Chemphyschem*, 3(3):255–261, March 2002. ISSN 1439-4235. doi: 10.1002/1439-7641(20020315)3:3⟨255::AID-CPHC255⟩3.0.CO;2-M. URL http://www3.interscience.wiley.com/journal/91016383/abstract. Nice, general review of force spectroscopy to 2002, but not much detail.

[42] Jason Ming Zhao, Haeshin Lee, Rene A. Nome, Sophia Majid, Norbert F. Scherer, and Wouter D. Hoff. Single-molecule detection of structural changes during Per-Arnt-Sim (PAS) domain activation. *Proceedings of the National Academy of Sciences of the United States of America*, 103(31):11561–11566, 2006. doi: 10.1073/pnas.0601567103. URL http://www.pnas.org/cgi/content/abstract/103/31/11561.

[43] Felix Berkemeier, Morten Bertz, Senbo Xiao, Nikos Pinotsis, Matthias Wilmanns, Frauke Gräter, and Matthias Rief. Fast-folding α-helices as reversible strain absorbers in the muscle protein myomesin. *Proceedings of the National Academy of Sciences of the United States of America*, 108(34):14139–14144, August 2011. ISSN 1091-6490. doi: 10.1073/pnas.1105734108. URL http://www.ncbi.nlm.nih.gov/pubmed/21825161.

[44] Miklós S. Z. Kellermayer, S. B. Smith, Henk L. Granzier, and Carlos Bustamante. Folding-unfolding transitions in single titin molecules characterized with laser tweezers. *Science*, 276 (5315):1112–1116, May 1997. ISSN 0036-8075.

[45] Nancy R. Forde, David Izhaky, Glenna R. Woodcock, Gijs J. L. Wuite, and Carlos Bustamante. Using mechanical force to probe the mechanism of pausing and arrest during continuous elongation by escherichia coli RNA polymerase. *Proceedings of the National Academy of Sciences of the United States of America*, 99(18):11682–11687, September 2002. ISSN 0027-8424. doi: 10.1073/pnas.142417799. URL http://www.pnas.org/content/99/18/11682.

[46] S. B. Smith, L. Finzi, and Carlos Bustamante. Direct mechanical measurements of the elasticity of single DNA molecules by using magnetic beads. *Science*, 258(5085):1122–1126, November 1992. ISSN 0036-8075. doi: 10.1126/science.1439819. URL http://www.sciencemag.org/cgi/content/abstract/258/5085/1122.

[47] R. Merkel, P. Nassoy, A. Leung, K. Ritchie, and E. Evans. Energy landscapes of receptor-ligand bonds explored with dynamic force spectroscopy. *Nature*, 397(6714):50–53, January 1999. ISSN 0028-0836. doi: 10.1038/16219. URL http://www.nature.com/nature/journal/v397/n6714/full/397050a0.html.

[48] Ken Halvorsen and Wesley P. Wong. Massively parallel single-molecule manipulation using centrifugal force. *arXiv*, 2009. URL http://arxiv.org/abs/0912.5370.

[49] H. Jane Dyson and Peter E. Wright. Intrinsically unstructured proteins and their functions. *Nature Reviews Molecular Cell Biology*, 6(3):197–208, March 2005. ISSN 1471-0072. doi: 10.1038/nrm1589. URL http://www.ncbi.nlm.nih.gov/pubmed/15738986.

[50] Mariano Carrion-Vazquez, Piotr E. Marszalek, Andres F. Oberhauser, and Julio M. Fernandez. Atomic force microscopy captures length phenotypes in single proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 96(20):11288–11292, 1999. doi: 10.1073/pnas.96.20.11288. URL http://www.pnas.org/cgi/content/abstract/96/20/11288.

[51] Matthias Rief, H. Clausen-Schaumann, and Hermann E. Gaub. Sequence-dependent mechanics of single DNA molecules. *Nature Structural Biology*, 6(4):346–349, April 1999. ISSN 1072-8368. doi: 10.1038/7582. URL http://www.nature.com/nsmb/journal/v6/n4/abs/nsb0499_346.html.

[52] Gerd Binnig, Calvin F. Quate, and Christoph Gerber. Atomic force microscope. *Physical Review Letters*, 56(9):930–933, March 1986. ISSN 1079-7114. doi: 10.1103/PhysRevLett.56.930. URL http://www.ncbi.nlm.nih.gov/pubmed/10033323. Original AFM paper.

[53] Gerhard Meyer and Nabil M. Amer. Novel optical approach to atomic force microscopy. *Applied Physics Letters*, 53(12):1045–1047, September 1988. ISSN 0003-6951. doi: 10.1063/1.100061. URL http://apl.aip.org/resource/1/applab/v53/i12/p1045_s1.

[54] R Lévy and Mounir Maaloum. Measuring the spring constant of atomic force microscope cantilevers: Thermal fluctuations and other methods. *Nanotechnology*, 13(1):33–37, 2002. doi: 10.1088/0957-4484/13/1/307. URL http://stacks.iop.org/0957-4484/13/33. Good review of thermal calibration to 2002, but not much on the derviation of the Lorentzian fit.

[55] B. Drake, C. B. Prater, A. L. Weisenhorn, S. A. Gould, T. R. Albreacht, Calvin F. Quate, D. S. Cannell, H. G. Hansma, and Paul K. Hansma. Imaging crystals, polymers, and processes in water with the atomic force microscope. *Science*, 243(4898):1586–1589, March 1989. doi: 10.1126/science.2928794. URL http://www.sciencemag.org/content/243/4898/1586.abstract.

[56] M. Radmacher, R. W. Tillmann, M. Fritz, and Hermann E. Gaub. From molecules to cells: imaging soft samples with the atomic force microscope. *Science*, 257(5078):1900–1905, September 1992. doi: 10.1126/science.1411505. URL http://www.sciencemag.org/content/257/5078/1900.abstract.

[57] Carlos Bustamante, John F. Marko, Eric D. Siggia, and S. Smith. Entropic elasticity of lambda-phage DNA. *Science*, 265(5178):1599–1600, September 1994. ISSN 0036-8075. doi: 10.1126/science.8079175. URL http://www.sciencemag.org/cgi/content/abstract/265/5178/1599. WLC interpolation formula.

[58] Siegfried Labeit and B. Kolmerer. Titins: Giant proteins in charge of muscle ultrastructure and elasticity. *Science*, 270(5234):293–296, October 1995. ISSN 0036-8075. URL http://www.ncbi.nlm.nih.gov/pubmed/7569978.

[59] Olympus. Small spring constant silicon nitride cantilevers, 2012. URL http://probe.olympus-global.com/en/product/omcl_tr400psa_hw/. OMCL-TR400PSA. From Asylum Research, the small cantilever has a spring constant $\kappa = 0.08(0.02 - 0.23)$ N/m, a length of $100(90 - 110)$ μm, a width of $15(14 - 16)$ μm, and a thickness of $0.4(0.3 - 0.5)$ μm. The tip is a four-sided pyramid with a height of 3.0 ± 0.5 μm, a side angle of $35 \pm 1°$, and a tip radius of 20 ± 5 nm. The cantilevers are silicon nitride, with chromium and gold reflex coatings that are 5 and 30 nm thick respectively. From Olympus, the resonant frequency of the small cantilever in water is around 7 kHz.

[60] AthenaES. I27O™ AFM reference protein, 2012. URL http://www.athenaes.com/I27OAFMReferenceProtein.php.

[61] Frank Wilco Bartels, Birgit Baumgarth, Dario Anselmetti, Robert Ros, and Anke Becker. Specific binding of the regulatory protein expG to promoter regions of the galactoglucan biosynthesis gene cluster of sinorhizobium meliloti–a combined molecular biology and force spectroscopy investigation. *Journal of Structural Biology*, 143(2):145–152, August 2003. ISSN 1047-8477. URL http://www.ncbi.nlm.nih.gov/pubmed/12972351.

[62] Liang Ma. *The Nanomechanics of Polycystin-1: A Kidney Mechanosensor*. PhD thesis, University of Texas Medical Branch, August 2010. URL http://etd.utmb.edu/theses/available/etd-07072010-132038/.

[63] Agilent Technologies. Sure 2 supercompetent cells, 2012. URL http://www.chem.agilent.com/en-US/Search/Library/_layouts/Agilent/PublicationSummary.aspx?whid=72739&liid=6524. Catalog #200152.

[64] Mariano Carrion-Vazquez, Andres F. Oberhauser, T. E. Fisher, Piotr E. Marszalek, Hongbin Li, and Julio M. Fernandez. Mechanical design of proteins studied by single-molecule force spectroscopy and protein engineering. *Progress in Biophysics and Molecular Biology*, 74(1-2):63–91, 2000. ISSN 0079-6107. doi: 10.1016/S0079-6107(00)00017-1. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1302160. Surface contact Fig. 2 is a modified version of Baljon and Robbins [210] Fig. 1. They are both good pictures for explaining that the tip's radius of curvature ($\sim$ 20 nm) is larger than the I27 domainsImprota et al. [15] ($\sim$ 2 nm).

[65] Abraham Ulman. Formation and structure of self-assembled monolayers. *Chemical reviews*, 96 (4):1533–1554, June 1996. ISSN 1520-6890. URL http://www.ncbi.nlm.nih.gov/pubmed/11848802.

[66] Yu-Shiu Lo, Ying-Jie Zhu, and Thomas P. Beebe. Loading-rate dependence of individual ligandreceptor bond-rupture forces studied by atomic force microscopy. *Langmuir*, 17(12):3741–3748, 2001. doi: 10.1021/la001569g. URL http://pubs.acs.org/doi/abs/10.1021/la001569g. These guys seem to be pretty thorough, give this one another read.

[67] David J. Brockwell, Godfrey S. Beddard, John Clarkson, Rebecca C. Zinober, Anthony W. Blake, John Trinick, Peter D. Olmsted, D. Alastair Smith, and Sheena E. Radford. The effect of core destabilization on the mechanical resistance of I27. *Biophysical Journal*, 83 (1):458–472, July 2002. ISSN 0006-3495. doi: 10.1016/S0006-3495(02)75182-5. URL http://www.biophysj.org/cgi/content/abstract/83/1/458.

[68] Miklós S. Z. Kellermayer, Carlos Bustamante, and Henk L. Granzier. Mechanics and structure of titin oligomers explored with atomic force microscopy. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, 1604(2):105–114, 2003. ISSN 0005-2728. doi: 10.1016/S0005-2728(03)00029-X. URL http://dx.doi.org/10.1016/S0005-2728(03)00029-X.

[69] Jacob J. Schmidt, Xingqun Jiang, and Carlo D. Montemagno. Force tolerances of hybrid nanodevices. *Nano letters*, 2(11):1229–1233, 2002. doi: 10.1021/nl025773v. URL http://pubs.acs.org/doi/abs/10.1021/nl025773v.

[70] Hiroyasu Itoh, Akiri Takahashi, Kengo Adachi, Hiroyuki Noji, Ryohei Yaduso, Masasuke Yoshida, and Kazuhiko Kinosita Jr. Mechanically driven ATP synthesis by F1-ATPase. *Nature*, 427(6973):465–468, January 2004. ISSN 1476-4687. doi: 10.1038/nature02212. URL http://www.ncbi.nlm.nih.gov/pubmed/14749837.

[71] Naoyoshi Sakaki, Rieko Shimo-Kon, Kengo Adachi, Hiroyasu Itoh, Shou Furuike, Eiro Muneyuki, Masasuke Yoshida, and Kazuhiko Kinosita Jr. One rotary mechanism for F1-ATPase over ATP concentrations from millimolar down to nanomolar. *Biophysical Journal*,

88(3):2047–2056, March 2005. ISSN 0006-3495. doi: 10.1529/biophysj.104.054668. URL http://www.ncbi.nlm.nih.gov/pubmed/15626703.

[72] Mark Sundberg, Jenny P. Rosengren, Richard Bunk, Joakim Lindahl, Ian A. Nicholls, Sven Tågerud, Pär Omlink, Lars Montelius, and Alf Månsson. Silanized surfaces for in vitro studies of actomyosin function and nanotechnology applications. *Analytical biochemistry*, 323(1):127–138, December 2003. ISSN 0003-2697. doi: 10.1016/j.ab.2003.07.022. URL http://www.ncbi.nlm.nih.gov/pubmed/14622967.

[73] Sunyoung Lee. Chemical functionalization of afm cantilevers. Master's thesis, Massachusetts Institute of Technology, September 2005. URL http://dspace.mit.edu/handle/1721.1/34205. Binding proteins to gold-coated cantilevers via EDC (among other things in this thesis.

[74] Lutz Schmitt, Markus Ludwig, Hermann E. Gaub, and Robert Tampé. A metal-chelating microscopy tip as a new toolbox for single-molecule experiments by atomic force microscopy. *Biophysical Journal*, 78(6):3275–3285, June 2000. ISSN 0006-3495. doi: 10.1016/S0006-3495(00)76863-9. URL http://www.ncbi.nlm.nih.gov/pubmed/10828003.

[75] Dmitri K. Klimov and Devarajan Thirumalai. Stretching single-domain proteins: Phase diagram and kinetics of force-induced unfolding. *Proceedings of the National Academy of Sciences of the United States of America*, 96(11):6166–6170, May 1999. ISSN 0027-8424.

[76] N. D. Socci, J. N. Onuchic, and P. G. Wolynes. Stretching lattice models of protein folding. *Proceedings of the National Academy of Sciences of the United States of America*, 96(5):2031–2035, March 1999. ISSN 0027-8424.

[77] Dmitri K. Klimov and Devarajan Thirumalai. Native topology determines force-induced unfolding pathways in globular proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 97(13):7254–7259, June 2000. ISSN 0027-8424. doi: 10.1073/pnas.97.13.7254. URL http://www.pnas.org/cgi/content/abstract/97/13/7254. Simulated unfolding time scales for Ig27-like S1 and S2 domains.

[78] Hongbin Li, Andres F. Oberhauser, Sambra D. Redick, Mariano Carrion-Vazquez, Harold P. Erickson, and Julio M. Fernandez. Multiple conformations of PEVK proteins detected by single- molecule techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 98(19):10682–10686, 2001. doi: 10.1073/pnas.191189098. URL http://www.pnas.org/cgi/content/abstract/98/19/10682.

[79] Matthias Rief, Filipp Oesterhelt, Berthold Heymann, and Hermann E. Gaub. Single molecule force spectroscopy on polysaccharides by atomic force microscopy. *Science*, 275(5304):1295–1297, February 1997. ISSN 1095-9203. doi: 10.1126/science.275.5304.1295. URL http://www.sciencemag.org/cgi/content/abstract/275/5304/1295.

[80] Matthias Rief, Julio M. Fernandez, and Hermann E. Gaub. Elastically coupled two-level systems as a model for biopolymer extensibility. *Physical Review Letters*, 81(21):4764–4767, November 1998. doi: 10.1103/PhysRevLett.81.4764. URL http://prola.aps.org/abstract/PRL/v81/i21/p4764_1. Original details on mechanical unfolding analysis via Monte Carlo simulation.

[81] Robert B. Best, Susan B. Fowler, Jose L. Toca-Herrera, and Jane Clarke. A simple method for probing the mechanical unfolding pathway of proteins in detail. *Proceedings of the National Academy of Sciences of the United States of America*, 99(19):12143–12148, 2002. doi: 10.1073/pnas.192351899. URL http://www.pnas.org/cgi/content/abstract/99/19/12143. Points out order-of-magnitude errors in k_{u0} estimation from fitting Monte Carlo simulations.

[82] Rebecca C. Zinober, David J. Brockwell, Godfrey S. Beddard, Anthony W. Blake, Peter D. Olmsted, Sheena E. Radford, and D. Alastair Smith. Mechanically unfolding proteins: the effect of unfolding history and the supramolecular scaffold. *Protein Science*, 11 (12):2759–2765, December 2002. ISSN 0961-8368. doi: 10.1110/ps.0224602. URL `http://www.proteinscience.org/cgi/content/abstract/11/12/2759`. Introduces unfolding-order *scaffold effect* on average unfolding force.

[83] Ashlee Jollymore, Claire Lethias, Qing Peng, Yi Cao, and Hongbin Li. Nanomechanical properties of tenascin-X revealed by single- molecule force spectroscopy. *Journal of Molecular Biology*, 385(4):1277–1286, January 2009. ISSN 1089-8638. doi: 10.1016/j.jmb.2008.11.038. URL `http://dx.doi.org/10.1016/j.jmb.2008.11.038`.

[84] Helmut Grubmüller, Berthold Heymann, and P. Tavan. Ligand binding: molecular mechanics calculation of the streptavidin-biotin rupture force. *Science*, 271(5251):997–999, February 1996. ISSN 0036-8075. URL `http://www.ncbi.nlm.nih.gov/pubmed/8584939`.

[85] S. Izrailev, S. Stepaniants, M. Balsera, Y. Oono, and Klaus Schulten. Molecular dynamics study of unbinding of the avidin-biotin complex. *Biophysical Journal*, 72(4):1568–1581, April 1997. ISSN 0006-3495. URL `http://www.biophysj.org/cgi/content/abstract/72/4/1568`.

[86] E. Evans and K. Ritchie. Dynamic strength of molecular adhesion bonds. *Biophysical Journal*, 72(4):1541–1555, April 1997. ISSN 0006-3495. URL `http://www.biophysj.org/cgi/content/abstract/72/4/1541`.

[87] Michael Schlierf and Matthias Rief. Single-molecule unfolding force distributions reveal a funnel- shaped energy landscape. *Biophysical Journal*, 90(4):L33–L35, February 2006. ISSN 0006-3495. doi: 10.1529/biophysj.105.077982. URL `http://www.biophysj.org/cgi/content/abstract/90/4/L33`. The inspiration behind my sawtooth simulation. Bell model fit to $f_{unfold}(v)$, but Kramers model fit to unfolding distribution for a given v. Eq. (3) in the supplement is Evans and Ritchie [106] Eq. (2), but it is just [dying percent]·[surviving population] = [deaths]. $\nu \equiv k$ is the force/time-dependent off rate. The Kramers' rate equation (on page L34, the second equation in the paper) is Hänggi et al. [91] Eq. (4.56b) (page 275) and Socci et al. [211] Eq. (2) but Schlierf and Rief [87] gets the minus sign wrong in the exponent. $U_F(x = 0) \gg 0$ and $U_F(x_{\max}) \ll 0$ (*cf.* Schlierf and Rief [87] Fig. 1). Schlierf's integral (as written) contains $e^{-U_F(x_{\max})} \cdot e^{U_F(0)}$, which is huge, when it should contain $e^{U_F(x_{\max})} \cdot e^{-U_F(0)}$, which is tiny. For more details and a picture of the peak that forms the bulk of the integrand, see Eq. (3.12) and Fig. 3.6b. I pointed out this problem to Michael Schlierf, but he was unconvinced.

[88] Gerhard Hummer and Attila Szabo. Kinetics from nonequilibrium single-molecule pulling experiments. *Biophysical Journal*, 85(1):5–15, July 2003. ISSN 0006-3495. URL `http://www.biophysj.org/cgi/content/abstract/85/1/5`.

[89] Olga K. Dudko, Gerhard Hummer, and Attila Szabo. Intrinsic rates and activation free energies from single-molecule pulling experiments. *Physical Review Letters*, 96(10):108101, March 2006. ISSN 0031-9007. doi: 10.1103/PhysRevLett.96.108101.

[90] G. I. Bell. Models for the specific adhesion of cells to cells. *Science*, 200(4342):618–627, May 1978. ISSN 0036-8075. URL `http://www.jstor.org/stable/1746930`. The Bell model and a fair bit of cell bonding background.

[91] Peter Hänggi, Peter Talkner, and Michal Borkovec. Reaction-rate theory: Fifty years after Kramers. *Rev. Mod. Phys.*, 62(2):251–341, Apr 1990. doi: 10.1103/RevModPhys.62.251. URL `http://prola.aps.org/abstract/RMP/v62/i2/p251_1`. *The* Kramers' theory review article. See pages 268–279 for the Kramers-specific introduction.

[92] Olga K. Dudko, A. E. Filippov, J. Klafter, and M. Urbakh. Beyond the conventional description of dynamic force spectroscopy of adhesion bonds. *Proceedings of the National Academy of Sciences of the United States of America*, 100(20):11378–11381, September 2003. ISSN 0027-8424. doi: 10.1073/pnas.1534554100. URL `http://www.pnas.org/content/100/20/11378.abstract`.

[93] Changbong Hyeon and Devarajan Thirumalai. Can energy landscape roughness of proteins and RNA be measured by using mechanical unfolding experiments? *Proceedings of the National Academy of Sciences of the United States of America*, 100(18):10249–10253, September 2003. ISSN 0027-8424. doi: 10.1073/pnas.1833310100. URL `http://www.pnas.org/cgi/content/abstract/100/18/10249`. Derives the major theory behind my thesis. The Kramers rate equation is Hänggi et al.[91] Eq. (4.56c) (page 275).

[94] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, New Jersey, 2 edition, 1988.

[95] Python. URL `http://www.python.org/`.

[96] Hongbin Li, Andres F. Oberhauser, Susan B. Fowler, Jane Clarke, and Julio M. Fernandez. Atomic force microscopy reveals the mechanical design of a modular protein. *Proceedings of the National Academy of Sciences of the United States of America*, 97(12):6527–6531, 2000. doi: 10.1073/pnas.120048697. URL `http://www.pnas.org/cgi/content/abstract/97/12/6527`. Unfolding order not from protein-surface interactions. Mechanical unfolding of a chain of interleaved domains $ABABAB\ldots$ yielded a run of A unfoldings followed by a run of B unfoldings.

[97] John F. Marko and Eric D. Siggia. Stretching DNA. *Macromolecules*, 28(26):8759–8770, 1995. ISSN 0024-9297. URL `http://pubs3.acs.org/acs/journals/doilookup?in_doi=10.1021/ma00130a008`. Derivation of the Worm-like Chain interpolation function.

[98] Henk L. Granzier, Miklós S. Z. Kellermayer, M. Helmes, and K. Trombitás. Titin elasticity and mechanism of passive force development in rat cardiac myocytes probed by thin-filament extraction. *Biophysical Journal*, 73(4):2043–2053, October 1997. ISSN 0006-3495. doi: 10.1016/S0006-3495(97)78234-1. URL `http://www.cell.com/biophysj/retrieve/pii/S0006349597782341`.

[99] Wolfgang A. Linke, M. R. Stockmeier, M. Ivemeyer, H. Hosser, and P. Mundeol. Characterizing titin's I-band Ig domain region as an entropic spring. *Journal of Cell Science*, 111 (Pt 11): 1567–1574, June 1998. ISSN 0021-9533. URL `http://jcs.biologists.org/cgi/content/abstract/111/11/1567`.

[100] Douglas B. Staple, Stephen H. Payne, Andrew L. C. Reddin, and Hans Jürgen Kreuzer. Model for stretching and unfolding the giant multidomain muscle protein using single-molecule force spectroscopy. *Physical Review Letters*, 101(24):248301, December 2008. ISSN 0031-9007. doi: 10.1103/PhysRevLett.101.248301. URL `http://dx.doi.org/10.1103/PhysRevLett.101.248301`.

[101] Andreas Janshoff, Marcus Neitzert, York Oberdörfer, and Harald Fuchs. Force spectroscopy of molecular systems-single molecule spectroscopy of polymers and biomolecules. *Angew. Chem. Int. Ed. Engl.*, 39(18):3212–3237, September 2000. ISSN 1521-3773. doi: 10.1002/1521-3773(20000915)39:18⟨3212::AID-ANIE3212⟩3.0.CO;2-X. URL `http://dx.doi.org/10.1002/1521-3773(20000915)39:18<3212::AID-ANIE3212>3.0.CO;2-X`.

[102] Peter H. Verdier. Relaxation behavior of the freely jointed chain. *The Journal of Chemical Physics*, 52(11):5512–5517, 1970. doi: 10.1063/1.1672818. URL `http://link.aip.org/link/?JCP/52/5512/1`.

[103] John William Hatfield and Stephen R. Quake. Dynamic properties of an extended polymer in solution. *Physical Review Letters*, 82(17):3548–3551, Apr 1999. doi: 10.1103/PhysRevLett.82.3548. URL http://link.aps.org/abstract/PRL/v82/p3548. Defines WLC and FJC models, citing textbooks.

[104] Elias M. Puchner, Gereon Franzen, Mathias Gautel, and Hermann E. Gaub. Comparing proteins by their unfolding pattern. *Biophysical Journal*, 95(1):426–434, July 2008. ISSN 1542-0086. doi: 10.1529/biophysj.108.129999. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2426622/. Contour length space and barrier position fingerprinting. There are errors in Eq. (3), propagated from Livadaru et al. [212]. I contacted Elias Puchner and pointed out the typos, and he revised his FRC fit parameters from $\gamma = 22°$ and $b = 0.4$ nm to $\gamma = 41°$ and $b = 0.11$ nm. The combined effect on Fig. 3 of fixing the equation typos and adjusting the fit parameters was small, so their conclusions are still sound.

[105] T. E. Fisher, Piotr E. Marszalek, Andres F. Oberhauser, Mariano Carrion-Vazquez, and Julio M. Fernandez. The micro-mechanics of single molecules studied with atomic force microscopy. *Journal of Physiology*, 520 Pt 1:5–14, October 1999. ISSN 0022-3751. URL http://www.ncbi.nlm.nih.gov/pubmed/10517795.

[106] E. Evans and K. Ritchie. Strength of a weak bond connecting flexible polymer chains. *Biophysical Journal*, 76(5):2439–2447, May 1999. ISSN 0006-3495. URL http://www.biophysj.org/cgi/content/abstract/76/5/2439. Develops Kramers improvement on Bell model for domain unfolding. Presents unfolding under variable loading rates. Often cited as the "Bell–Evans" model. They derive a unitless treatment, scaling force by f_β, time by τ_f, and elasiticity by compliance $c(f)$. The appendix has relaxation time formulas for WLC and FJC polymer models.

[107] R.E. Jones and D.P. Hart. Force interactions between substrates and SPM cantilevers immersed in fluids. *Tribology International*, 38(3):355–361, 2005. ISSN 0301-679X. doi: 10.1016/j.triboint.2004.08.016. URL http://dx.doi.org/10.1016/j.triboint.2004.08.016.

[108] Olga K. Dudko, Jérôme Mathé, Attila Szabo, Amit Meller, and Gerhard Hummer. Extracting kinetics from single-molecule force spectroscopy: Nanopore unzipping of DNA hairpins. *Biophysical Journal*, 92(12):4188–4195, June 2007. ISSN 0006-3495. doi: 10.1529/biophysj.106.102855.

[109] Emily B. Walton, Sunyoung Lee, and Krystyn J. Van Vliet. Extending Bell's model: How force transducer stiffness alters measured unbinding forces and kinetics of molecular complexes. *Biophysical Journal*, 94(7):2621–2630, April 2008. ISSN 1542-0086. doi: 10.1529/biophysj.107.114454. Some detailed estimates at U(x).

[110] H. A. Kramers. Brownian motion in a field of force and the diffusion model of chemical reactions. *Physica*, 7(4):284–304, April 1940. ISSN 0031-8914. doi: 10.1016/S0031-8914(40)90098-2. URL http://dx.doi.org/10.1016/S0031-8914(40)90098-2. Seminal paper on thermally activated barrier crossings.

[111] Julian Shillcock and Udo Seifert. Escape from a metastable well under a time-ramped force. *Phys Rev E Stat Nonlin Soft Matter Phys*, 57(6):7301–7304, Jun 1998. doi: 10.1103/PhysRevE.57.7301. URL http://link.aps.org/abstract/PRE/v57/p7301.

[112] N.G. van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, North-Holland Personal Library, Amsterdam, 3 edition, 2007.

[113] NIST/SEMATECH. *e-Handbook of Statistical Methods*, chapter 1.3.6.6.16: Extreme Value Type I Distribution. In NIST/SEMATECH [213], October 2009. URL http://www.itl.nist.gov/div898/handbook/eda/section3/eda366g.htm. This manual was developed from seed material produced by Mary Natrella.

[114] Olga K. Dudko, Gerhard Hummer, and Attila Szabo. Theory, analysis, and interpretation of single-molecule force spectroscopy experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 105(41):15755–15760, October 2008. ISSN 1091-6490. doi: 10.1073/pnas.0806085105. URL http://www.ncbi.nlm.nih.gov/pubmed/18852468.

[115] Fabrizio Benedetti, Cristian Micheletti, Giovanni Bussi, Sergey K. Sekatskii, and Giovanni Dietler. Nonkinetic modeling of the mechanical unfolding of multimodular proteins: theory and experiments. *Biophysical Journal*, 101(6):1504–1512, September 2011. ISSN 1542-0086. doi: 10.1016/j.bpj.2011.07.047. URL http://www.ncbi.nlm.nih.gov/pubmed/21943432.

[116] Gregory E. Sims, Se-Ran Jun, Guohong A. Wu, and Sung-Hou Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences of the United States of America*, 106(8):2677–2682, February 2009. ISSN 1091-6490. doi: 10.1073/pnas.0813249106. URL http://www.pnas.org/content/106/8/2677.

[117] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991. ISSN 0018-9448. doi: 10.1109/18.61115. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=2227&arnumber=61115&count=35&index=9.

[118] NIST/SEMATECH. *e-Handbook of Statistical Methods*, chapter 1.3.5.15: Chi-Square Goodness-of-Fit Test. In NIST/SEMATECH[213], May 2013. URL http://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm. This manual was developed from seed material produced by Mary Natrella.

[119] Marisa B. Roman. *Macromolecular crowding effects in the mechanical unfolding forces of proteins*. PhD thesis, Drexel University, May 2012. URL http://hdl.handle.net/1860/3854.

[120] Alfred North Whitehead. *An introduction to mathematics*. Williams & Norgate, London, 1911. URL http://archive.org/details/introductiontoma00whitiala. The "civilization" quote is on page 61.

[121] Jon F. Claerbout and Martin Karrenbach. Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992*, pages 601–604. Society of Exploration Geophysicists, 1992. doi: 10.1190/1.1822162. URL http://library.seg.org/doi/abs/10.1190/1.1822162.

[122] Jonathan B. Buckheit and David L. Donoho. Wavelab and reproducible research. In Anestis Antoniadis and Georges Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 55–81. Springer Science + Business Media, LLC, 1995. ISBN 978-0-387-94564-4. doi: 10.1007/978-1-4612-2544-7_5. URL http://dx.doi.org/10.1007/978-1-4612-2544-7_5.

[123] M. Schwab, Martin Karrenbach, and Jon F. Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering*, 2(6):61–67, November–December 2000. ISSN 1521-9615. doi: 10.1109/5992.881708.

[124] Patrick Vandewalle, Jelena Kovacevic, and Martin Vetterli. Reproducible research in signal processing - what, why, and how. *IEEE Signal Processing Magazine*, 26(3):37–47, May 2009. ISSN 1053-5888. doi: 10.1109/MSP.2009.932122. URL http://rr.epfl.ch/17/.

[125] National Instruments. LabVIEW. URL http://www.ni.com/labview/. A graphical programming language designed for developing experiment control software.

[126] WaveMetrics. IGOR Pro. URL http://www.wavemetrics.com/products/igorpro/igorpro.htm.

[127] National Instruments. URL https://www.ni.com/.

[128] Edsger Wybe Dijkstra. *Notes on Structured Programming*. Technische Hogeschool Eindhoven, Nederland, Onderafdeling der Wiskunde, April 1970. URL `http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF`. T.H. Report 70-WSK-03.

[129] Niklaus Wirth. On the composition of well-structured programs. *ACM Computing Surveys*, 6(4):247–259, December 1974. ISSN 0360-0300. doi: 10.1145/356635.356639. URL `http://doi.acm.org/10.1145/356635.356639`.

[130] Ben Shneiderman and Richard Mayer. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Computer & Information Sciences*, 8(3):219–238, 1979. ISSN 0091-7036. doi: 10.1007/BF00977789. URL `http://dx.doi.org/10.1007/BF00977789`.

[131] John Hughes. Why functional programming matters. *The Computer Journal*, 32(2):98–107, 1989. doi: 10.1093/comjnl/32.2.98. URL `http://comjnl.oxfordjournals.org/content/32/2/98.abstract`.

[132] National Instruments. Best practices for managing NI LabVIEW applications using the Project Explorer. *Software Engineering with LabVIEW*, January 2013. URL `http://www.ni.com/white-paper/7197/en`.

[133] Thomas B. Hilburn. A top-down approach to teaching an introductory computer science course. *ACM Special Interest Group on Computer Science Education Bulletin*, 25(1):58–62, March 1993. ISSN 0097-8418. doi: 10.1145/169073.169349. URL `http://doi.acm.org/10.1145/169073.169349`.

[134] Greg Wilson. Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science & Engineering*, November–December 2006.

[135] Elijah Kerry. Source code control: Using TortoiseSVN (Subversion) with LabVIEW for diff and merge operations, December 2012. URL `https://decibel.ni.com/content/docs/DOC-2936`.

[136] National Instruments. How can I determine the LabVIEW version that was used to save a VI?, April 2012. URL `http://digital.ni.com/public.nsf/allkb/0C72D335AA87DD6486256FC40069C17F`. Document ID: 3JDC8IZH.

[137] National Instruments. How to upgrade or revert a VI to a different version of LabVIEW, August 2012. URL `http://www.ni.com/white-paper/8387/en`.

[138] National Instruments. Using NI-DAQmx in LabWindows/CVI, March 2007. URL `http://www.ni.com/white-paper/2931/en`. Control libraries for data aquisition. While these libraries form the basis of LabVIEW's control stack, you can also use them directly from other languages (like C).

[139] Cygwin. URL `http://www.cygwin.com/`. A POSIX emulation layer for Microsoft Windows. With Cygwin, Windows seems more like UNIX or Linux.

[140] EPICS: Experimental physics and industrial control system. URL `http://www.aps.anl.gov/epics/`.

[141] David M. Beazley. SWIG: An easy to use tool for integrating scripting languages with C and C++. In *Proceedings of the 4th conference on USENIX Tcl/Tk Workshop*, volume 4 of *Tcl/Tk 1996*, pages 15–15, Berkeley, CA, USA, 1996. USENIX Association. URL `http://www.swig.org/`. SWIG, the simplified wrapper and interface generator, makes it easy to wrap C and C++ libraries for use from higher level scripting languages like Python. Often, you don't need much more than the library's header file to generate a full wrapper. However, the generated wrappers are usually very thin, which can make them awkward to use. If you want to write a thicker, more idiomatic wrapper, it's probably easier to use a tool like Cython instead of SWIG.

[142] Frederick P. Brooks, Jr. *The mythical man-month*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 20th anniversary edition, 1995. ISBN 0-201-83595-9. URL `http://dl.acm.org/citation.cfm?id=207583`. First published in 1975.

[143] Cython: C-extensions for Python. URL `http://www.cython.org/`.

[144] William Trevor King. H5config, 2013. URL `http://pypi.python.org/pypi/h5config/`. Version 0.3+. I used commit ec3a4a0 (`storage:hdf5: Add support for dtype(object) and 0d string arrays`, 2013-01-18).

[145] `scipy.optimize.leastsq`. URL `http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.leastsq.html`.

[146] MINPACK. URL `http://www.netlib.org/minpack/`.

[147] The HDF Group. Hierarchical data format version 5, 2013. URL `http://www.hdfgroup.org/HDF5/`. Version 1.8.10.

[148] YAML ain't markup language (YAML™) version 1.2, October 2009. URL `http://www.yaml.org/`.

[149] Douglas W. Jones. Control of stepping motors, 1995. URL `http://homepage.cs.uiowa.edu/~jones/step/`. This material expands on material originally posted to the rec.railroad newsgroup in 1990. Significant parts of this material have been republished as sections 5.2.10, 10.8, 10.9 and 10.10 of the Handbook of Small Electric Motors edited by W. H. Yeadon and A. W. Yeadon, McGraw-Hill, 2001, and as Applications Note 907 published by Microchip Inc in 2004.

[150] Pymodbus. URL `https://github.com/bashwork/pymodbus`. I used a snapshot from between v1.0.0 and v1.1.0 in the Subversion repository that pymodbus used before migrating to Git. The exact commit is now Git ID 3548a7d (`Updating the documentation for the serial client/servers`, 2011-11-20).

[151] Melcor. Companies don't stay in business forever, but lab equipment does. Our controller is still going strong (since 1999), but Melcor has moved around. According to their 2005 announcement the Laird Group PLC purchased Melcor from Fedders Corporation, and by 2009 (according to the Internet Archive Wayback Machine) they phased out the old website at melcor.com in favor of their own thermal site. It looks like there is no longer support for the older MTCA controllers.

[152] J.G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Transactions of the American Society of Mechanical Engineers*, 64:759–765, November 1942. URL `http://www.driedger.ca/Z-N/Z-N.html`.

[153] G. H. Cohen and G. A. Coon. Theoretical considerations of retarded control. *Transactions of the American Society of Mechanical Engineers*, 75:827–834, 1953.

[154] F. S. Wang, F. S. Juang, and C. T. Chan. Optimal tuning of PID controllers for single and cascade control loops. *Chemical Engineering Communications*, 132(1):15–34, 1995. ISSN 0098-6445. doi: 10.1080/00986449508936294. URL `http://www.tandfonline.com/doi/abs/10.1080/00986449508936294`.

[155] K. J. Åström, T. Hägglund, C. C. Hang, and W. K. Ho. Automatic tuning and adaptation for PID controllers—a survey. *Control Engineering Practice*, 1(4):699–714, 1993. ISSN 0967-0661. doi: 10.1016/0967-0661(93)91394-C. URL `http://dx.doi.org/10.1016/0967-0661(93)91394-C`.

[156] Daniel Aioanei, Marco Brucale, and Bruno Samorì. Open source platform for the execution and analysis of mechanical refolding experiments. *Bioinformatics (Oxford, England)*, 27(3): 423–425, February 2011. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq663. URL `http://www.ncbi.nlm.nih.gov/pubmed/21123222`. Refolding is a suite for performing and analyzing double-pulse refolding experiments. The experiment-driver is mostly written in Java with the analysis code in Python. The driver is curious; it uses the NanoScope scripting interface to drive the experiment through the NanoScope software by impersonating a mouse-wielding user (like Selenium does for web browsers). See the `RobotNanoDriver.java` code for details. There is also support for automatic velocity clamp analysis.

[157] Donatello Materassi, Paolo Baschieri, Bruno Tiribilli, Giampaolo Zuccheri, and Bruno Samorì. An open source/real-time atomic force microscope architecture to perform customizable force spectroscopy experiments. *Review of Scientific Instruments*, 80(8):084301, August 2009. ISSN 1089-7623. doi: 10.1063/1.3194046. URL `http://www.ncbi.nlm.nih.gov/pubmed/19725671`. Although this paper claims to present an open source experiment control framework (on Linux!), it doesn't actually link to any source code. This is puzzling and frusterating.

[158] Hans-Jürgen Butt and Manfred Jaschke. Calculation of thermal noise in atomic force microscopy. *Nanotechnology*, 6(1):1–7, 1995. doi: 10.1088/0957-4484/6/1/001. URL `http://stacks.iop.org/0957-4484/6/1`. Corrections to basic $kx^2 = kBT$ due to higher order modes in rectangular cantilevers.

[159] R. W. Stark, T. Drobek, and W. M. Heckl. Thermomechanical noise of a free v-shaped cantilever for atomic- force microscopy. *Ultramicroscopy*, 86(1–2):207–215, January 2001. ISSN 0304-3991. doi: http://dx.doi.org/10.1016/S0304-3991(00)00077-2. Higher mode adjustments for v-shaped cantilevers from simulation.

[160] Jeffrey L. Hutter. Comment on tilt of atomic force microscope cantilevers: Effect on spring constant and adhesion measurements. *Langmuir*, 21(6):2630–2632, March 2005. ISSN 0743-7463. doi: 10.1021/la047670t. Tilted cantilever corrections (not needed? see Ohler/VEECO note).

[161] Benjamin Ohler. Cantilever spring constant calibration using laser doppler vibrometry. *Review of Scientific Instruments*, 78(6):063701, 2007. doi: 10.1063/1.2743272. URL `http://link.aip.org/link/?RSI/78/063701/1`. Excellent review of thermal calibration to 2007, but nothing in the way of derivations. Compares thermal tune and Sader method with laser Doppler vibrometry.

[162] Andreas Roters and Diethelm Johannsmann. Distance-dependent noise measurements in scanning force microscopy. *Journal of Physics: Condensed Matter*, 8(41):7561–7577, 1996. doi: 10.1088/0953-8984. URL `http://stacks.iop.org/0953-8984/8/7561`. They actually write down a Lagrangian formula and give a decent derivation of PSD, but don't show or work out the integrals.

[163] Fabrizio Benedetti. *Statistical Study of the Unfolding of Multimodular Proteins and their Energy Landscape by Atomic Force Microscopy*. PhD thesis, Lausanne, 2012. URL `http://infoscience.epfl.ch/record/181215`.

[164] Kirstine Berg-Sørensen and Henrik Flyvbjerg. Power spectrum analysis for optical tweezers. *Review of Scientific Instruments*, 75(3):594–612, 2004. ISSN 0034-6748. doi: 10.1063/1.1645654. URL `http://rsi.aip.org/resource/1/rsinak/v75/i3/p594_s1`.

[165] Kirstine Berg-Sørensen and Henrik Flyvbjerg. The colour of thermal noise in classical brownian motion: a feasibility study of direct experimental observation. *New Journal of Physics*, 7(1):38, February 2005. doi: 10.1088/1367-2630/7/1/038. URL `http://stacks.iop.org/1367-2630/7/i=1/a=038`.

[166] Frederick Gittes and Christoph F. Schmidt. Thermal noise limitations on micromechanical experiments. *European Biophysics Journal*, 27(1):75–81, January 1998. ISSN 0175-7571. doi: 10.1007/s002490050113. URL `http://dx.doi.org/10.1007/s002490050113`.

[167] J. Howard and A. J. Hudspeth. Compliance of the hair bundle associated with gating of mechanoelectrical transduction channels in the bullfrog's saccular hair cell. *Neuron*, 1:189–199, May 1988. doi: 10.1016/0896-6273(88)90139-0. URL `http://www.cell.com/neuron/retrieve/pii/0896627388901390`. Initial thermal calibration paper as cited by Florin et al.[27]. This is not an AFM paper, but it uses the equipartition theorem to calculate the spring constant of hair fibers by measuring their tip displacement variance. The discussion occurs in the *Manufacture and Calibration of Fibers* section on pages 197–198. Actual details are scarce, but I believe this is the original source of the "Lorentzian" and "10% accuracy" ideas that have haunted themal calibration ever since.

[168] Eric W. Weisstein. Lorentzian function, June 2013. URL `http://mathworld.wolfram.com/LorentzianFunction.html`. Defines the standard Lorentzian function.

[169] S. Thornton and J. Marion. *Classical Dynamics of Particles and Systems*, chapter Appendix D, page 609. Brooks/Cole, Belmont, CA, 5 edition, 2004. ISBN 0-534-40896-6. See Eq. (12.0.13).

[170] February 2008. See Wikipedia's currently excellent page (Feb 15th, 2008) `http://en.wikipedia.org/wiki/Fourier_transform#Functional_relationships`, or derive it for yourself in about three lines.

[171] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipies in C: The Art of Scientific Computing*, chapter 12, page 498. In Press et al.[214], 2 edition, 1992. See Eq. (12.0.13).

[172] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipies in C: The Art of Scientific Computing*, chapter 12, page 498. In Press et al.[214], 2 edition, 1992. See Eq. (12.0.14).

[173] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipies in C: The Art of Scientific Computing*, chapter 12, page 498. In Press et al.[214], 2 edition, 1992. See Eq. (12.0.12).

[174] Wiener–Khinchin theorem, June 2013. URL `http://en.wikipedia.org/wiki/Wiener%E2%80%93Khinchin_theorem`.

[175] John Bechhoefer and Scott Wilson. Faster, cheaper, safer optical tweezers for the undergraduate laboratory. *American Journal of Physics*, 70(4):393–400, 2002. doi: 10.1119/1.1445403. URL `http://link.aip.org/link/?AJP/70/393/1`. Good discussion of the effect of correlation time on calibration. References work on deconvolving thermal noise from other noise[215]. Excellent detail on power spectrum derivation and thermal noise for extremely overdamped oscillators in Appendix A (references Rief[216]), except that their equation A12 is missing a factor of $1/\pi$. I pointed this out to John Bechhoefer and he confirmed the error.

[176] C. Grossman and A. Stout. Optical tweezers advanced lab. 2005. Fairly complete overdamped PSD derivation in Section 4.3. Cites Tlusty et al.[217] and Bechhoefer and Wilson[175] for further details. However, Tlusty (listed as reference 8) doesn't contain the thermal response fn. derivation it was cited for. Also, the single sided PSD definition credited to reference 9 (listed as Bechhoefer) looks more like Press (listed as reference 10). I imagine Grossman and Stout mixed up their references, and meant to refer to Bechhoefer and Wilson[175] and Press et al.[214] respectively instead.

[177] OMEGA engineering, inc. URL `http://www.omega.com/`.

[178] H. H. Ku. Notes on the use of propagation of error formulas. *Journal of Research of the National Bureau of Standards. Section C: Engineering and Instrumentation*, 70C(4):263–273, October 1966. ISSN 0022-4316. URL http://nistdigitalarchives.contentdm.oclc.org/cdm/compoundobject/collection/p13011coll6/id/78003/rec/5.

[179] Proceedings of the National Academy of Sciences of the United States of America. PNAS information for authors, February 2013. URL http://www.pnas.org/site/misc/iforc.pdf.

[180] Nature. Guide to publication policies of the nature journals, April 2013. URL http://www.nature.com/authors/gta.pdf.

[181] The HDF Group. HDF group history, 2013. URL http://www.hdfgroup.org/about/history.html.

[182] The HDF Group. Who uses HDF?, 2013. URL http://www.hdfgroup.org/users.html.

[183] Y. Martin, C. C. Williams, and H. K. Wickramasinghe. Atomic force microscope—force mapping and profiling on a sub 100-åscale. *Journal of Applied Physics*, 61(10):4723–4729, May 1987. ISSN 0021-8979. doi: 10.1063/1.338807. URL http://jap.aip.org/resource/1/japiau/v61/i10/p4723_s1.

[184] Jeffrey L. Hutter and John Bechhoefer. Erratum: Calibration of atomic-force microscope tips. *Review of Scientific Instruments*, 64(11):3342, November 1993. doi: 10.1063/1.1144449. URL http://rsi.aip.org/resource/1/rsinak/v64/i11/p3342_s1. V. Croquette pointed out that they should calibrate the response of their optical-detection electronics.

[185] Michael Kuhn, Harald Janovjak, Maurice Hubain, and Daniel J. Müller. Automated alignment and pattern recognition of single-molecule force spectroscopy data. *Journal of Microscopy*, 218(2):125–132, May 2005. ISSN 0022-2720. doi: 10.1111/j.1365-2818.2005.01478.x. URL http://www.ncbi.nlm.nih.gov/pubmed/15857374. Development stalled in 2005 after Michael graduated.

[186] Philippe Carl and Paul Dalhaimer. PUNIAS: Protein unfolding and nano-indentation analysis software, October 2005. URL http://punias.voila.net/. 4 Int. Workshop, Scanning Probe Microscopy in Life Sciences.

[187] Philippe Carl and Hermann Schillers. Elasticity measurement of living cells with an atomic force microscope: data acquisition and processing. *Pflügers Archiv: European journal of physiology*, 457(2):551–559, November 2008. ISSN 0031-6768. doi: 10.1007/s00424-008-0524-3. URL http://www.ncbi.nlm.nih.gov/pubmed/18481081. Mentions PUNIAS as if it was in-house software, which makes sense because Philippe Carl seems to be a major author.

[188] JPK Instruments AG. Forcerobot. URL http://www.jpk.com/forcerobot-r-300-overview.500.en.html.

[189] Jens Struckmeier, Reiner Wahl, Mirko Leuschner, Joao Nunes, Harald Janovjak, Ulrich Geisler, Gerd Hofmann, Torsten Jähnke, and Daniel J. Müller. Fully automated single-molecule force spectroscopy for screening applications. *Nanotechnology*, 19(38):384020, September 2008. ISSN 0957-4484. doi: 10.1088/0957-4484/19/38/384020. URL http://www.ncbi.nlm.nih.gov/pubmed/21832579. An advertisement for JPK's ForceRobot.

[190] Bill Andreopoulos and Dirk Labudde. Efficient unfolding pattern recognition in single molecule force spectroscopy data. *Algorithms for molecular biology: AMB*, 6(1):16, June 2011. ISSN 1748-7188. doi: 10.1186/1748-7188-6-16. URL http://www.ncbi.nlm.nih.gov/pubmed/21645400.

[191] Asylum Research. MFP-3D. URL http://www.asylumresearch.com/Products/Mfp3DSA/Mfp3DSA.shtml.

[192] Bruker Corporation. PicoForce. URL http://www.bruker.com/products/surface-analysis/atomic-force-microscopy/modes/modes-techniques/nanomechanical-modes/forcedistance.html.

[193] Massimo Sandal, Francesco Valle, Isabella Tessari, Stefano Mammi, Elisabetta Bergantino, Francesco Musiani, Marco Brucale, Luigi Bubacco, and Bruno Samorì. Conformational equilibria in monomeric α-synuclein at the single-molecule level. *PLOS Biology*, 6(1):e6, January 2008. ISSN 1545-7885. doi: 10.1371/journal.pbio.0060006. URL http://www.ncbi.nlm.nih.gov/pubmed/18198943.

[194] Fisher Scientific. Calcium Chloride, anhydrous. CAS number 10043-52-4, catalog number C614-500, lot number 065514.

[195] Franz Hofmeister. Zur lehre von der wirkung der salze. *Archive für experimentelle Pathologie und Pharmakologie*, 25(1):1–30, March 1888. doi: 10.1007/BF01838161. URL http://link.springer.com/article/10.1007/BF01838161.

[196] V. P. Chauhan, I. Ray, A. Chauhan, J. Wegiel, and H. M. Wisniewski. Metal cations defibrillize the amyloid beta-protein fibrils. *Neurochemical research*, 22(7):805–809, July 1997. ISSN 0364-3190. doi: 10.1023/A:1022079709085. URL http://www.ncbi.nlm.nih.gov/pubmed/9232632. From page 806, "The exact mechanism by which these metal ions affect the fibrillization of Aβ is not known.".

[197] Anna Itkin, Vincent Dupres, Yves F. Dufrêne, Burkhard Bechinger, Jean-Marie Ruysschaert, and Vincent Raussens. Calcium ions promote formation of amyloid β-peptide (1-40) oligomers causally implicated in neuronal toxicity of Alzheimer's disease. *PLOS ONE*, 6(3):e18250, March 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0018250. URL http://www.ncbi.nlm.nih.gov/pubmed/21464905. 2 mM of Ca^{2+} is the *extracellular* concentration. Cytosol concetrations are in the μM range.

[198] Yanjie Zhang and Paul S. Cremer. Interactions between macromolecules and ions: The Hofmeister series. *Current Opinion in Chemical Biology*, 10(6):658–663, December 2006. ISSN 1367-5931. doi: 10.1016/j.cbpa.2006.09.020. URL http://www.ncbi.nlm.nih.gov/pubmed/17035073. A quick pass through Hofmeister history, but no discussion of cations ("A complete picture will inevitably involve an integrated understanding of the role of cations (including guanidinium ions) and osmolytes (such as urea and tri-methylamine N-oxide) as well. There has been some progress in these fields, although such subjects are generally beyond the scope of this short review.").

[199] Ran Friedman. Ions and the protein surface revisited: extensive molecular dynamics simulations and analysis of protein structures in alkali-chloride solutions. *The Journal of Physical Chemistry B*, 115(29):9213–9223, July 2011. ISSN 1520-5207. doi: 10.1021/jp112155m. URL http://www.ncbi.nlm.nih.gov/pubmed/21688775.

[200] Jernej Zidar and Franci Merzel. Probing amyloid-beta fibril stability by increasing ionic strengths. *The Journal of Physical Chemistry B*, 115(9):2075–2081, March 2011. ISSN 1520-5207. doi: 10.1021/jp109025b. URL http://www.ncbi.nlm.nih.gov/pubmed/21329333. Only study NaCl over the range to 308 mM, but show a general decreased hydrogen bonding as concentration increases.

[201] Micholas Dean Smith and Luis Cruz Cruz. Effect of ionic aqueous environments on the structure and dynamics of the aβ_{21-30} fragment: a molecular-dynamics study. *The Journal of Physical Chemistry B*, 117(22):6614–6624, June 2013. ISSN 1520-5207. doi: 10.1021/jp312653h. URL http://www.ncbi.nlm.nih.gov/pubmed/23675877.

[202] Adrian M. Isaacs, David B. Senn, Menglan Yuan, James P. Shine, and Bruce A. Yankner. Acceleration of amyloid β-peptide aggregation by physiological concentrations of calcium. *The Journal of Biological Chemistry*, 281(38):27916–27923, September 2006. ISSN 0021-9258. doi:

10.1074/jbc.M602061200. URL http://www.ncbi.nlm.nih.gov/pubmed/16870617. Physiological levels of NaCl are ~ 150 mM. Ca^{2+} is ~ 2 mM.

[203] Bruker. MultiMode atomic force microscope with a nanoscope controller. This microscope was originally developed by Digital Instruments (DI). DI was aquired by Veeco in February of 1998, and passed off to Bruker in August 2010.

[204] Manfred Jaschke and Hans-Jürgen Butt. Height calibration of optical lever atomic force microscopes by simple laser interferometry. *Review of Scientific Instruments*, 66(2):1258–1259, 1995. ISSN 0034-6748. doi: 10.1063/1.1146018. URL http://rsi.aip.org/resource/1/rsinak/v66/i2/p1258_s1.

[205] Git. URL http://git-scm.com/.

[206] Greg Wilson. Where's the real bottleneck in scientific computing? *American Scientist*, January–February 2006.

[207] Dhavide A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Katy Huff, Ian Mitchell, Mark Plumbley, Ben Waugh, Ethan P. White, Greg Wilson, and Paul Wilson. Best practices for scientific computing. *arXiv Computing Research Repository*, abs/1210.0530, November 2012. URL http://arxiv.org/abs/1210.0530. v3: Thu, 29 Nov 2012 19:28:27 GMT.

[208] C. V. Heer. *Statistical mechanics, kinetic theory, and stochastic processes*. Academic Press, New York, 1972. ISBN 0-123-36550-3.

[209] Hendrik Dietz and Matthias Rief. Exploring the energy landscape of GFP by single-molecule mechanical experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16192–16197, 2004. doi: 10.1073/pnas.0404549101. URL http://www.pnas.org/cgi/content/abstract/101/46/16192. Towards use of Green Flourescent Protein (GFP) as an embedded force probe. Nice energy-landscape-to-one-dimension compression graphic.

[210] Arlette R. C. Baljon and Mark O. Robbins. Energy dissipation during rupture of adhesive bonds. *Science*, 271(5248):482–484, January 1996. doi: 10.1126/science.271.5248.482. URL http://www.sciencemag.org/content/271/5248/482.abstract.

[211] N. D. Socci, J. N. Onuchic, and P. G. Wolynes. Diffusive dynamics of the reaction coordinate for protein folding funnels. *The Journal of Chemical Physics*, 104(15):5860–5868, 1996. doi: 10.1063/1.471317. URL http://link.aip.org/link/?JCP/104/5860/1. A nice introduction to some quantitative ramifications of the funnel energy landscape. There's also a bit of Kramers' theory and graph theory thrown in for good measure.

[212] L. Livadaru, R. R. Netz, and Hans Jürgen Kreuzer. Stretching response of discrete semiflexible polymers. *Macromolecules*, 36(10):3732–3744, April 2003. doi: 10.1021/ma020751g. URL http://pubs.acs.org/doi/abs/10.1021/ma020751g. There are two typos in Eq. (46). Livadaru et al.[212] have

$$\frac{R_z}{L} = \begin{cases} \frac{fa}{3k_BT} & \frac{fb}{k_BT} < \frac{b}{l} \\ 1 - \left(\frac{fl}{4k_BT}\right)^{-0.5} & \frac{b}{l} < \frac{fb}{k_BT} < \frac{l}{b} \\ 1 - \left(\frac{fb}{ck_BT}\right)^{-1} & \frac{1}{b} < \frac{fb}{k_BT} \ , \end{cases} \tag{8.1}$$

but the correct formula is

$$\frac{R_z}{L} = \begin{cases} \frac{fa}{3k_BT} & \frac{fb}{k_BT} < \frac{b}{l} \\ 1 - \left(\frac{4fl}{k_BT}\right)^{-0.5} & \frac{b}{l} < \frac{fb}{k_BT} < \frac{l}{b} \\ 1 - \left(\frac{cfb}{k_BT}\right)^{-1} & \frac{1}{b} < \frac{fb}{k_BT} \ , \end{cases} \tag{8.2}$$

with both the 4 and the c moved into their respective numerators. I pointed these errors out to Roland Netz in 2012, along with the fact that even with the corrected formula there is a discontinuity between the low- and moderate-force regimes. Netz confirmed the errors, and pointed out that the discontinuity is because Eq. (46) only accounts for the scaling (without prefactors). Unfortunately, there does not seem to be a published erratum pointing out the error and at least Puchner et al. [104] have quoted the incorrect form.

[213] NIST/SEMATECH. *e-Handbook of Statistical Methods*. NIST/SEMATECH, Boulder, Colorado, May 2013. URL `http://www.itl.nist.gov/div898/handbook/`. This manual was developed from seed material produced by Mary Natrella.

[214] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipies in C: The Art of Scientific Computing*. Columbia University Press, New York, 2 edition, 1992. See Sections 12.0, 12.1, 12.3, and 13.4 for a good introduction to Fourier transforms and power spectrum estimation.

[215] Glen Cowan. *Statistical Data Analysis*. Oxford University Press, New York, 1998. Noise deconvolution in Chapter 11.

[216] Frederick Rief. *Fundamentals of Statistical and Thermal Physics*. McGraw-Hill, New York, 1965. Thermal noise for simple harmonic oscillators, in Chapter 15, Sections 6 and 10.

[217] Tsvi Tlusty, Amit Meller, and Roy Bar-Ziv. Optical gradient forces of strongly localized fields. *Physical Review Letters*, 81(8):1738–1741, August 1998. doi: 10.1103/PhysRevLett.81.1738. also at `http://nanoscience.bu.edu/papers/p1738_1_Meller.pdf`. Cited by Grossman and Stout [176] for derivation of thermal response functions. However, I only see a referenced thermal energy when they list the likelyhood of a small partical (radius $< R_c$) escaping due to thermal energy, where R_c is roughly $R_c \sim (k_B T/\alpha I_0)^{1/3}$, α is a dielectric scaling term, and I_0 is the maximum beam energy density. I imagine Grossman and Stout mixed up this reference.

[218] Peter J. Mohr, Barry N. Taylor, and David B. Newell. CODATA recommended values of the fundamental physical constants: 2006, June 2008. URL `http://physics.nist.gov/cgi-bin/cuu/Value?k`.

[219] W. W. Cleland. Dithiothreitol, a new protective reagent for sh groups. *Biochemistry*, 3(4): 480–482, April 1964. ISSN 0006-2960. doi: 10.1021/bi00892a002. URL `http://www.ncbi.nlm.nih.gov/pubmed/14192894`.

Appendix A: Cantilever calibration

A.1 Contour integration

As a brief review, some definite integrals from $-\infty$ to ∞ can be evaluated by integrating along the contour $\mathcal{C}$ shown in Fig. A.1.

A sufficient condition on the function $f(z)$ to be integrated, is that $\lim_{|z|\to\infty}|f(z)|$ falls off at least as fast as $\frac{1}{z^2}$. When this is the case, the integral around the outer semicircle of $\mathcal{C}$ is 0, so the $\int_{\mathcal{C}} f(z)\mathrm{d}z = \int_{-\infty}^{\infty} f(z)\mathrm{d}z$.

We can evaluate the integral using the residue theorem,

$$\int_{\mathcal{C}} f(x)\mathrm{d}z = \sum_{z_p\in\{\text{poles in } \mathcal{C}\}} 2\pi i \,\mathrm{Res}\left(z = z_p, f(z)\right) \ , \tag{A.1}$$

where for simple poles (single roots)

$$\mathrm{Res}\left(z = z_p, f(z)\right) = \lim_{z\to z_p} (z - z_p) f(z) \ , \tag{A.2}$$

and in general for a pole of order n

$$\mathrm{Res}\left(z = z_p, f(z)\right) = \frac{1}{(n-1)!} \cdot \lim_{z\to z_p} \frac{\mathrm{d}^{n-1}}{\mathrm{d}z^{n-1}} \left[(z - z_p)^n \cdot f(z)\right] \ . \tag{A.3}$$

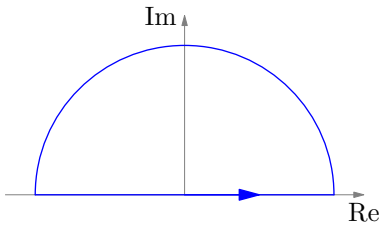


Figure A.1: Integral contour $\mathcal{C}$ enclosing the upper half of the complex plane. If the integrand $f(z)$ goes to zero "quickly enough" as the radius of $\mathcal{C}$ approaches infinity, then the only contribution comes from integration along the real axis (see text for details).

A.2 Integrals

In the following sections I work out derivations for integrals that are important in Section 5.3.

A.2.1 Highly damped integral

$$I = \int_0^\infty \frac{1}{a^2+z^2}\mathrm{d}z = \frac{1}{2}\int_{-\infty}^\infty \frac{1}{a^2+z^2}\mathrm{d}z = \frac{1}{2}\int_{-\infty}^\infty \frac{1}{a^2+(au)^2}\cdot a\mathrm{d}u = \frac{1}{2a}\int_{-\infty}^\infty \frac{1}{u^2+1}\mathrm{d}u\ , \tag{A.4}$$

where $u \equiv z/a$ and $du = dz/a$. The integrand $f(u) \equiv (u^2+1)^{-1}$ has simple poles at $u_p = \pm i$. Using Eq. (A.2),

$$I = \frac{1}{2a}\cdot 2\pi i \operatorname{Res}(u=i, f(u)) = \frac{1}{2a}\cdot 2\pi i \lim_{u\to i}(u-i)\frac{1}{u^2+1} = \frac{1}{2a}\cdot 2\pi i \lim_{u\to i}\frac{1}{u+i} \tag{A.5}$$

$$= \frac{1}{2a}\cdot\frac{2\pi i}{i+i} = \frac{\pi}{2a}\ . \tag{A.6}$$

This result is used in Eq. (5.29).

A.2.2 General case integral

We will show that, for any $(a, b > 0) \in \mathbb{R}$,

$$I = \int_{-\infty}^\infty \frac{1}{(a^2-z^2)^2+b^2z^2}\mathrm{d}z = \frac{\pi}{ba^2}\ . \tag{A.7}$$

First we note that $|f(z)| \to 0$ like $|z^{-4}|$ for $|z| \gg 1$, and that $f(z)$ is even, so

$$I = \int_{\mathcal{C}} \frac{1}{(a^2-z^2)^2+b^2z^2}\mathrm{d}z\ , \tag{A.8}$$

where $\mathcal{C}$ is the contour shown in Fig. A.1.

Because the denominator is of the form A^2+B^2, we can factor it into $(A+iB)(A-iB)$.

$$(a^2-z^2)^2+b^2z^2 = (a^2-z^2+ibz)(a^2-z^2-ibz) \tag{A.9}$$

The roots of $z^2 \pm ibz - a^2$ are given by

$$z_{r\pm} = \pm\frac{ib}{2}\left(1 \pm \sqrt{1 - 4\frac{-a^2}{(ib)^2}}\right) = \pm\frac{ib}{2}\left(1 \pm \sqrt{1 - 4\frac{a^2}{b^2}}\right) = \pm\frac{ib}{2}\left(1 \pm S\right) \ , \tag{A.10}$$

where $S \equiv \sqrt{1 - 4\frac{a^2}{b^2}}$.

To determine the nature and locations of the roots, consider the following cases

- $a < b/2$, overdamped.
- $a = b/2$, critically damped.
- $a > b/2$, underdamped.

In the overdamped case $S \in \mathbb{R}$ and $S > 0$, so $z_{r\pm}$ is purely imaginary, and $z_{r+} \neq z_{r-}$. For any $a < b/2$, we have $0 < S < 1$, so $\mathrm{Im}(z_{r\pm}) > 0$. Thus, there are two single poles in the upper half plane ($z_{r\pm}$), and two single poles in the lower half plane ($-z_{r\pm}$).

In the critically damped case $S = 0$, so $z_{r+} = z_{r-}$, and we have double poles at $\pm z_{r+} = \frac{ib}{2}$.

In the underdamped case S is purely imaginary, so $z_{r\pm}$ is complex, with z_{r+} in the 2^{nd}quarter, and z_{r-} in the 1^{st}quarter. The other two simple poles, $-z_{r-}$ and $-z_{r+}$, are in the $3^{\text{rd}}$and 4^{th}quarters respectively.

Our contour $\mathcal{C}$ always encloses the poles $z_{r\pm}$. We will deal with the simple pole cases first, and then return to the critically damped case.

Over- and under-damped

Our factored function $f(z)$ is

$$f(z) = \frac{1}{(z - z_{r+})(z + z_{r+})(z + z_{r-})(z - z_{r-})} \ . \tag{A.11}$$

Applying Eqs. (A.1) and (A.2) we have

$$I = 2\pi i \left(\mathrm{Res}\left(z = z_{r+}, f(z)\right) + \mathrm{Res}\left(z = z_{r-}, f(z)\right)\right) \tag{A.12}$$

$$= 2\pi i \left(\frac{1}{(z_{r+} + z_{r+})(z_{r+} + z_{r-})(z_{r+} - z_{r-})} + \frac{1}{(z_{r-} - z_{r+})(z_{r-} + z_{r+})(z_{r-} + z_{r-})} \right) \tag{A.13}$$

$$= \frac{\pi i}{z_{r+}^2 - z_{r-}^2} \left(\frac{1}{z_{r+}} - \frac{1}{z_{r-}} \right) = \frac{\pi i}{\left(\frac{ib}{2}(1+S)\right)^2 - \left(\frac{ib}{2}(1-S)\right)^2} \cdot \frac{z_{r-} - z_{r+}}{z_{r+} z_{r-}} \tag{A.14}$$

$$= \frac{-4\pi i / b^2}{(1 + 2S + S^2) - (1 - 2S + S^2)} \cdot \frac{\frac{ib}{2}[(1-S) - (1+S)]}{\left(\frac{ib}{2}\right)^2 (1+S)(1-S)} = \frac{-8\pi/b^3}{4S} \cdot \frac{-2S}{(1-S^2)} \tag{A.15}$$

$$= \frac{4\pi}{b^3(1-S^2)} = \frac{4\pi}{b^3[1 - (1 - 4\frac{a^2}{b^2})]} = \frac{4\pi}{b^3 \cdot 4\frac{a^2}{b^2}} = \frac{\pi}{ba^2} \,. \tag{A.16}$$

Critically damped

Our factored function $f(z)$ is

$$f(z) = \frac{1}{(z - z_{r+})^2 (z - z_{r-})^2} \,. \tag{A.17}$$

Applying Eqs. (A.1) and (A.3) we have

$$I = 2\pi i \,\mathrm{Res}\left(z = z_{r+}, f(z)\right) = 2\pi i \left(\frac{1}{2!} \lim_{z \to z_{r+}} \frac{\mathrm{d}}{\mathrm{d}z} \frac{1}{(z + z_{r+})^2} \right) = \pi i \lim_{z \to z_{r+}} -2 \cdot \frac{1}{(z_{r+} + z_{r+})^3} \tag{A.18}$$

$$= -2\pi i \frac{1}{z_{r+}^3} = -2\pi i \frac{1}{(\frac{ib}{2})^3} = \frac{\pi}{b(\frac{b}{2})^2} = \frac{\pi}{ba^2} \,, \tag{A.19}$$

which matches Eq. (A.16).

This result is used in Eq. (5.39).

Nomenclature

$\langle s(t) \rangle$ Mean (expectation value) of a time-series $s(t)$

$$\langle A \rangle \equiv \lim_{t_T \to \infty} \frac{1}{t_T} \int_{-t_T/2}^{t_T/2} A \mathrm{d}t \ . \tag{A.20}$$

$\equiv$ Defined as (*i.e.* equivalent to).

∞ Infinity

$|z|$ Absolute value (or magnitude) of z. For complex z, $|z| \equiv \sqrt{z\overline{z}}$.

$\overline{z}$ Complex conjugate of z.

$r_{xx}(t)$ Autocorrelation function (Eq. (5.23)).

$S_{xx}(\omega)$ Two sided power spectral density in angular frequency space (Eq. (5.22)).

coth Hyperbolic cotangent,

$$\coth(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}} \ . \tag{A.21}$$

$\dot{s}$ First derivative of the time-series $s(t)$ with respect to time. $\dot{s} = \frac{\mathrm{d}s}{\mathrm{d}t}$.

$\ddot{s}$ Second derivative of the time-series $s(t)$ with respect to time. $\ddot{s} = \frac{\mathrm{d}^2 s}{\mathrm{d}t^2}$.

$\mathcal{F}\{s(t)\}$ Fourier transform of the time-series $s(t)$. $s(f) = \mathcal{F}\{s(t)\} \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t) e^{-i\omega t} \mathrm{d}t$.

$\mathcal{L}$ The Langevin function, $\mathcal{L}(\alpha) \equiv \coth \alpha - \frac{1}{\alpha}$

PSD_f Power spectral density in frequency space

$$\mathrm{PSD}_f(g, f) \equiv \lim_{t_T \to \infty} \frac{1}{t_T} \, 2 \left| \mathcal{F}_f \{g(t)\} (f) \right|^2 \ . \tag{A.22}$$

PSD Power spectral density in angular frequency space

$$\mathrm{PSD}(g, w) \equiv \lim_{t_T \to \infty} \frac{1}{t_T} \, 2 \left| \mathcal{F} \left\{ g(t) \right\} (\omega) \right|^2 \, . \tag{A.23}$$

α The mode unfolding force, $\alpha \equiv -\rho \ln(N_f k_{u0} \rho / \kappa v)$ (Eq. (3.19)).

α' The mode unfolding force for a single folded domain, $\alpha' \equiv -\rho \ln(k_{u0} \rho / \kappa v)$ (Eq. (3.45)).

β Damped harmonic oscillator drag-acceleration coefficient $\beta \equiv \gamma / m$ (Eq. (5.36)).

γ Damped harmonic oscillator drag coefficient $F_{\mathrm{drag}} = \gamma \dot{x}$ (Eq. (5.15)).

γ_e Euler–Macheroni constant, $\gamma_e = 0.577\ldots$.

Δx_u Distance between a domain's native state and the transition state along the pulling direction.

η Dynamic viscocity (Eq. (3.8)).

κ Spring constant (newtons per meter).

π Archmides' constant, $\pi = 3.14159\ldots$. The ratio of a circle's circumference to its diameter.

ρ The characteristic unfolding force, $\rho \equiv k_B T / \Delta x_u$ (Eq. (3.19)).

ρ_b The density of states in the bound state (Eq. (3.46)).

σ Standard deviation. For example, σ is used as the standard deviation of an unfolding force distribution in Eq. (3.19). Not to be confused with the photodiode sensitivity σ_p.

σ_p The linear photodiode sensitivity to cantilever displacement (Fig. 2.1a and Eq. (5.2)).

χ^2 The chi-squared distribution.

ω Angular frequency (radians per second).

ω_0 Resonant angular frequency (radians per second, Eq. (5.36)).

Å Ångström, a unit of length. $1\ \text{Å} = 1 \cdot 10^{-10}$ m.

$\mathbb{R}$ Real numbers.

D Diffusion coefficient (square meters per second).

D_{JS} The Jensen–Shannon divergence (Eq. (3.29)).

D_{LK} The Kullback–Leibler divergence (Eq. (3.30)).

D_{χ^2} Pearson's χ^2 test (Eq. (3.32)).

e Euler's number, $e = 2.718\ldots$.

e^x Exponential function,

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \ldots \ . \tag{A.24}$$

F Force (newtons).

f Frequency (hertz).

f_0 Resonant frequency (hertz).

$f_{\max}$ The frequency of the peak power in PSD_f (Eq. (5.80)).

G_0 The power spectrum of the thermal noise in angular frequency space (Eq. (5.27)).

G_1 The scaled power spectrum of the thermal noise in angular frequency space (Eq. (5.55)).

i Imaginary unit $i \equiv \sqrt{-1}$.

k Rate constant for general state transitions (inverse seconds).

k_B Boltzmann's constant, $k_B = 1.38065 \cdot 10^{-23}$ J/K[218].

k_u Unfolding rate constant.

k_{u0} Unforced unfolding rate constant.

L Contour length in a polymer tension model (Eqs. (3.4) and (3.6)).

l Kuhn length in the freely-jointed chain (Fig. 3.3a and Eq. (3.6)).

l_b The characteristic length of the bound state $l_b \equiv 1/\rho_b$ (Eq. (3.46)).

l_{ts} The characteristic length of the transition state (Eq. (3.46)).

m Effective mass of a damped harmonic oscillator (Eq. (5.15)).

N_f The number of folded domains in a protein chain (Section 3.3.2).

N_u The number of unfolded domains in a protein chain (Section 3.3.2).

P Probability for at least one domain unfolding in a given time step (Eq. (3.11)).

p Persistence length of a wormlike chain (Eq. (3.4))).

$p_m(i)$ The symmetrized probability distribution used in calculating the Jensen–Shannon divergence (Eqs. (3.29) and (3.31)).

Q Quality factor of a damped harmonic oscillator. $Q \equiv \frac{\sqrt{\kappa m}}{\gamma}$ (Eq. (5.74)).

r_{uF} Unfolding loading rate (newtons per second).

T Absolute temperature (Kelvin).

t Time (seconds).

$U_b(F)$ The barrier energy as a function of force (Eq. (3.46)).

$U_F(x)$ Protein free energy along the unfolding coordinate x (joules).

v Cantilever retraction speed in velocity-clamp unfolding experiments.

V_p The vertical photodiode deflection voltage (Fig. 2.1a and Eq. (5.2)).

W Bin width of an unfolding force histogram (Eq. (3.33)).

x Displacement (meters).

ADC Analog to digital converter. A device that digitizes an analog signal. The inverse of a DAC.

AFM Atomic force microscope (or microscopy).

Ala Alanine, an amino acid.

Arg Arginine, an amino acid.

Asn Asparagine, an amino acid.

Asp Aspartic acid, an amino acid.

Bacterial transformation The process by which bacterial cells take up exogenous DNA molecules.

cDNA Complementary DNA.

CLI Command line interface. A textual computing environtment, where the user controls execution by typing commands at a prompt (*cf.* GUI and UI).

Cys Cystine, an amino acid.

DAC Digital to analog converter. A device that converts a digital signal into an analog signal. The inverse of an ADC

DAQ Data acquisition. Although the term only refers to input, it is sometimes implicitly extended to include signal output as well (for controlling experiments as well as measuring results).

DNA Deoxyribonucleic acid.

DTT Dithiothreitol ($C_4H_{10}O_2S_2$), also known as Cleland's reagent[219]. It can be used to reduce disulfide bonding in proteins.

EGTA Ethylene glycol tetraacetic acid

Exogenous DNA DNA that is outside of a cell.

FJC Freely-jointed chain, an entropic spring model (Eq. (3.6)).

force curve Or force–distance curve. Cantilever-force versus piezo extension data aquired during a force spectroscopy experiment (Fig. 2.5b).

FRC Freely-rotating chain (like the FJC, except that the bond angles are fixed. The torsional angles are not restricted).

Gln Glutamine, an amino acid.

Glu Glutamic acid, an amino acid.

Gly Glycine, an amino acid.

GUI Graphical user interface. A graphical computing environment, where the user controls execution through primarily through mouse clicks and interactive menus and widgets (*cf.* CLI and UI).

His Histidine, an amino acid.

I27 Immunoglobulin-like domain 27 from human titin.

Ile Isoleucine, an amino acid.

IMAC Immobilized metal ion affinity chromatography.

Leu Leucine, an amino acid.

Lys Lysine, an amino acid.

MD Molecular dynamics simulation. Simulate the physical motion of atoms and molecules by numerically solving Newton's equations.

Met Methionine, an amino acid.

MPI Message passing interface, a parallel computing infrastructure.

Ni-NTA Nickle nitrilotriacetic acid.

OS Operating system.

PBS Phosphate buffered saline.

PBS Portable batch system, a parallel computing infrastructure. You should be able to distinguish this from the other PBS (phosphate buffered saline) based on the context.

PCR Polymerase chain reaction.

Phe Phenylalanine, an amino acid.

PID Proportional-integral-derivative feedback. For a process value p, setpoint p_0, and manipulated variable m, the standard PID algorithm is

$$m(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{A.25}$$

$$e(t) = p_0 - p \,, \tag{A.26}$$

where e is the error function, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain.

playlist Playlists are containers in Hooke that hold lists of unfolding curves along with some additional metadata.

Pro Proline, an amino acid.

Ser Serine, an amino acid.

SMFS Single molecule force spectroscopy.

tarball A single file containing a collection of files and directories. Created by Tar, tarballs were originally used for tape archives (hence the name), but they are now often used for distributing project source code.

Thr Threonine, an amino acid.

Trp Tryptophan, an amino acid.

Tyr Tyrosine, an amino acid.

UI User interface. What a user uses to interact with a software package (*cf.* GUI and CLI).

Val Valine, an amino acid.

VCS Version control system. A system for tracking project development by recording versions of the project in a repository.

VI Virtual instrument. LabVIEW's analog to functions for encapsulating subroutines.

WLC Wormlike chain, an entropic spring model.

Vita

Contact

W. Trevor King

120 Miller St.

Highlands, NJ, 07732

Tel.: (215) 284 6634

Email: wking@tremily.us

Education

09/2000–05/2004 Colgate University, Hamilton, NY (B.A. in Physics and Math)

09/2004–05/2006 University of Rochester, Rochester, NY (M.A. in Physics)

09/2006–05/2013 Drexel University, Philadelphia, PA (Ph.D. track in Physics)

Publications

Papers William Trevor King, Meihong Su, and Guoliang Yang. Monte Carlo simulation of mechanical unfolding of proteins based on a simple two-state model. *International Journal of Biological Macromolecules*, 46(2):159–166, March 2010. Sawsim is available at `http://blog.tremily.us/posts/sawsim/`.

Talks W. Trevor King. Manipulating combination locks & ray tracing with polarization. Drexel Physics Graduate Student Association, June 2008.

W. Trevor King. Software life-cycles and alphabet soup. Drexel Physics Graduate Student Association, October 2009.

W. Trevor King. Collaborative version control with Git. Software Carpentry boot camp, Columbia University, January 2013.

Posters

W. Trevor King. Effects of cantilever stiffness on unfolding force in afm protein unfolding. Biophysical Society Annual Meeting, February 2008.

W. Trevor King. Simulated mechanical unfolding of single proteins. Drexel CoAS Research Day, April 2008.

W. Trevor King. Experimental estimation of the free energy landscape roughness of protein molecules. Biophysical Society Annual Meeting, March 2009.

W. Trevor King. Open source software in experimental protein unfolding. Drexel CoAS Research Day, April 2010.

W. Trevor King. Flexible parallel simulations and packaging. Drexel CoAS Research Day, April 2011.

W. Trevor King. Thermally calibrating AFM cantilever spring constants. Drexel CoAS Research Day, April 2012.

W. Trevor King. Teaching software carpentry: Better science through science. Drexel CoAS Research Day, April 2013.

Teaching assistant-ships

09/2004–05/2005	General Physics I (Mechanics)
Fall 2006	PHYS 152: Introductory Physics I (Mechanics)
Winter 2007	PHYS 101: Fundamentals of Physics I (Mechanics)
Spring 2007	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)

Spring 2008	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)
Fall 2008	PHYS 154: Introductory Physics III (Electricity, magnetism, and quantum mechanics)
Winter 2009	PHYS 201: Fundamentals of Physics III (Modern physics)
Summer 2009	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)
Fall 2010	PHYS 405: Advanced Computational Physics—Parallel Computing (MPI and CUDA)
Winter 2011	PHYS 101: Fundamentals of Physics I (Mechanics)
Spring 2011	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)
Fall 2011	PHYS 106: Introduction to Scientific Computing (Algorithms in Maple)
Winter 2012	PHYS 305: Computational Physics II (Algorithms in C and C++)
Spring 2012	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)
Fall 2012	PHYS 405: Advanced Computational Physics—Parallel Computing (MPI and CUDA)
Winter 2013	PHYS 305: Computational Physics II (Algorithms in C and C++)
Spring 2013	PHYS 102: Fundamentals of Physics II (Electricity and magnetism)

Research

05/2002–09/2002	Vortex pinning in superconducting thin films. Wrote experimental control software in LabVIEW and fabricated samples using Nb sputtering, photolithography, and wet etching.

05/2003–09/2003 Navier–Stokes simulator. Ported a program simulating two dimensional incompressible fluid flow from MatLab to C.

09/2003–05/2004 Surface tension and lipid phase separation. Designed and began construction of an automated system for making rod pull equilibrium surface pressure measurements on phospholipid vesicle suspensions. Completed drive chain hardware and LabVIEW software for sequentially moving samples to the rod pull mechanism for measurement.

05/2005–05/2006 Superconducting phase qubit quantum computing. Wrote experiment control software in LabVIEW and data analysis software in MatLab. Managed the ^{3}He dilution refrigerator and assembled custom circuit boards.

09/2006–07/2013 Single molecule force spectroscopy via AFM. Designed calibration, experiment control, data processing, and Monte Carlo simulation software in Python and C. Performed necessary hardware troubleshooting and maintenance.

System administration

09/2009–09/2011 Drexel physics department webmaster. Maintained and streamlined department website and assisted with Apache-to-SiteCore migration.

09/2005–Present Home newtork system adminstrator. I run Linux systems (primarily Gentoo) at home. I use my home network to test installation, deployment, and maintenance of the software infrastructure I'd like to see at work, which has lead to experience with many packages (highlights below). Since 2008 I've been publishing my notes on these systems in my blog at `http://blog.tremily.us/`.

Software

Operating systems Linux (Gentoo)

Languages	Python, C, Bash, HTML, XML, CSS, SQLite, Django, JavaScript, LabVIEW, PHP
Creator	pygrader, pyassuan, pgp-mime, quizzer, sawsim, pycomedi, pypiezo, stepper, pypid, pyafm, calibcant, unfold-protein, chemdb, igor, h5config, FFT-tools, update-copyright, mutt-ldap, course-website, problempack, assignment-template, wtk-overlay, dotfiles-framework
Maintainer	rss2email, Bugs Everywhere, drexel-thesis, Hooke
Contributor	Software Carpentry, Git, Pelican, Gentoo catalyst, python-kmod, Comedilib, Kerberos, Linux kernel, Cython, ikiwiki, gnuplot, and more ...
User	Emacs, LaTeX, Nginx, OpenLDAP, Postfix, BIND, CUPS, OpenSSH kerberized NFSv4, mlmmj