THE UNIVERSITY OF CHICAGO

FILAMENT RECYCLING IN 2D ACTIVE NETWORKS AS A MODEL FOR THE ACTOMYOSIN CORTEX

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL & BIOLOGICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF BIOPHYSICAL SCIENCE

BY
WILL MCFADDEN

CHICAGO, ILLINOIS
FEBRUARY 2016

To Claire

ABSTRACT

Actomyosin-based cortical flow is a fundamental engine for cellular morphogenesis. Cortical flows are generated by cross-linked networks of actin filaments and myosin motors, in which active stress produced by motor activity is opposed by passive resistance to network deformation. Continuous flow requires local remodeling through crosslink unbinding and and/or filament disassembly. But how local remodeling tunes stress production and dissipation, and how this in turn shapes long range flow, remains poorly understood. Here, we introduce a computational model for a cross-linked networks with active motors based on minimal requirements for production and dissipation of contractile stress, namely asymmetric filament compliance, spatial heterogeneity of motor activity, reversible cross-links and filament turnover. We characterize how the production and dissipation of network stress depend, individually, on cross-link dynamics and filament turnover, and how these dependencies combine to determine overall rates of cortical flow. Our analysis predicts that filament turnover is required to maintain active stress against external resistance and steady state flow in response to external stress. Steady state stress increases with filament lifetime up to a characteristic time τ_m, then decreases with lifetime above τ_m. Effective viscosity increases with filament lifetime up to a characteristic time τ_c, and then becomes independent of filament lifetime and sharply dependent on crosslink dynamics. These individual dependencies of active stress and effective viscosity define multiple regimes of steady state flow. In particular our model predicts the existence of a regime, when filament lifetimes are shorter than both τ_c and τ_m, in which dependencies of effective viscosity and steady state stress cancel one another, such that flow speed is insensitive to filament turnover, and shows simple dependence on motor activity and crosslink dynamics. These results provide a framework for understanding how animal cells tune cortical flow through local control of network remodeling.

ACKNOWLEDGEMENTS

I'd like to thank my advisor, Ed Munro, Birali Runesha, Francois Robin, Patrick McCall, Jon Michaux, Younan Li, Baixue Yao, Shiladitya Banerjee, the University of Chicago Biophysical Sciences program, Michele Wittels, Artifice, Pete Dahlberg, Adam Hammond, James Crooks, the uchicago myCHOICE program, the Insight Data Science Fellowship Program, Color Genomics, Robbie Sliwinski, my fiancee, Claire Stevenson, her family, David Stevenson and Marilyn Lucas, my brother, Andrew McFadden, and my mom, Deb McFadden. Without all of these people and programs, I wouldn't have made it through this work and come out the other side with something of value.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Biological Context of Cortical Flow

Cortical flow is a fundamental and ubiquitous form of cellular deformation that underlies cell polarization, cell division, cell crawling and multicellular tissue morphogenesis[9, 40, 6, 100, 75, 68]. These flows originate within a thin layer of cross-linked actin filaments and myosin motors, called the actomyosin cortex, that lies just beneath the plasma membrane [81]. Local forces produced by bipolar myosin filaments are integrated within cross-linked networks to build macroscopic contractile stress[70, 5, 43]. At the same time, cross-linked networks resist deformation and this resistance must be dissipated by network remodeling to allow macroscopic deformation and flow. How force production and dissipation depend on motor activity and network remodeling remains poorly understood.

Recent work has also begun to reveal mechanisms for active stress generation in disordered actomyosin networks. Theoretical studies suggest that spatial heterogeneity in motor activity along individual filaments, and asymmetrical filament compliance (stiffer in extension than in compression), are sufficient for macroscopic contraction [51, 50], although other routes to contractility may also exist [50]. Local interactions among actin filaments and myosin motors are sufficient to drive macroscopic contraction of disordered networks *in vitro* [71], and the kinematics of contraction observed in these studies support a mechanism based on asymmetrical filament compliance and filament buckling. However, in these studies, the filaments were preassembled and network contraction was transient, because of irreversible network collapse[1], or buildup of elastic resistance[69], or because network rearrangements (polarity sorting) dissipate the potential to generate contractile force [24, 76, 74, 87]. This suggests that network turnover may play essential role(s) in allowing sustained production of contractile force. Recent theoretical and modeling studies have begun to explore how this might work [39, 61, 104], and to explore dynamic behaviors that can emerge when contractile material undergoes turnover [80, 21]. However, it remains a challenge to understand how force production and dissipation depend individually on the local interplay of network architecture, motor activity and filament turnover, and how these dependencies combine to mediate tunable control of long range cortical flow.

Studies in living cells reveal fluid-like stress relaxation on timescales of 10-100s [64, 40, 9, 41, 25, 3], which is thought to arise through a combination of cross link unbinding and actin filament turnover [19, 18, 81]. Theoretical [11, 67] and computational [47, 54, 52] studies reveal that cross-link unbinding can endow actin networks with complex time-dependent viscoelasticity. However, while cross-link unbinding is sufficient for viscous relaxation (creep) on very long timescales *in vitro*, it is unlikely to account for the rapid cortical deformation and flow observed in living cells [94, 53, 54, 102, 57]. Experimental studies in living cells reveal rapid turnover of cortical actin filaments on timescales comparable to stress relaxation (10-100s) [77, 29, 28, 13, 49]. Perturbing turnover can lead to changes in cortical mechanics and in the rates and patterns of cortical flow[92, 28]. However, the specific contributions of actin turnover to stress relaxation and how these depend on network architecture remain unclear.

1.2 Active fluid models of pattern formation and cortical flow

One successful approach to modeling cortical flow has relied on coarse-grained phenomenological descriptions of actomyosin networks as active fluids, whose motions are driven by gradients of active contractile stress and opposed by an effectively viscous resistance[64]. In these models, spatial variation in active stress is typically assumed to reflect spatial variation in motor activity and force transmission[8], while viscous resistance is assumed to reflect the internal dissipation of elastic resistance due to local remodeling of filaments and/or cross-links[81, 19]. Models combining an active fluid description with simple kinetics for network assembly and disassembly, can successfully reproduce the spatiotemporal dynamics of cortical flow observed during polarization [64], cell division [90, 80], cell motility [45, 62] and tissue morphogenesis [4]. However, it remains a challenge to connect this coarse-grained description of cortical flow to the microscopic origins of force generation and dissipation within cross-linked actomyosin networks.

Much of the recent theoretical interest in cortical flow centers on the biological

context of pattern formation. In particular, in the developing *C. elegans* embryo, an initial small asymmetry is magnified by cortical flow to produce a global polarized cell[68]. This global polarization from mechanical flow constitutes a mechanism of macroscopic pattern formation beyond the conventional Turing reaction-diffusion mechanism[42]. Therefore, it becomes all the more important to elucidate the specific micromechanical mechanisms that make it possible for a system to exhibit acive fluid-like properties.

To understand how cells exert physiological control over cortical deformation and flow, or to build and tune networks with desired properties *in vitro*, it is essential to connect this coarse-grained description to the microscopic origins of force generation and dissipation within cross-linked actomyosin networks. Both active stress and effective viscosity depend sensitively on microscopic parameters including densities of filaments, motors and cross-links, force-dependent motor/filament interactions, cross-link dynamics and network turnover rates. Thus a key challenge is to understand how tuning these microscopic parameters controls the dynamic interplay between active force generation and passive relaxation to control macroscopic dynamics of cortical flow.

1.3 Rheology and Theory of Semi-flexible Cross-linked Networks

Cross-linked networks of semi-flexible polymers are a class of materials with poorly understood but highly interesting properties. Early studies of semi-flexible polymer networks reconstituted *in vitro* revealed novel, nonlinear rheology, spurring interest from materials scientists[12]. Cross-linked networks of cytoskeletal polymers have been a subject of great interest to biologists because of their importance as structural components of cells[27, 82].

On shorter timescales, the response of cross-linked polymer networks to applied stress can be well-described theoretically in terms of purely elastic mechanical resistance. On longer timescales, the network's elastic resistance begins to give way to a viscous relaxation of stored stress, but the mechanisms that govern this viscous

relaxation remain poorly understood. It is important to understand the mechanism behind this long timescale relaxation of cross-linked polymer networks both for understanding their novel material properties as well as understanding how this effect may govern physiologically important cellular processes[85].

For *in vitro* reconstitutions, this viscous relaxation is thought to result from transient unbinding and rebinding of intermolecular cross-links[94, 11]. However, there is still no clear understanding of how local relaxations of network connectivity would give rise to a global viscous relaxation. In our work, we wish to expand upon a well-established mechanical picture of cross-linked semi-flexible polymer networks to incorporate slippage of cross-links over longer timescales.

1.4 Short Timescale Mechanics of Cross-linked Actin Filament Networks

Early *in vitro* studies of cross-linked actin filament networks revealed strikingly different elastic behaviors compared to the already well-understood flexible polymer gels [88]. The complexity of these behaviors drove a surge in both experimental and theoretical studies of semi-flexible networks. For a comprehensive review of this field we recommend [12], but we will shortly repeat some important milestones here.

1.4.1 Theories of Semi-flexible Filament Networks

Diversity and discrepancy in observations led a drive toward systematic *in vitro* experimental explorations of the rheology of cross-linked semi-flexible polymer networks at short timescales. In studies with rigid irreversibly cross-linked networks, it was found that differences in network structure could lead to remarkably different elastic moduli, suggesting distinct phases of mechanical response [33]. These discoveries in turn begat theoretical work on the basic implications of the semi-flexible nature of filaments on network mechanics.

Prior work on the basic physics of individual semi-flexible polymers [63, 22], and comprehensive theories of semi-flexible filament solutions, [66] laid a groundwork

for theoretical considerations of cross-linked networks. Beginning with the so-called mikado model descriptions[37, 99], it was determined that there should exist a minimum rigidity percolation threshold, and that the connectivity of the network determined whether the mechanical response was dominated by non-affine bending or affine stretching of filaments. Continuing to more explicit theories[86], the mechanics of rigidly cross-linked networks were shown to be well-described in terms of purely elastic stretching of filaments between cross-linked points.

1.4.2 Incorporating Effects of Cross-link Compliance

Despite the success of the theory for rigid cross-links, early studies showed that surprising qualitative differences in mechanical response could be traced to differences in the chosen cross-linker[96, 55]. In addition, many studies using more compliant cross-linkers showed that cross-linker compliance could give rise to different nonlinear rheological properties on short timescales[31, 32, 44, 56]. Making matters even more complicated, ongoing research has begun to uncover added complexity from more highly complex issues such as filament bundling[67, 17]and the effects of active cross-linking by molecular motors[48].

While theorists have built a number of largely successful models that help characterize different aspects of the cross-link dominated response[38, 101, 10], the diversity of behaviors of these networks makes a precise yet general theory more difficult.

1.5 Long Timescale Stress Relaxation from Transient Cross-link Unbinding

At long timescales, the purely elastic behavior of cross-linked networks gives way to fluid-like stress relaxation. Additionally, fluid-like flows have been observed in a number of cellular processes[64, 40, 9, 41, 25, 3]. In *in vitro* studies, long timescale creep behaviors are thought to arise predominantly from the transient nature of filament binding for most biologically relevant cross-linkers[53, 54, 102, 57]. While the importance of cross-link dynamics in determining the mechanical response of semi-flexible

polymer networks has been known for at least 20 years[94], there is still a gap in our understanding of how microscopic cross-link unbinding relates to viscous flows.

1.5.1 Models of Stress Relaxation with Transient Cross-links

The dependence of network rheology on cross-link unbinding is an active subject of theoretical research[67]. Several theoretical methods have addressed cross-link binding and unbinding directly [11, 67] in analytical approaches that allowed well-constrained fits for specific cross-linkers. These theories have therefore focused conceptually at the level of the cross-linked filament and were extended analytically to macroscopic networks. In another approach, modelers have taken cross-links as extended springlike structures [47] that are able to bind and unbind in simulated filament networks. Finally, other more ambitious simulations have even sought to interrogate the effects of cross-link unbinding in combination with the more complex mechanics of filament bundles[54, 52].

Ultimately, the complexity of the many theoretical approaches that have been applied to this problem have made it difficult to distinguish what, if any, core physical mechanisms may be sufficient to explain the observed forms of stress relaxation. We believe that serious qualitative understanding can be generated by focusing on some of the common elements exhibited in the aforementioned literature.

1.6 Goals of Thesis

1.6.1 Elucidating the Importance of Filament Turnover

The fundamental importance of actin filament turnover in setting the mechanical properties of a cell has been known to biologists for decades [16]. Nevertheless, theoretical investigators have only begun to explore the role of turnover quite recently [39, 61, 104]. A major factor contributing to the slow advancement of our understanding of the role of turnover in cell mechanics is the difficulty in experimentation. However, recent advances in experimental techniques of reconstituted systems should make turnover a viable avenue of study in the near future.

Another major factor preventing the advancement of our understanding of turnover's impact on cell mechanics and dynamics is the complexity already inherent to our computational models of polymer systems at many length and time scales[60]. In this work we wish to accelerate the interest in filament turnover as a fundamental contributor to cellular mechanics. To do so, we will deemphasize the particularities of the polymer and cross-linker models and create a simplified generic model. With this simplified model we will then be in a position to analyze the fundamental role of filament turnover.

1.6.2 Novelty of Cross-link Slip Approach

We introduce a coarse-grained representation of filament cross-linking in which cross-linked filaments are able to slide past each other as molecular bonds form and rupture, akin to coarse-grained models of molecular friction[93, 84, 26]. This drag-like coupling has been shown to be an adequate approximation in the case of ionic cross-linking of actin[95, 14], and can be found in the theoretical basis of force-velocity curves for myosin bound filaments[2]. We propose that it will form a suitable bulk approximation in the presence of super molecular cross-links as well.

Importantly, this simplification allows us to extend our single polymer models to dynamical systems of larger network models for direct comparison between theory and modeling results. This level of coarse graining will therefore make it easier to understand classes of behavior for varying compositions of cross-linked filament networks. In addition, it allows us to compute a new class of numerical simulations efficiently, which gives us concrete predictions for behaviors in widely different networks with measurable dependencies on molecular details.

1.6.3 Bridging the Theoretical Gap Between Active Fluids and Polymer Models

The goal of this work is to build a computational bridge between the microscopic description of cross-linked actomyosin networks and the coarse grained macroscopic description of an active fluid. We seek to capture the essential microscope features

(dynamic cross-links, active motors and semi flexible actin filaments with asymmetric compliance and continuous filament recycling), but in a way that is sufficiently simple to allow systematic exploration of how parameters that govern network deformation and flow in an active fluid theory depend on microscopic parameters. To this end, we introduce several coarse-grained approximations into our representation of filament networks. First, we represent semi-flexible actin filaments as simple springs with asymmetric compliance (stronger in extension than compression). Second, we replace dynamic binding/unbinding of elastic cross-links with a coarse-grained representation in terms of molecular friction [93, 84, 26], such that filaments can slide past each other against a constant fictional resistance. Third, we used a similar scheme to introduce active motors at filament crossover points with a simple linear force/velocity relationship, and we introduce dispersion of motor activity by making only a subset of filament overlaps active [2]. Finally, we model filament turnover by allowing entire filaments to appear and disappear with a fixed probabilities per unit time. Importantly, these simplifications allow us to extend our single polymer models to dynamical systems of larger network models for direct comparison between theory and modeling results. This level of coarse graining will therefore make it easier to understand classes of behavior for varying compositions of cross-linked filament networks. In addition, it allows us to compute a new class of numerical simulations efficiently, which gives us concrete predictions for behaviors in widely different networks with measurable dependencies on molecular details.

We use this model first to characterize the passive response of a cross-linked network to externally applied stress, then the buildup and maintenance of active stress against an external resistance, and finally the steady state flows produced by an asymmetric distribution of active motors in which active stress and passive resistance are dynamically balanced across the network. Our results reveal how network remodeling can tune cortical flow through simultaneous effects on active force generation and passive resistance to network deformation.

CHAPTER 2
IMPACT OF FILAMENT RECYCLING ON CORTICAL FLOW IN ANIMAL CELLS

2.1 Measuring *in vivo* turnover rates with smPReSS

Dynamic remodeling of the embryonic cell surface is essential for the control of cell polarity, division, shape change and movement during early development. This remodeling involves the local movement and exchange of proteins at the interface between the plasma membrane and the actin-rich cell cortex. Quantifying these processes in vivo remains a challenge. One promising approach is single-molecule imaging combined with SPT, which can yield quantitative measurements of local mobilities, binding states and exchange kinetics that are inaccessible with ensemble measurements[59]. Combining these approaches with genetic tools in a classical model organism could be a powerful way to investigate subcellular dynamics in embryonic cells.

One key impediment to performing such experiments has been the lack of simple and reliable methods for tunable and noninvasive labeling of target molecules. Optimal labeling densities are different for each target and must balance the need for high-density sampling of molecular behavior against practical requirements for accurate and unbiased single-molecule detection and tracking. Toward the goal of measuring molecular remodeling and actomyosin dynamics, we developed a simple, versatile and minimally invasive method[78] for single-molecule imaging at the cell surface in C. elegans embryos of transgenic strains expressing GFP-tagged actin proteins. In particular, we showed how these data could be used to extract quantitative information about surface density and turnover.

2.1.1 smPReSS measurement technique

We sought to exploit the intrinsic exchange dynamics of surface-associated proteins to create a self-renewing pool of GFP-tagged single molecules at the cell surface that could be followed over time to measure mobility and turnover. To establish a kinetic basis for this approach, we consider a GFP-tagged protein that exchanges dynamically between the bulk cytoplasm and a region of the cell surface, which is observed by near-TIRF microscopy (Fig. 2.1a). The number of molecules N(t) within this region over time is governed by

$$\frac{dN}{dt} = k_{app} - (k_{off} + k_{ph})N \tag{2.1}$$

where k_{app} is an observable appearance rate that depends on the cytoplasmic concentration of GFP-tagged protein ($k_{app} = k_{on} \times Y$, where Y is the cytoplasmic concentration) and the nature of the binding process. k_{off} and k_{ph} are pseudo-first-order rate constants such that $k_{off} \times N$ is the rate (in molecules per second) at which particles disappear owing to unbinding or disassembly, and $k_{ph} \times N$ is the rate (in molecules per second) at which they disappear owing to irreversible photobleaching (Fig. 2.1a). Prior to illumination, $k_{ph} = 0$ and the steady state density is $N_{ss} \approx k_{app}/k_{off}$. During illumination, k_{ph} becomes nonzero; if the cytoplasmic pool were infinite, the system would approach a new steady-state density given by $N_{ssobs} = k_{app}/(k_{off} + k_{ph})$, which is a fixed fraction $f = k_{off}/(k_{off} + k_{ph})$ of the initial unobserved value (Fig. 2.1b). In practice, irreversible photobleaching will gradually deplete a finite cytoplasmic pool. A variant of the kinetic model that accounts for this depletion predicts a biphasic response to the onset of illumination: a fast relaxation toward N_{ssobs} followed by a slower decay toward 0, at a rate that depends on the photobleaching rate and the size of the cytoplasmic pool (Fig. 2.1b). For a given target molecule, k_{off} is fixed. However, k_{ph} depends on imaging conditions (i.e., the intensity and duty ratio of the laser illumination), and k_{app} can be adjusted by tuning the initial size of the GFP-tagged pool. Thus, co-tuning these factors should make it possible to target a desired density N_{ssobs} at quasi-steady state for a range of imaging conditions.

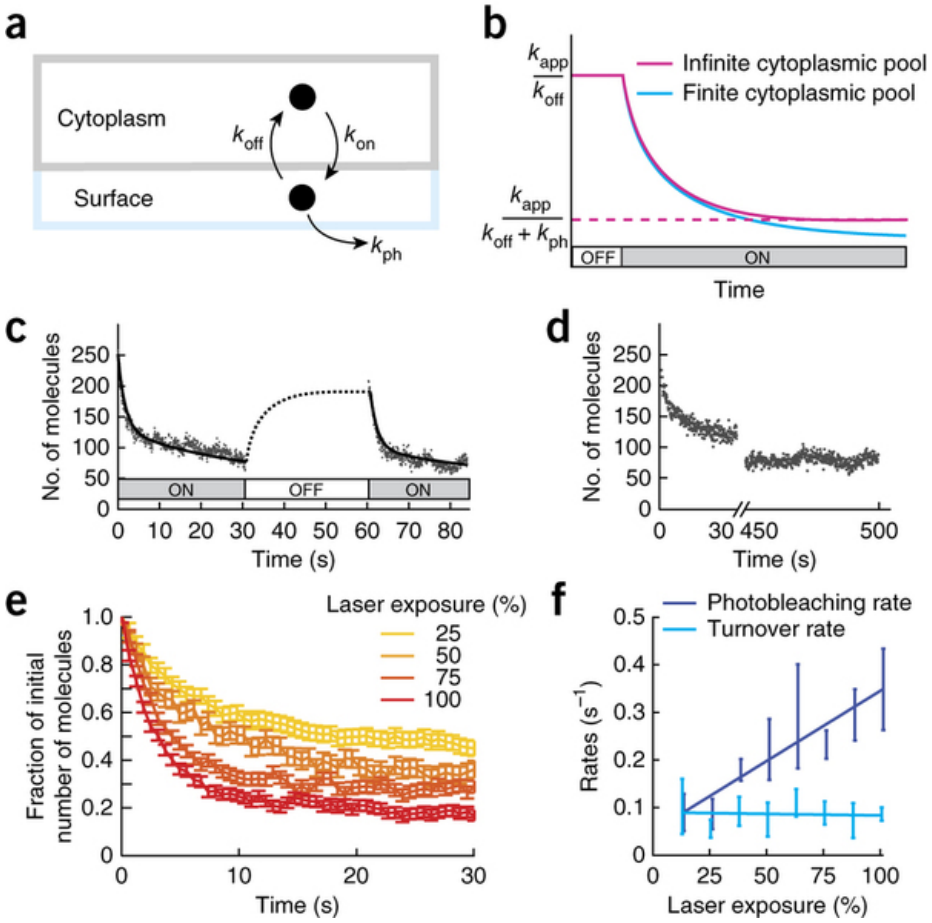


Figure 2.1: smPReSS measurement technique. **a)** The basic kinetic principle: during imaging, the level of surface-associated proteins is set by a dynamic balance of appearance (binding or assembly) at an observable rate k_{app}, disappearance (unbinding or disassembly) at a per-molecule rate k_{off}, and photobleaching at a per-molecule rate k_{ph}. **b)** Predicted response of an initially unobserved cell at steady state to a step change in illumination. For an infinite cytoplasmic pool, the surface density relaxes to a new illuminated steady state. For a finite cytoplasmic pool, fast relaxation is accompanied by a slower decay caused by irreversible photobleaching. **c)** Fast relaxation to a quasi-stable density during illumination is rapidly reversed when the laser is turned off. **d)** Biphasic response for GFP::actin under illumination conditions that allow accurate single-molecule detection and tracking. Time from t = 30-450 s is discontinuous. **e)** Surface density of GFP::actin vs. time at various laser exposures, shown as a fraction of the initial unobserved density. Error bars, s.e.m. (n = 7, 9, 6, 7 embryos for laser exposures from low to high). textbff) Estimates of per-molecule turnover (k_{off}) and photobleaching (k_{ph}) rates as a function of laser exposure. Error bars, s.d. (n = 12, 7, 8, 9, 7, 6, 7, 7 embryos, low to high laser exposure). Solid lines show a linear regression against the data. 100% laser power $\approx 1.6\mu W/\mu m^2$.

To test this approach, we chose a representative strain that expressed GFP::actin. Actin monomers exchange dynamically with the cell surface through local filament assembly and disassembly, and on the basis of previous work, we expected GFP::F-actin to be relatively immobile at the cell surface and to turn over in a few tens of seconds. In contrast, fast-diffusing monomers of GFP::G-actin should produce highly blurred images and thus escape detection under our imaging conditions6. Focusing again on the polarity maintenance phase, we reduced densities to single-molecule levels, allowed the system to equilibrate unobserved and then recorded data for a range of laser intensities and exposure times (Fig. 2.1ce). We observed the predicted biphasic response to a step change in illumination: a rapid initial decrease in the number of molecules to a quasi-stable value followed by slower decay (Fig. 2.1c,d). Notably, the initial decrease was reversed with equally rapid kinetics when the laser was turned off (Fig. 2.1c), a result confirming that the quasi-steady state is set by a dynamic balance of exchange and photobleaching. We could therefore obtain robust estimates for effective dissociation and photobleaching rate constants k_{off} and k_{ph} by fitting the predicted biphasic kinetics to the change in single-molecule density over time; for each strain, we optimized fitting conditions by adjusting laser exposure. As expected, estimates of k_{ph} varied linearly with laser exposure, whereas estimates of k_{off} remained fixed over a range of exposures (Fig. 2.1f). Our estimate of k_{off} for actin::GFP of $koff = 0.10.07s^1$ demonstrates that actin filaments astonishingly short-lived, with a typical lifetime of only approximately 10s.

2.1.2 Measurements of turnover rate in dividing C. elegans cortex

We measured spatiotemporal modulation of actin assembly and disassembly during the first cell division in the C. elegans embryo (Fig. 2.2a). Both actin assembly and disassembly are thought to be modulated during cytokinesis, but their relative contributions to actin filament accumulation are not well understood. We performed these experiments in embryos depleted of nonmuscle myosin II to remove the confounding effects of surface deformations and flow and to remove myosin-dependent effects on turnover. We verified strong depletion of myosin II by the complete fail-

ure of cytokinesis and a complete absence of local surface deformation and cortical flow during early anaphase. In the case of GFP::actin, the turnover rates measured by smPReSS ($k_{off} \approx 0.1s^1$) were similar to the photobleaching rates ($k_{ph} \approx 0.1s^1$) required for accurate particle tracking and agreed well with estimates of k_{off} from particle tracking (Fig. 2.2b), suggesting that in this case, we could use SPT to measure spatiotemporal variations in turnover, with the smPReSS measurements of k_{ph} used to correct for photobleaching. We measured roughly uniform values for k_{app} and k_{off} along the anterior-posterior axis during maintenance phase (Fig. 2.2ce), a result consistent with a lack of cortical asymmetry at this stage in myosin-depleted embryos. During the transition into anaphase, when the contractile ring normally assembles, we observed a net increase in actin density at the equator as anticipated, and we also observed a net decrease at the poles (Fig. 2.2a,c). Surprisingly, these changes involved strong modulation of both filament assembly and disassembly (Fig. 2.2d,e). The equatorial increase was associated with a small increase in assembly rate and a larger decrease in turnover, whereas the polar decrease in density was associated with both a decrease in assembly and an increase in turnover. Thus, our ability to simultaneously resolve assembly, disassembly and density revealed an unappreciated dimension to the control of cortical microfilaments during cell division.

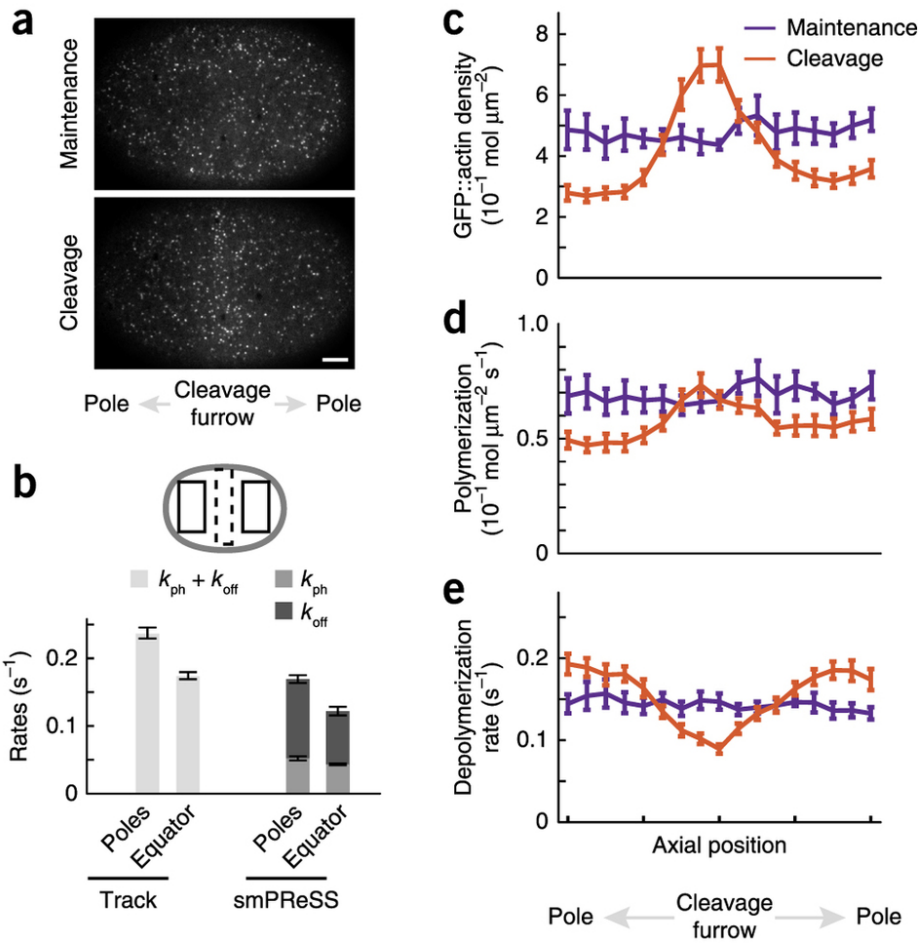


Figure 2.2: Measurement of cortical actin lifetimes. **a)** Near-TIRF micrographs of GFP::actin during the polarity maintenance phase (top) and cleavage (bottom). **b)** Measurements of the indicated turnover rates at the equator and poles during anaphase using tracking (left) or smPReSS (right). Schematic indicates the equatorial (dashed box) and polar (solid boxes) regions in which the measurements were made. For tracking, the sum of koff and kph is displayed; for smPReSS, the values for k_{ph} and k_{off} are stacked. **c-e)** Spatial variation in actin density and turnover kinetics during maintenance phase and cleavage measured by tracking and binned along the antero-posterior axis. **c)** Cortical density. **d)** Polymerization rate. **e)** Depolymerization rate (instantaneous disappearance rate minus estimated photobleaching rate). In b-e, error bars indicate cell-to-cell s.e.m. (n = 7 embryos, maintenance, and n = 16 embryos, cleavage). Scale bar, 5μm.

2.2 Active Fluid Model Fitting of Cortical Flow Measurement in *C. elegans* embryos

To ascertain the manner in which active stress and effective viscosity depend upon actin and myosin densities, I studied cortical flow in the *C. elegans* single cell embryo during polarity maintenance phase. The single cell *C. elegans* embryo poses a viable model system wherein to determine how active stress and effective viscosity depend on constitutive properties of the cytoskeleton. During the *C. elegans* first division cycle, cell fate determinants and small regulatory GTPases are segregated between distinct anterior and posterior domains, and these asymmetries are actively sustained through regulatory feedback for 5 minutes during the polarity maintenance phase. [11] As displayed in Fig. 2.3 this system is characterized by a spatially varying distribution of myosin motors in the presence of persistent unidirectional cortical flow from the posterior pole towards a band of cytoskeletal aggregation near the midline.

In addition to the steady state velocity profile, the *C. elegans* embryo provide a number of advantages that made these measurements easier. The events of polarization are highly stereotyped and reproducible, allowing for comparison between experiments. Transgenic strains expressing fluorescently-tagged variants of myosin-II and F-actin binding proteins (e.g. utrophin and moesin), allowing for simultaneous visualization of F-actin and myosin by fluorescence microscopy. In addition, the superficial cortical layer allows for high quality microscopy necessary for the experiments, and the cell geometry allows simplification to a 2D system. Finally, an important factor necessary for constraining the theory is a fine level of control over the densities of both actin and myosin, which can be supplied in the C. elegans embryo by RNAi depletion of nearly any gene expressed in the germline. [12]

2.2.1 Obtaining spatially resolved simultaneous measurement of actin and myosin densities.

Rationale: Actin and myosin are assumed to be the drivers of cortical flow and their density profiles and gradients are therefore important variables necessary to

parametrize the functions of active stress and effective viscosity upon which the theory is based.

To obtain the density profile, I used interleaved HILO microscopy timelapses to obtain simultaneous images of myosin and actin in the C. elegans embryo during the single celled polarity maintenance phase. Next, I coarse grained the image using image processing blurs or a fixed square bin so that microscopic inhomogeneity is removed from the measurement. Limiting the analysis to one dimension improved the measurement quality and made the mathematics simpler so an average density taken along cross section lines running perpendicular to the embryo AP axis was computed. The example in Fig. 2.3 displays a typical fluorescence image of the myosin localization in the embryo along with the AP projection axis. The myosin fluorescence profile changes gradually over the course of five minutes with a period of particular stability lasting roughly two minutes.

2.2.2 Obtaining spatially resolved measurement of cortical flow velocities.

To obtain a flow profile, I analyzed the myosin interframe displacements with particle image velocimetry (PIV). I averaged a set of contiguous frames from the videos prior to blurring or binning in order to obtain a more clear image and provide a larger flow displacement between PIV analyzed frames. The PIV algorithm is assumed to maintain a nearly constant absolute margin of error so maximizing interframe flow distance while still maintaining consistent results minimizes relative error. I selected a region of interest that remained consistent throughout all experiments and encompassed the entirety of the dilating and contracting regions.

The PIV algorithm was an FFT multipass with window deformation and a post-processing clean up that rejects and interpolates any velocity with a speed greater than 3 times the standard deviation.[13] Only the on axis motion is relevant at this time so the output velocity vectors will be projected onto a line parallel to the AP axis. This component of the flow velocity was averaged along the cross sections running perpendicular to the AP axis as before. The diagram below illustrates a typical

PIV flow analysis on the left. On the right, the projection of the flow velocity onto the AP axis is shown to vary slowly in magnitude over the course of 5 minutes and to maintain the same overall shape.

2.2.3 Using the data to constrain a basic active fluid model

I compared these measurements to the theoretical predictions by setting up a few non-linear functional fits of the velocity, effective viscosity and active stress as a function of myosin density using a nonlinear least-squares fitting solver with the equation

$$v(x) = C_1 e^{x/l} + C_2 e^{-x/l} + \int_a^b \frac{\delta\mu}{\delta x}(e^{(x'-x)/l} + e^{(x-x')/l} \tag{2.2}$$

where l is a length scale of force falloff and the constants C_1, C_2 are constrained by the boundary conditions at the points a and b in the embryo, $v(a) = v_a$ and $v(b) = v_b$. See [Bois et al] for more details on this equations derivation.

Data fitting was carried out in MATLAB by passing a starting guess for the fitting coefficients to lsqnonlin along with the function defined above.

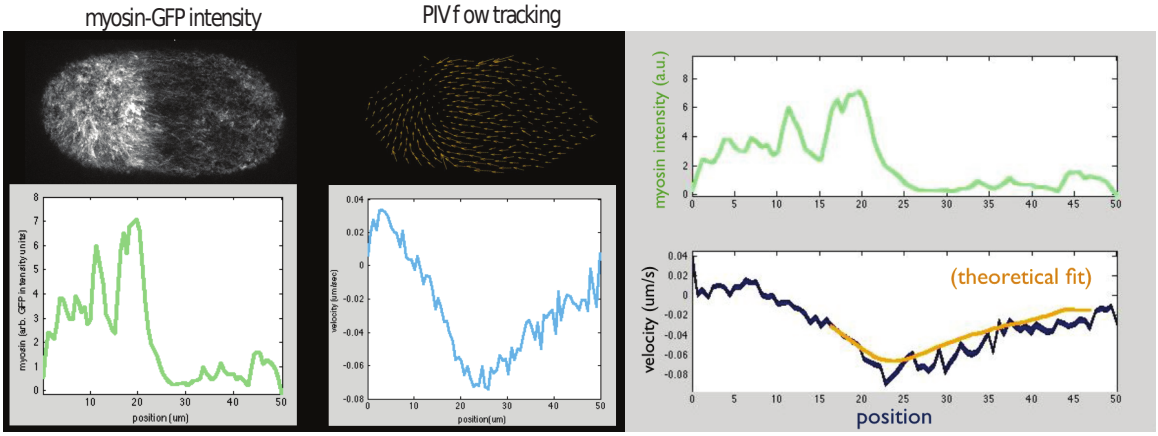


Figure 2.3: Measurement of myosin intensity and flow profile. These measurements fit a theoretical model of active fluid flow.

The results of this analysis showed that, indeed, an active fluid function akin to that used in [Mayer and Grill] is capable of explaining the observed flow profiles in *C. elegans* embryos.

2.2.4 Disruption of Flow by Myosin Depletion

The above experiment only sampled a limited region of myosin-actin state space if only analyzed in wild type. Using only data from this limited region does not uniquely determine the form of the functions that satisfy the equations of motion. I needed to test outside this range by varying the concentration of actin and myosin independently in order to adequately constrain the fitted equations of active stress and effective viscosity.

RNAi depletion is a common practice used to decrease the cellular concentration of endogenously expressed proteins in C. elegans. By feeding ssRNA containing *E. coli* to mother nematodes, the mRNA in the nematode germline is progressively silenced and the remaining protein decays away. I used this technique to lower the expression of myosin regulatory light chain kinase (MRCK), which is an upstream regulator of myosin. Overall myosin density decreases with a depletion of MRCK. We assume that the degree of depletion can be assessed by integrating the myosin density profile over the length of the cell.

Gene depleted embryos were imaged and processed in the exact same manner as described above for wild type.

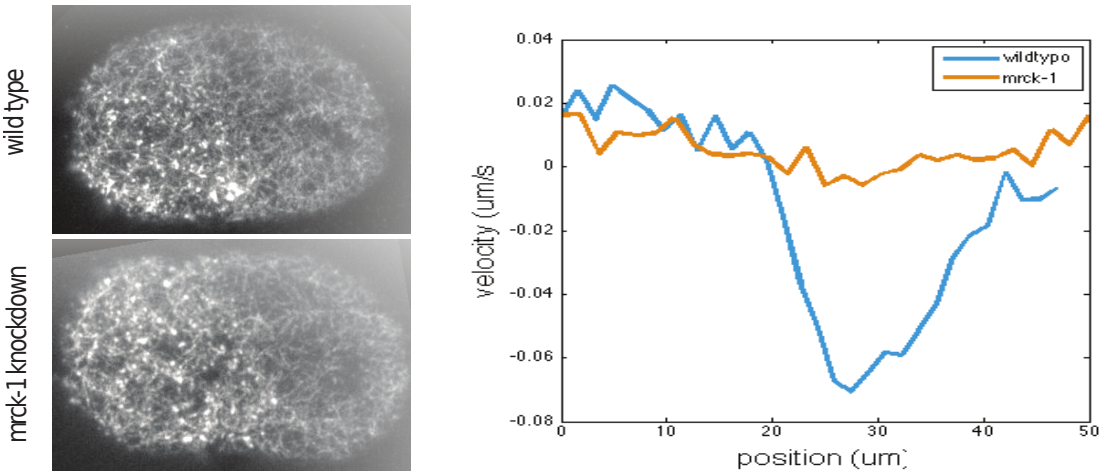


Figure 2.4: Measurement of actin distributions and flow profiles for mrck-1 knockdown embryos. MRCK-1 is an upstream regulator of myosin.

As shown in Fig. 2.4, the actin densities were largely unaffected. Nevertheless, the

myosin density (not shown) and flow profile (Fig. 2.4, right) were reduced. Progressive depletion of MRCK also showed a graded response to myosin depletion, indicating that myosin activity imparts internal force in a dose dependent manner.

2.3 Stabilization of Filament Turnover with Jasplakinolide Treatment

Jasplakinolide is a small molecule inhibitor of actin depolymerization. When treated with jasplakinolide, actin filaments can be stabilized to have very long filament lifetimes. To study the effect of filament stabilization on cortical flows, Jon Michaux treated embryos with jasplakinolide and imaged the actin cytoskeleton with utrophin::GFP. The results were striking, and clearly showed that loss of turnover resulted in disrupted flows. Moreover, in some cases, the treatment didn't just result in a stalling of flows, but also led to a tearing of the cortex followed by further transient contraction. Nevertheless, all embryos with jasplakinolide treatment eventually stalled and failed to continue with normal polarization and division.

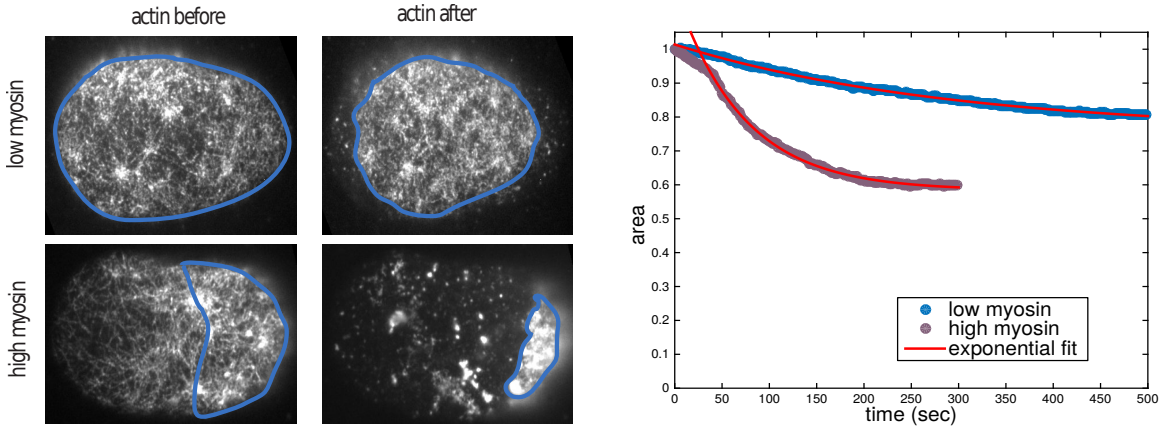


Figure 2.5

I made an attempt to quantify the degree of contraction possible in jasplakinolide treated embryos using hand tracking of fiducial markers in Jon's videos. As can be seen in Fig. 2.5, whether the cortex stalled or tore, the resulting strain asymptotically approached some limiting strain. This is in sharp contrast to the persistent strains

($\gamma > 1$) possible in the steady state flows found when turnover is present.

CHAPTER 3
MODELING 2D ACTIVE NETWORKS WITH RECYCLING

Our goal was to construct a minimal model that is detailed enough to capture essential microscopic features of cross-linked actomyosin networks (actin filaments with asymmetric compliance, dynamic cross-links, active motors and and continuous filament turnover), but simple enough to explore, systematically, how these microscopic features control macroscopic deformation and flow. We focus on 2D networks because they capture a reasonable approximation of the quasi-2D cortical actomyosin networks that govern flow and deformation in many eukaryotic cells[64, 15], or the quasi-2D networks studied recently *in vitro* [71, 83].

3.1 Mathematical Summary of Modeling Methodology

3.1.1 Schematic Overview

Fig. 3.1 provides a schematic overview of our model's assumptions. We model each filament as an oriented elastic spring with relaxed length l_s. The state of a filament is defined by the positions of its endpoints $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$ marking its (-) and (+) ends respectively. The index i enumerates over all endpoints of all filaments. We refer to the filament connecting endpoint i and i+1 as filament i, and we define $\hat{\mathbf{u}}_\mathbf{i}$ to be the unit vector oriented along filament i from endpoint i to endpoint i+1.

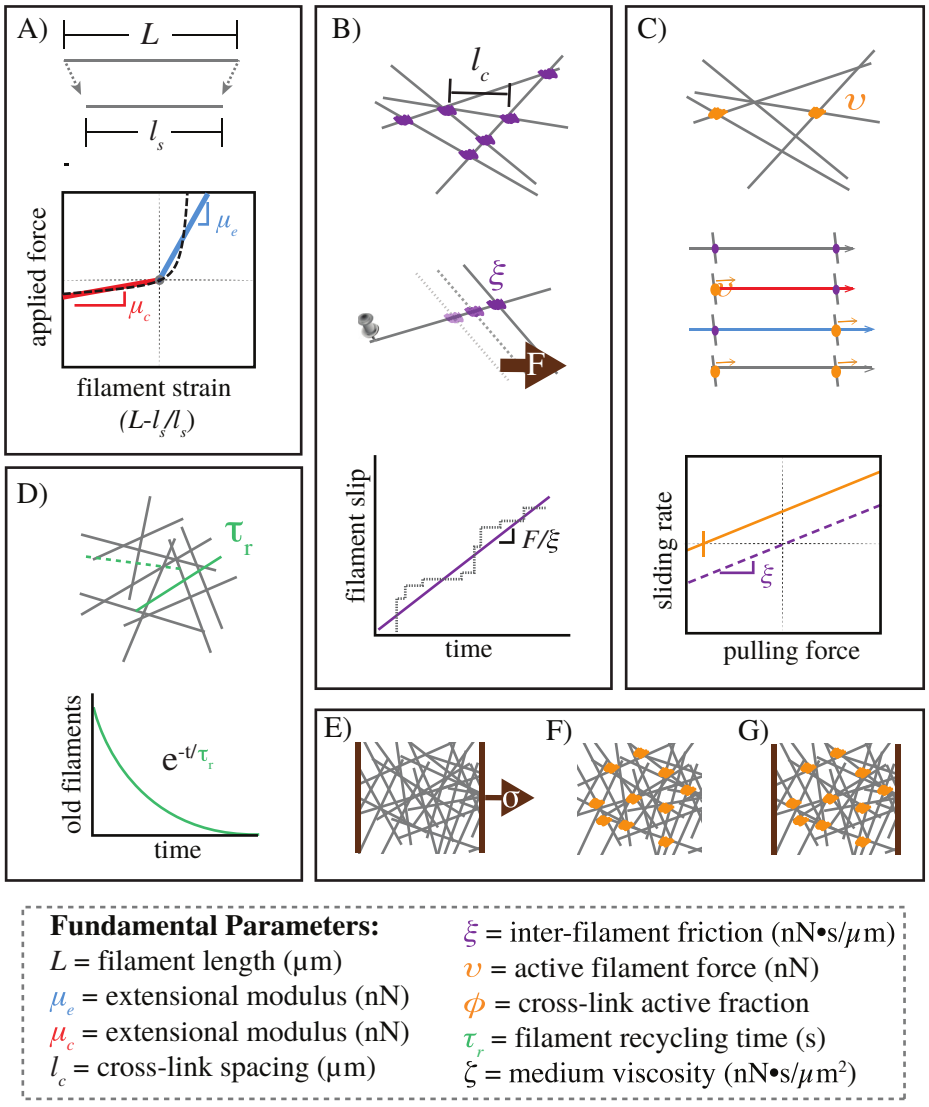


Figure 3.1: Schematic overview of modeling framework and assumptions. **A)** Filaments are oriented linear springs that are stiffer in extension than in compression. **B)** Cross-linking occurs at all filament crossings; we represent cross link resistance as an effective drag, proportional to the relative velocity of the overlapping filaments. **C)** We represent motor activity as a linear force-velocity relationship with a fixed force at zero velocity directed towards a filament's (-) end. We implement spatial heterogeneity by imposing motor activity at a fixed fraction of filament crossover points, resulting in variation in the magnitudes of compressive vs extensile vs translational forces along individual filament segments. **D)** Whole filaments disappear at a constant rate; new filaments appear with random positions and orientations at the constant rate per unit area, such that entire network refreshes on a characteristic timescale τ_r. **e-g)** Three different simulation scenarios: **E)** Passive response to uniaxial stress, **F)** Free contraction of an active network and **G)** Isometric contraction against a fixed boundary.

3.1.2 Asymmetric filament compliance

We assume (Fig. 3.1A) that local deformation of filament i gives rise to an elastic force:

$$\mathbf{F}^{\mu}_{\mathbf{i,i+1}} = \mu\gamma_i\hat{\mathbf{u}}_\mathbf{i} \tag{3.1}$$

where $\gamma_i = (|\mathbf{x_{i-1}} - \mathbf{x_i}| - l_s)/l_s$ is the strain on filament i, and the elastic modulus μ is a composite quantity that represents both filament and cross-linker compliance as in the effective medium theory of Broederz and colleagues [10]. To model asymmetric filament compliance, we set $\mu = \mu_e$ if the strain is positive (extension), and $\mu = \mu_c$ if the strain is negative (compression). The total elastic force on a filament endpoint **i** can be written as:

$$\mathbf{F^{elas}_i} = \mathbf{F}^{\mu}_{\mathbf{i,i+1}} - \mathbf{F}^{\mu}_{\mathbf{i-1,i}} \tag{3.2}$$

In the limit of highly rigid cross-links and flexible filaments, our model approaches the pure semi-flexible filament models of [37, 99]. In the opposite limit (nearly rigid filaments and highly flexible cross links), our model approaches that of [10] in small strain regimes before any nonlinear cross link stiffening.

3.1.3 Drag-like coupling between overlapping filaments

Previous models represent cross-linkers as elastic connections between pairs of points on neighboring filaments that appear and disappear with either fixed or force-dependent probabilities [47, 10]. Here, we introduce a coarse-grained representation of crosslink dynamics by introducing an effective drag force that couples every pair of overlapping filaments, and which represents a molecular friction arising from the time-averaged contributions of many individual transient crosslinks (Fig. 3.1B). This coarse-grained approximation has been shown to be adequate in the case of ionic cross-linking of actin[95, 14], and has been used to justify simple force-velocity curves for myosin bound filaments in other contexts [2].

To implement coupling through effective drag, for any pair of overlapping filaments

j and k, we write the drag force on filament j as:

$$\mathbf{F}^{\xi}_{\mathbf{j,k}} = -\xi(\mathbf{v_j} - \mathbf{v_k}) \tag{3.3}$$

where ξ is the drag coefficient and $\mathbf{v_j}$, $\mathbf{v_k}$ are the average velocities of filaments j and k. We apportion this drag force to the two endpoints (j, j+1) of filament j as follows: If $\mathbf{x_{j,k}}$ is the position of the filament overlap, then we assign $(1 - \lambda_{\mathbf{j,k}})\mathbf{F}^{\xi}_{\mathbf{j,k}}$ to endpoint j and $\lambda_{\mathbf{j,k}}\mathbf{F}^{\xi}_{\mathbf{j,k}}$ to endpoint j+1, where $\lambda_{\mathbf{j,k}} = |\mathbf{x_{j,k}} - \mathbf{x_j}|/|\mathbf{x_{j+1}} - \mathbf{x_j}|$.

The total crosslink coupling force on endpoint i due to overlaps along filament i and i-1 can then be written:

$$\mathbf{F^{xl}_i} = \sum_j (1 - \lambda_{\mathbf{i,j}})\mathbf{F}^{\xi}_{\mathbf{i,j}} + \sum_k \lambda_{\mathbf{i-1,k}}\mathbf{F}^{\xi}_{\mathbf{i-1,k}} \tag{3.4}$$

where the sums are taken over all filaments j and k that overlap with filaments i and i-1 respectively.

This model assumes a linear relation between the drag force and the velocity difference between attached filaments. Although non-linearities can arise through force dependent detachment kinetics and/or non-linear force extension of cross-links, we assume here that these non-linear effects are of second or higher order.

3.1.4 Active coupling for motor driven filament interactions

To add motor activity at the point of overlap between two filaments j and k ; for each filament in the pair, we impose an additional force of magnitude υ, directed towards its (-) end (Fig. 3.1C):

$$\mathbf{F}^{\upsilon}_{\mathbf{i}} = -\upsilon\hat{\mathbf{u}}_{\mathbf{i}} \tag{3.5}$$

and we impose an equal and opposite force on its overlapping partner. We distribute these forces to filament endpoints as described above for crosslink coupling forces. Thus, the total force on endpoint i due to motor activity can be written as:

$$\mathbf{F_i^{motor}} = \upsilon \sum_j (1 - \lambda_{\mathbf{i,j}}) \left(\hat{\mathbf{u}}_\mathbf{i} - \hat{\mathbf{u}}_\mathbf{j}\right) q_{i,j} + \upsilon \sum_k (\lambda_{\mathbf{i-1,k}}) \left(\hat{\mathbf{u}_{\mathbf{i-1}}} - \hat{\mathbf{u}}_\mathbf{k}\right) q_{i-1,k} \tag{3.6}$$

where j and k enumerate over all filaments that overlap with filaments i and i-1 respectively, and $q_{j,k}$ equals 0 or 1 depending on whether there is an "active" motor at this location. To model dispersion of motor activity, we set $q_{i,j} = 1$ on a randomly selected subset of filament overlaps, such that $\bar{q} = \phi$, where $\bar{q}$ indicates the mean of q (Fig. 3.1C).

3.1.5 Equations of motion

To write the full equation of motion for a network of actively and passively coupled elastic filaments, we assume the low Reynold's number limit in which inertial forces can be neglected, and we equate the sum of all forces acting on each filament endpoint to zero to obtain:

$$0 = -l_s \zeta \mathbf{v_i} - \mathbf{F_i^{xl}} + \mathbf{F_i^{elas}} + \mathbf{F_i^{motor}} \tag{3.7}$$

where the first term represents the hydrodynamic drag on the half-filament adjoining endpoint i with respect to motion against the surrounding fluid, and ζ is the drag coefficient.

3.1.6 2D network formation

We used a mikado model approach [91] to initialize a minimal network of overlapping unstressed linear filaments in a rectangular 2D domain. We generate individual filaments by laying down straight lines, of length L, with random position and orientation. We define the density using the average distance between cross-links along a filament, l_c. A simple geometrical argument can then be used to derive the number of filaments filling a domain as a function of L and l_c [37]. Here, we use the approximation that the number of filaments needed to tile a rectangular domain of size $D_x \times D_y$

is $2D_xD_y/Ll_c$, and that the length density is therefore simply, $2/l_c$.

3.1.7 Modeling filament turnover

In living cells, actin filament assembly is governed by multiple factors that control filament nucleation, branching and elongation. Likewise filament disassembly is governed by multiple factors that promote filament severing and monomer dissociation at filament ends. Here, we implement a very simple model for filament turnover in which entire filaments appear with a fixed rate per unit area, k_{app} and disappear at a rate $k_{diss}\rho$, where ρ is a filament density (Fig. 3.1D). With this assumption, in the absence of network deformation, the density of filaments will equilibrate to a steady state density, k_{app}/k_{diss}, with time constant $\tau_r = 1/k_{diss}$. In deforming networks, the density will be set by a competition between strain thinning ($\gamma > 0$) or thickening ($\gamma < 0$), and density equilibration via turnover. To implement this model, at fixed time intervals $\tau_s < 0.01 \cdot \tau_r$ (i.e. 1% of the equilibration time), we selected a fraction, τ_s/τ_r, of existing filaments (i.e. less than 1% of the total filaments) for degradation. We then generated a fixed number of new unstrained filaments $k_{app}\tau_s D_x D_y$ at random positions and orientations within the original domain. We refer to $k_{diss} = 1/\tau_r$ as the turnover rate, and to τ_r as the turnover time.

3.1.8 Simulation methods

Further details regarding our simulation approach and references to our code can be found in the Supplementary Information (S1 Appendix A.1). Briefly, equations 1-7 define a coupled system of ordinary differential equations that can be written in the form:

$$\mathbf{A} \cdot \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{3.8}$$

where $\mathbf{x}$ is a vector of filament endpoint positions, $\dot{\mathbf{x}}$ the endpoint velocities, $\mathbf{A}$ is a matrix with constant coefficients that represent crosslink coupling forces between overlapping filaments, and $\mathbf{f}(\mathbf{x})$ represents the active (motor) and elastic forces on fil-

ament endpoints. We smoothed all filament interactions, force fields, and constraints linearly over small regions such that the equations contained no sharp discontinuities. We numerically integrate this system of equations to find the time evolution of the positions of all filament endpoints. We generate a network of filaments with random positions and orientations as described above within a domain of size D_x by D_y. For all simulations, we imposed periodic boundaries in the y-dimension. To impose an extensional stress, we constrained all filament endpoints within a fixed distance $0.05 \cdot D_x$ from the left edge of the domain to be non-moving, then we imposed a rightwards force on all endpoints within a distance $0.05 \cdot D_x$ from the right edge of the domain. To simulate free contraction, we removed all constraints at domain boundaries; to assess buildup and maintenance of contractile stress under isometric conditions, we used periodic boundary conditions in both x and y dimensions.

We measured the local velocity of the network at different positions along the axis of deformation as the mean velocity of all filaments intersecting that position; we measured the internal network stress at each axial position by summing the axial component of the tensions on all filaments intersecting that position, and dividing by network height; finally, we measured network strain rate as the average of all filament velocities divided by their positions.

We assigned biological plausible reference values for all parameters (See Table 3.1). For individual analyses, we sampled the ranges of parameter values around these reference values shown in S1 Table.

Table 3.1: Simulation parameters with reference values

Parameter	**Symbol**	**Reference Value**
extensional modulus	μ_e	$1nN$
compressional modulus	μ_c	$0.01nN$
cross-link drag coefficient	ξ	$unknown$
solvent drag coefficient	ζ	$0.0005\frac{nNs}{\mu m^2}$
filament length	L	$5\mu m$
cross-link spacing	l_c	$0.5\mu m$
active filament force	υ	$0.1nN$
active cross-link fraction	ϕ	$0.1 < 0.9$
domain size	$D_x \times D_y$	$20 \times 50\mu m$

3.2 Practical Implementation of Model Framework

Although the prior description is useful for communicating the mathematical principles underlying my modeling framework, I also want to provide a practical explanation of the logic of the code structure. This will help to clarify the implementation details that may be ambiguous in the above formulation and make it easier for others to approach and modify my code.

The simulation code is entirely built on top of MATLAB's ode solving functions. Therefore, at the highest level, the entire simulation framework simply boils down to providing functions to define the ode's left and right hand side, along with the initial conditions for the system. However, because MATLABs solvers do not allow discontinuities, discontinuous filament recycling events cannot be implemented directly in the ODE. Effectively, it is necessary to stop the simulation solver at fixed times, reset some subset of filaments, and restart the solver with the new state every time a turnover event happens.

In the following sections, I'll walk through the code's logic in more detail, emphasizing the packages and files that modularly summarize the core functions. First I will begin with an explanation of the command line interface used to set model parameters and deploying simulations.

To find the *activnet* simulation code, visit:

`https://github.com/wmcfadden/activnet/tree/master/simulation`

3.2.1 Launching simulations from the command line

The activnet package includes a function called activnet_gen.m, which can be used to launch simulations on any MATLAB system. The following is the documentation of the parameters passed to this function.

```
function p = activnet_gen(zet,L,mu,kap,lc,xi,ups,phi,psi,
                          r,sig,Dx,Dy,Df,Dw,ls,lf,tinc,tfin,nonlin)
% generates an active network simulation and prints node positions
% at time steps.  Parameters are defined as follows:
```

```
%
%   zet - medium viscosity
%   L - length of the filament
%   mu - compressional modulus of the filament
%   kap - bending modulus of a filament if ls<L
%   lc - average distance between filament overlaps
%   xi - frictional resistance between two overlapping segments
%   ups - motor force at filament overlaps
%   phi - fraction of overlaps that receive a motor force
%   psi - spatial variation in motor force
%         (if external force applied, psi sets the periodicity of
%          force application, and psi<0 sets square wave.)
%   r - recycling rate
%   sig - applied stress in the x direction applied at Df*Dx
%         (sig<0 sets stress to be applied in y direction)
%   Dx - x-dimension of domain
%   Dy - y-dimension of domain
%   Df - position at which external force is applied
%        (if Df<0, also position in x dimension where network stops)
%   Dw - width of window in x-dimension where forces are applied
%   ls - length of filament segments
%   lf - length of force falloff at end of filament
%         (just ensures continuous forces)
%   tinc - time increment to return solutions
%          (will be decreassed automatically if r is too large)
%   tfin - end time of simulation
%   nonlin - factor by which to make filament stiffer for extension
%            (if nonlin<0, add spacing so filaments dont reach edge)
```

These parameters will be explained in more detail below, but here I'll provide a brief practical explanation for their use in setting up simulations. The parameters

`zet`, `L`, `mu`, `kap`, `lc`, `xi`, `ups`, `phi`, `r`, `sig`, `Dx`, and `Dy` are defined in the mathematical methods section above as ζ, L, μ_c, κ, l_c, ξ, υ, ϕ, $1/\tau_r$, σ, D_x, and D_y, respectively. The parameter `nonlin` is used to internally calculate the extensional modulus $\mu_e = \texttt{nonlin} \times \mu_c$ (i.e. if `mu` is 1 and `nonlin` is 100, then $\mu_c = 1$ and $\mu_e = 100$). The parameter `psi` is used to generate a spatial gradient in internal activity or temporal periodicity if an external force is applied. The spatial gradient is linear with a maximum at the center of the domain. The parameter `Df` defines the fraction along the x-domain where any external stress will be applied (i.e. stress is applied at `Df`*`Dx`).

All of these terms are positive by definition so setting certain terms negative is used to trigger certain behavior in the simulation environment:

- Setting `mu` < 0 enables the extensional spring constant to be different than the compressive.
- Setting `sig` < 0 causes the external stress to be applied in the y direction rather than the x direction, resulting in shear stress simulations.
- Setting `nonlin` < 0 results in space being added to the edge of the domain so that no filaments intersect with the boundaries of the simulation, allowing the network to undergo unconstrained contraction.
- Setting `psi` < 0 results in oscillating positive and negative stresses being applied to the network (as opposed to a sinusoidal pattern regularly).
- Setting `Df` < 0 will cause the rightward edge of the initially generated network to end at the location where the force is applied.

The parameter `ls` sets the segment size of the filament. Theoretically this segment size is very small (the size of a single actin monomer perhaps), but for computational reasons this value is set much larger. The parameters `Dw` and `lf` set the regions over which forces will fall off for the domain and individual filaments, respectively. These are just there to ensure there are no discontinuous changes in the ODE equation as filaments move around, and should be set to some smallish number, like 0.05, but don't affect the results too much overall. Finally, `tfin` and `tinc` set the end time of

the simulation and the timesteps at which to print results, respectively. The program will automatically shrink `tinc` if the recycling rate, `r` is large enough that manual position updates must occur on timescales smaller than `tinc`.

To run a simulation with an external stress the following code could be used.

```
activnet_gen(0.001, 1, -0.01,  0,  0.8, 10, 0, 0, 0,
             0, -0.002, 20, 4, -0.33, 0.05, 1, 0.025, 1, 10000, 100)
```

To run a simulation with internal filament activity the following code could be used.

```
activnet_gen(0.001, 5, -0.01,  0,  0.3, 10, 0.1, 0.25, 0,
             0.001, 0, 25, 25, 0.5, 0.05, 5, 0.025, 1, 10000, -100)
```

Both of these have nonlinear filament extension because the third argument (`mu`) is less than 0. The results will be printed to the MATLAB console.

The currently deployed package can also be used to launch simulations on any linux system with MATLAB 2014b or the MATLAB 2014b Compiler Runtime installed (you will need to have a $MATLAB environment variable enabled. Please contact your sysadmin.). In addition, the package can be recompiled on any system running MATLAB to create a new deployment that will run on the compiled MATLAB version and operating system (see the README.txt file for more information about deployment).

```
./run_activnet_gen.sh $MATLAB 0.001 1 -0.01  0  0.8 10 0 0 0 0
         -0.002 20 4 -0.33 0.05 1 0.025 1 10000 100 > output.file
```

3.2.2 Package structure of activnet

To understand the high-level overview of this code, you can analyze the basic package structure and the small number of important top level packages.

- activnet_gen.m: topmost wrapping function to call for running simulations; confirms input, generates initial conditions, and calls activnet.m

- activnet.m: Chooses between which ODE implementation to run; stops ODE solver execution to perform filament recycling.
- odes folder: Contains all the ODE computation internals.
- helpers folder: Some additional functions written to perform line intersection tests.
- analysis folder: All the code used to generate the figures in the accompanying publication.

Note: the rest of this documentation is most useful if one is looking over the internals of the code. Trying to understand this without working with the code is probably not very useful.

3.2.3 ODE Wrapper functions

As mentioned above, the initialization and launch of the simulation takes place using the function activnet_gen.m. This function is quite simple. It just ensures proper input parameters, builds a randomized network of filaments, prints the network node positions, and then calls the function activnet.m to compute the simulation results. From there, activnet.m takes care of pausing solvers to update filament position for recycling as well as choosing between solvers depending on the type of problem being solved.

Generating Initial Conditions

Before an ODE solve can be called one needs to set up the initial conditions of the network. Every solver implementation in MATLAB operates on 1D vectors where every element represents a variable to be integrated. Therefore, the data structure for this model is going to need to be set up to be encoded as a 1D vector of numbers. Because the model framework is based around the 2D position of points, I will continually be shifting between the 1D vector, `z` and the 2D vector `p` using the following transformations.

```
p = reshape(z0,[],2);  % z -> p
z0 = reshape(p,1,[]);  % z -> p
```

To select the number of filaments, `N`, to generate using the input parameters, we use the approximating formula `N` $\approx$ `floor(2*Dx*Dy/lc/L)`. This formula is derived from the tiling of a $Dx \times Dy$ domain with lines of length L and spacings between lines lc. This diverges from the exact number of filaments needed when L/lc is small, but we ignore this discrepancy because we mostly operate in the regime where $L/lc > 10$.

We therefore begin by creating an initial set of filament segment endpoints `p`, where we want to have `N` total filaments each with `ncnt` segment endpoints. We do this by selecting a random starting point in our domain (`Dx` by `Dy`) with random orientation (`thet`).

```
p = zeros(N*ncnt,2);   %initialize all endpoints to 0
for i=1:N
p((i-1)*ncnt+1,:) = [Dp*Dx*rand Dy*rand]; %random starting location
thet = rand*2*pi;                         %random direction
for j = 2:ncnt            % iterate through remaining segments to add endpoints
p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(ncnt-1.0)*[cos(thet) sin(thet)];
end
end
```

In most cases, the network should be assembled such that the domain is filled entirely in the y dimension ($y = 0$ and $y = Dy$) and filled to the left edge ($x = 0$) and up to the position $x = Dp * Dx$ in the in the x-dimension. However, in the case that the parameter `nonlin` is less than 0, we set the initial conditions such that there is an empty space of size $0.2Dx$ and $0.2Dy$ at the x and y boundaries.

To keep our density the same we should adjust our calculation of N above, however, in the current implementation this adjustment is not made. This is a known bug that was corrected for after-the-fact in the publication by multiplication of l_c by a constant factor depending on the simulation setup.

Choosing between active and driven simulations

During development I noticed that there were some cases where I could skip certain sections of the computation to speed up integration of the ODEs. In particular, when there is no internal forces generated by filament activity, the right hand side of the differential equation is easier to compute (see below). Therefore, in any situation where internal activity is set to 0, we use the more efficient code. The activnet.m code makes this decision by determining if there is pulling of filaments rather than internal force generation.

```
pull = (isempty(nu)&&sig~=0)
```

In this case it runs a different pair of functions to compute the ODE right hand side and mass matrix (see below).

Implementing Filament Recycling

The core function of activnet.m is to serve as a wrapper for the ODE solvers mentioned above and described in greater detail below. However, since the ODE solvers won't allow discontinuous changes to the positions of nodes, activnet.m must also carryout the task of stitching together smaller simulations and manually updating positions in between. The code in activnet.m implements a repeated loop with calls to the underlying ode solvers. If $r > 0$, a solver computes the integral between two adjacent time points in the vector of all time points to evaluate, tt. Then the positions are updated and the solver is used to evaluate integral for the next time interval. Logic implemented in activnet_gen.m ensures that the timesteps are selected small enough so that there will not be too many filaments disappearing at once.

The following code chunk implements the position update for a randomly selected subset of nodes whenever the recycling rate, r, is greater than 0. The logic implemented in activnet_gen.m ensures that the timesteps are not so large that more than 5% of filaments will be undergoing a discontinuous jump at any one time (i.e $r * istep * tinc < 0.05$).

```
% % select random indices
i = randi(N,floor(r*istep*tinc*N)+(rand<mod(r*istep*tinc*N,1)),1);
% % reset the position of the first segment endpoint
p((i-1)*ncnt+1,:) = [Dp*Dx*rand(size(i)) Dy*rand(size(i))];
% % pick random orientation for each filament
thet = rand(size(i))*2*pi;
for j = 2:ncnt  % % iterate over rest of endpoints
  p((i-1)*ncnt+j,:) =
  p((i-1)*ncnt+j-1,:)+L/(ncnt-1.0)*[cos(thet) sin(thet)];
end
```

3.2.4 Overview of Numerical Integration Implementations

As mentioned above, the solution is numerically integrated using MATLAB's built-in ODE solvers. The ODE equation is the low-Reynolds limit of Newton's equation of motion for all the filaments. So to implement, I simply supply an ode solver with a function to compute the right hand side of a system of differential equations ($\mathbf{f}(\mathbf{x})$) along with a function to compute the mass matrix ($\mathbf{A}$) that connects the derivatives on the left hand side of the ODE system.

$$\mathbf{A} \cdot \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{3.9}$$

With those two matrices the system of differential equations can then be integrated to the desired level of precision by one of MATLABs implicit ode solvers (e.g. ode15s or ode23). In that equation, $\mathbf{x}$ simply represents the positions of filament endpoints (nodes). Therefore $\dot{\mathbf{x}}$ is just a way to represent the velocity of every endpoint.

The right hand side represents the forces (external or internal) driving the motion of the filaments. The mass matrix $mathbf{A}$ represents the coupling of filaments to each other (off diagonal elements) and to the solvent in which they are embedded (diagonal elements). Once those are provided, MATLAB's solvers take care of the integration (see MATLAB ode solver documentation for more info).

Right hand sides: *activnet_act_ode.m vs. activenet_pull_ode.m*

The right hand side of the equation constitutes all the non-viscous forces in the simulation. The most important and fundamental forces are those of the intrafilament spring mechanics. In these simulations, filaments are represented by chains of filament segments that act as springs with particular properties. As described above, each segment has a compressive (`mu`) and extensional (`mu` $\times$ `nonlin`) spring constant that governs the motion between endpoints along the length of each filament. There is also a bending spring constant (`kap`) which effectively tries to straighten adjoining segments if they are not already colinear. The internal mechanical forces of all filaments in the simulation is represented in the following code snippet.

```
%% compute intrafilament forces
l0 = L/(ncnt-1);           % length of segment
dp = zeros(size(p));
for n=1:ncnt:length(p)
  va_orth=[0 0];
  va = [0 0];
  la = 0;
  for i=0:ncnt-2
  % % custom subtraction minding boundaries
    vb = mydiff(p(n+i,:),p(n+i+1,:),Dx,Dy);
    lb = sqrt(vb*vb');
    gam = (lb-l0)/l0;      % extension of the segment
    f = mu*vb/lb*gam;      % longitudinal spring force
    if(mu<0)               % allow nonlinearity
      f = -f*(1+(muN-1)*double(gam>0));
    end
    dp(n+i,:) = dp(n+i,:) + f;
    dp(n+i+1,:) = dp(n+i+1,:) - f;

    % % below is for bending stiffness
```

```
    vb_orth = [-vb(2) vb(1)];
    if(i>0)
      if(va_orth*vb'>0)
        va_orth = -va_orth;
      end
      if(vb_orth*va'<0)
        vb_orth = -vb_orth;
      end
      tor = kap/l0^2*acos(max(min(va*vb'/la/lb,1),0));
      dp(n+i-1,:)=dp(n+i-1,:)+tor*va_orth/la;
      dp(n+i,:)=dp(n+i,:)-tor*va_orth/la;
      dp(n+i+1,:)=dp(n+i+1,:)+tor*vb_orth/lb;
      dp(n+i,:)=dp(n+i,:)-tor*vb_orth/lb;
    end
    va = vb;
    va_orth = vb_orth;
    la = lb;
  end
end
```

The code loops over every `ncnt` nodes, which constitute the nodes of a single filament. Next the code loops through each node of the individual filament. First, the code evaluates the extensional force on each node given by `f = mu*vb/lb*gam` with the modification that if `mu`¡0 and `gam`¿0, the force constant is modified to incorporate the nonlinear extensional stiffness. Next, the code evaluates the bending forces for adjacent segments that are not aligned using

`tor = kap/l0^2*acos(max(min(va*vb'/la/lb,1),0))`.

The `max(min(...))` is used simply to ensure that there are no aberrantly large values due to errors in evaluation of `acos` with arguments very slightly greater than 1 or less than 0. The two vectors `va_orth` and `vb_orth` are calculated to direct the force

orthogonal to the orientation of the filament segment. Finally, it should be noted that a bending force is generated from both the segment behind and te segment in front of the current node of interest and that the force from these need to be assigned at both the current node and the equal and opposite force assigned at the other end of each filament.

After the basic mechanical properties of the filament are accounted for, we need to add extra forces (otherwise the simulations won't be any more interesting than just a bunch of inert springs). What we do depends on whether the simulation is meant to represent active motors or a passive network with an external driver. In the next two sections we cover the specifics of the force equations for both of these cases.

Active ODE

For the ODE with internal activity, we need to compute a very different force on each node. In particular, we have to check for intersections and add forces to filaments that intersect as if they are being pulled on by an active motor at their overlap point. To do this we will use one of the helper functions described below to return all of our overlapping lines. After that we will step through all pairs of intersecting lines and add the appropriate force in the appropriate direction. The following code snippet shows the process with documentation.

```
g = lineSegmentGrid(indL,XY,Dx,Dy,l0);  % find instersecting lines

% find distance of intersections from endpoints
f = min(1,max(0,(g-lf/2)/(1-lf)));
for ind=1:size(g,1)
  i = g(ind,3);
  j = g(ind,4);

  vm = mydiff(p(j,:),p(j+1,:),Dx,Dy);
  lm = sqrt(vm*vm');
```

```
  edg = 1;   % this will be used to reduce the force
 % as it gets closer to the edge of a filament

  if(g(ind,1)<lf)
    edg = edg*g(ind,1)/lf;
  elseif((1-g(ind,1))<lf)
    edg = edg*(1-g(ind,1))/lf;
  end

  if(g(ind,2)<lf)
    edg = edg*g(ind,2)/lf;
  elseif((1-g(ind,2))<lf)
    edg = edg*(1-g(ind,2))/lf;
  end

  mul = 1;   % this will be used to modulate the force if
             % psi says there should be a spatial gradient

  if(psi>0)
    mul = double(g(ind,5)>=psi*abs(-Dx*Df));
  end

  tnu = nu(ceil(i/ncnt),ceil(j/ncnt))*mul;  % variation in
% filament force (phi)

  dp(i:i+1,:) = dp(i:i+1,:) + edg*tnu/lm*[vm*(1-f(ind,1));vm*f(ind,1)];
  dp(j:j+1,:) = dp(j:j+1,:) - edg*tnu/lm*[vm*(1-f(ind,2));vm*f(ind,2)];

end
```

The last line shows the net force that is added to both pairs of nodes that are

intersecting (`i:i+1` and `j:j+1`) with one positive and the other negative to conserve the net force to be 0. The term `f` is calculates the distance of the overlap point between the two nearest node locations, and is used to set how much force is applied to the first vs second node. For each iteration of the loop, the force is calculated for the `j:j+1` filament, and each interaction is stepped through twice (once with each filament in the `j:j+1` role). The vm calculation determines the direction in which the forces will be applied. The terms `edg` and `tnu` simply modify the applied force by scalar quantities based on closeness to the end of the filament and spatial variation in motor force intensity.

Pulling ODE

For the case of the system with an external stress, we will need to add forces at the desired location of stress, and we will need to constrain the filaments located at the edge of the domain. The following two code snippets show these two behaviors.

```
%% add external force at centerline
if(psi>0)
val = sig*sin(psi*t);
elseif(psi<0)
val = sig*round(mod(0.5+-psi*t,1)).*(round(mod(0.55+-psi*t/2,1))-0.5)*2;
else
val = sig;
end
```

```
subp = p(:,1)>(Df-Dw)*Dx&p(:,1)<(Df+Dw)*Dx;
ff = 1-abs(p(subp,1)-Df*Dx)/Dw/Dx;
if(sig<0)
  dp(subp,1)=dp(subp,1) - Dy*val.*ff/sum(ff);
else
  dp(subp,2)=dp(subp,2) - Dy*val.*ff/sum(ff);
end
```

The top section of this code calculates stress that should be applied and sets that as `val`. If `psi` is nonzero then we are looking to have a temporally varying applied stress. If `psi` > 0, then we want a sinusoidally varying stress with frequency psi. If `psi` < 0, then we want a square wave with frequency psi. The max stress in each case is still `sig`.

The bottom section selects the nodes that will have force added to them in `subp`. The force to be applied to each node is not equally distributed, but instead, `ff` sets the fraction of force to be proportional to the distance to the center of the region of applied stress (set by the width `Dw`). This distribution function is normalized so that the total amount of force is equal to the amount of force needed to set the stress to `val` (i.e. $F_{total} =$ `val` $\times$ `Dy`). Finally the force is applied in the x direction or the y direction based on whether `sig` is greater or less than 0.

```
% % constrain edges
subp = p(:,1)<Dw*Dx;
dp(subp,:)=dp(subp,:).*repmat(4*p(subp,1)/Dw/Dx-3,1,2);

subp = p(:,1)>Dx*(1-Dw);
dp(subp,:)=dp(subp,:).*repmat(4*abs(p(subp,1)-Dx)/Dw/Dx-3,1,2);

subp = p(:,1)<3*Dx/4*Dw|p(:,1)>Dx-3*Dx/4*Dw;
dp(subp,:)=0;
```

This last segment merely constrains the force at the far left and right edges of the domain to be 0 (within 0.75`Dw` $\times$ `Dx` from each edge). There is a linear transition to `dp=0` in the remaining 25% of the region.

It should be noted that in the pulling simulations, there is no need to compute the intersection of filaments. This means that there are far fewer computations than in the active case. I wrote two separate functions for each of the different cases simply to avoid having to perform redundant checks on the cases repeatedly (i.e. the choice is made once in the outer wrapping functions rather than having to repeatedly make the choice for which code to run on each iteration of the solver).

3.2.5 Left hand side: The mysterious mass matrix

To understand the logic of coupling filaments together by their velocities it might be helpful to look at a simple example. Imagine that you have a single particle in some one-dimensional fluid with viscosity, ζ, and with a force, F, acting on it. Assuming low Reynolds limit so there is no acceleration of the particle, the equation of motion would look like the following.

$$\zeta \dot{x} = F \tag{3.10}$$

Now assume you have two particles in that fluid, but (for now) we assume the particles can't interact in any way. Therefore, we can express the equation of motion for both particles pretty simply with the following equation.

$$\zeta \dot{\mathbf{x}} = \mathbf{F} \tag{3.11}$$

Obviously, all we did was replace the scalars with vectors. To make things a little neater we could replace the scalar ζ with a so called mass matrix that would simply look like the following matrix.

$$\mathbf{A} = \begin{bmatrix} \zeta & 0 \\ 0 & \zeta \end{bmatrix} \tag{3.12}$$

Now, if we had some drag-like coupling between the velocities of our two particles with a drag coefficient, ξ, we could simply add a term to the off diagonal.

$$\mathbf{A} = \begin{bmatrix} \zeta & -\xi \\ -\xi & \zeta \end{bmatrix} \tag{3.13}$$

And carrying out our math we can see that this just gives a frictional coupling between the two particles.

$$\begin{aligned} \zeta v_1 - \xi v_2 &= F_1 \\ \zeta v_2 - \xi v_1 &= F_2 \end{aligned} \tag{3.14}$$

This is the entire logic behind the construction of the activnet_mass.m function. After finding which filament segments are overlapping, I simply add off-diagonal terms to the mass matrix that couple the nodes of those filament segments together.

Similar to the previous section, there are computational efficiencies to be gained in computing the mass matrix, $\mathbf{A}$, if the system is being driven externally (as opposed to being driven by internal activity).

activnet_mass.m vs. activenet_mass_sp.m

When we are constraining filaments to remain motionless we run into one little problem with our mass matrix formulation as presented thus far. Essentially, the right hand side of the equation is being set to 0, but the left hand side still has cross terms . To remedy this, in the case where an external stress is applied, I need to manually decouple filaments that lie along the $x = 0$ boundary of the domain. This is carried out in activenet_mass_sp.m, but it is not implemented in activenet_mass.m because that code only runs in the case where there is internal activity of the filaments and no external constraints.

Additionally, in activenet_mass_sp.m, I utilize a sparse matrix for reasons that I honestly don't exactly remember. I assume that I found in that condition I could get some kind of marginal speedup by using a sparse matrix instead of the full matrix.

3.2.6 Line intersection helper Functions

There are a series of helper functions that are called to aid in the calculations of filament intersections. Yu can analyze the code directly for more detail but briefly, the lineSegmentIntersect.m implementation manually computes the intersection between all pairs of segments while lineSegmentGrid.m first bins lines into grids before testing intersection just on those in the bin. I saw a significant improvement when I moved to lineSegmentGrid even though asymptotically they both perform worst case $O(n^2)$, which I believe is due to the uniformity of the line segment distribution. Some mathematician might be able to prove that. It is interesting to note that the classical best-case line scanning algorithm (i.e. sweep line) is $O(nlogn)$.

3.2.7 Visualization code

In the analysis package, I have included some code called netplot_str.m to aid with visualizing the output of the simulations. This code opens the output file and an accompanying script file to get the parameters passed to the function. The code then goes on to render all the lines in a MATLAB plot. It could e modified easily to render the output in whatever format you may need. However, it is important to note that you need the input parameters to correctly associate output nodes to their correct filament.

CHAPTER 4
PHASES OF DEFORMATION IN FILAMENT NETWORKS WITH CROSS-LINK SLIP

4.1 Results

4.1.1 Steady-state Approximation of Effective Viscosity

We begin with a calculation of a strain rate estimate of the effective viscosity for a network described by our model in the limit of highly rigid filaments. We carry this out by assuming we have applied a constant stress along a transect of the network. With moderate stresses, we assume the network reaches a steady state affine creep. In this situation, we would find that the stress in the network exactly balances the sum of the drag-like forces from cross-link slip. So for any transect of length D, we have a force balance equation.

$$\sigma = \frac{1}{D} \sum_{filaments} \sum_{crosslinks} \xi \cdot (\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x})) \tag{4.1}$$

where $\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x})$ is the difference between the velocity of a filament, i, and the velocity of the filament, j, to which it is attached at the cross-link location, $\mathbf{x}$. We can convert the sum over cross-links to an integral over the length using the average density of cross-links, $1/l_c$ and invoking the assumption of (linear order) affine strain rate, $\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x}) = \dot{\gamma} x$. This results in

$$\sigma = \frac{1}{D} \sum_{filaments} \int_0^L \xi \cdot \left(\mathbf{v_i}(\mathbf{s}) - \mathbf{v_j}(\mathbf{s})\right) \frac{ds \cos\theta}{l_c}$$
$$= \sum_{filaments} \frac{\xi \dot{\gamma} L}{l_c} \cos\theta \cdot (x_l + \frac{L}{2} \cos\theta) \tag{4.2}$$

Here we have introduced the variables x_l, and θ to describe the leftmost endpoint and the angular orientation of a given filament respectively. Next, to perform the sum over all filaments we convert this to an integral over all orientations and endpoints that intersect our line of stress. We assume for simplicity that filament stretch and filament alignment are negligible in this low strain approximation. Therefore, the max distance for the leftmost endpoint is the length of a filament, L, and the maximum angle as a function of endpoint is $\arccos(x_l/L)$. The linear density of endpoints is the constant D/l_cL so our integrals can be rewritten as this density over x_l and θ between our maximum and minimum allowed bounds.

$$\sigma = \frac{1}{D}\int_0^L dx_l \int_{-\arccos(\frac{x_l}{L})}^{\arccos(\frac{x_l}{L})} \frac{d\theta}{\pi}\frac{\xi\dot{\gamma}L}{l_c}\cdot\frac{D}{Ll_c}\cdot(x_l\cos\theta + \frac{L}{2}cos^2\theta) \tag{4.3}$$

Carrying out the integrals and correcting for dangling filament ends leaves us with a relation between stress and strain rate.

$$\sigma = \frac{(L-2l_c)^2\xi}{4\pi l_c^2}\dot{\gamma} \tag{4.4}$$

We recognize the constant of proportionality between stress and strain rate as a viscosity. Therefore, our approximation for the effective viscosity, η_{eff}, at steady state creep in this low strain limit is

$$\eta_{eff} = \frac{(L-2l_c)^2\xi}{4\pi l_c^2}. \tag{4.5}$$

As illustrated in Figure 4.1, under moderate strains ($\gamma < 0.2$), our simulations show that in the high density limit, our theoretical approximation from Eqn 5.11 is highly accurate at explaining the network behavior. Aside from a geometrical factor, our approximation is valid for both shear and extensional stresses applied to the network.

As the density of the network approaches the breakdown limit, the effective viscosity diverges from our expected value. At the low connectivities, our expected viscosity goes to 0, but the medium viscosity begins to take over as we cross the percolation threshold at $L/l_c \sim 6$.

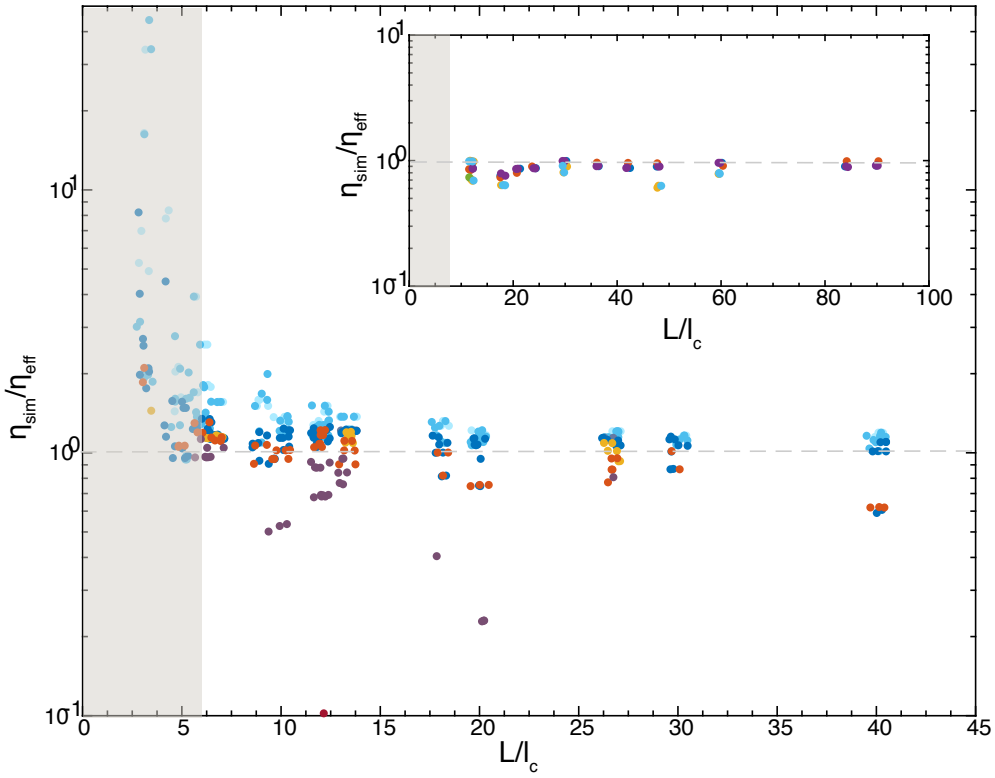


Figure 4.1: Ratio of effective viscosity measured by shear simulation to predicted effective viscosity as a function of connectivity, L/l_c. Inset: Same measurement for extensional simulations

In addition to changing the architecture and effective drag coefficient, we also validated the generality of our approximation by varying simulation size, medium viscosity, filament stiffness, and applied stress. We were able to find a slight trend that depended on filament stiffness as indicated in the difference between blue and red data points in Figure 4.1. The deviation from our approximation and variability in results manifested itself more strongly when filaments were highly compliant. To investigate this effect further, we next performed a more detailed analysis of the creep response while varying filament compliances.

4.1.2 Effects of Filament Compliance

The effect of filament compliance on cross-linked networks under strain is a subject of active research at the moment [Sayantan]. Therefore, we wished to use our computational approach to extend our understanding of filament networks in the regime of non-negligible filament compliance.

In irreversibly cross-linked polymer networks, filament compliance is known to give rise to elastic deformation of the network as described in[37, 99].

During the initial affine deformation immediately after the application of an external stress, we see a rapid stretch of filaments, $\langle \delta L/L \rangle_0$, in response to the affine purely mechanical strain, γ_{xy}, which closely follows $\langle \delta L/L \rangle_0 = \gamma_{xy} sin(\theta) cos(\theta)$. As shown in Figure 4.2, during the first phase in our simulations, the total network strain (solid) is described almost entirely by the strain of the filaments (dotted).

However, in the presence of cross-link slip, the filaments are not permanently constrained to remain at $\langle \delta L/L \rangle_0$. Interestingly, although the mean filament strain stays approximately constant, the distribution of individual filament strains broadens around the affine approximation as shown in the inset of Figure 4.2.

During the period where crosslink slip allows changes in the filament length distribution, we also find a long-lived intermediate relaxation phase that deviates from both the initial purely elastic relaxation and the later purely viscous behavior of section 5.3.2. In Panel B of Figure 4.2, we show that the standard deviation of the filament stretch distribution continues to increase throughout the period that the strain rate is non constant.

Approximating this broadening as a normally distributed variation in filament stretched length throughout the network ($\mathcal{N}$) with a time varying standard deviation, $\sigma(t)$, we have $\delta L/L = \langle \delta L/L \rangle_0 + \sigma(t) \cdot \mathcal{N}$. This has an effect on the total mechanical energy stored in the network $\mathcal{H} \sim \langle \delta L/L \rangle^2 = \langle \delta L/L \rangle_0^2 + \sigma(t)^2$. Therefore, the network will deform further while some strain energy is being stored in the further stretching filaments.

Eventually the contribution from slow filament stretching will become negligible compared to that from pure cross-link slip on rigid rods. This occurs on a timescale

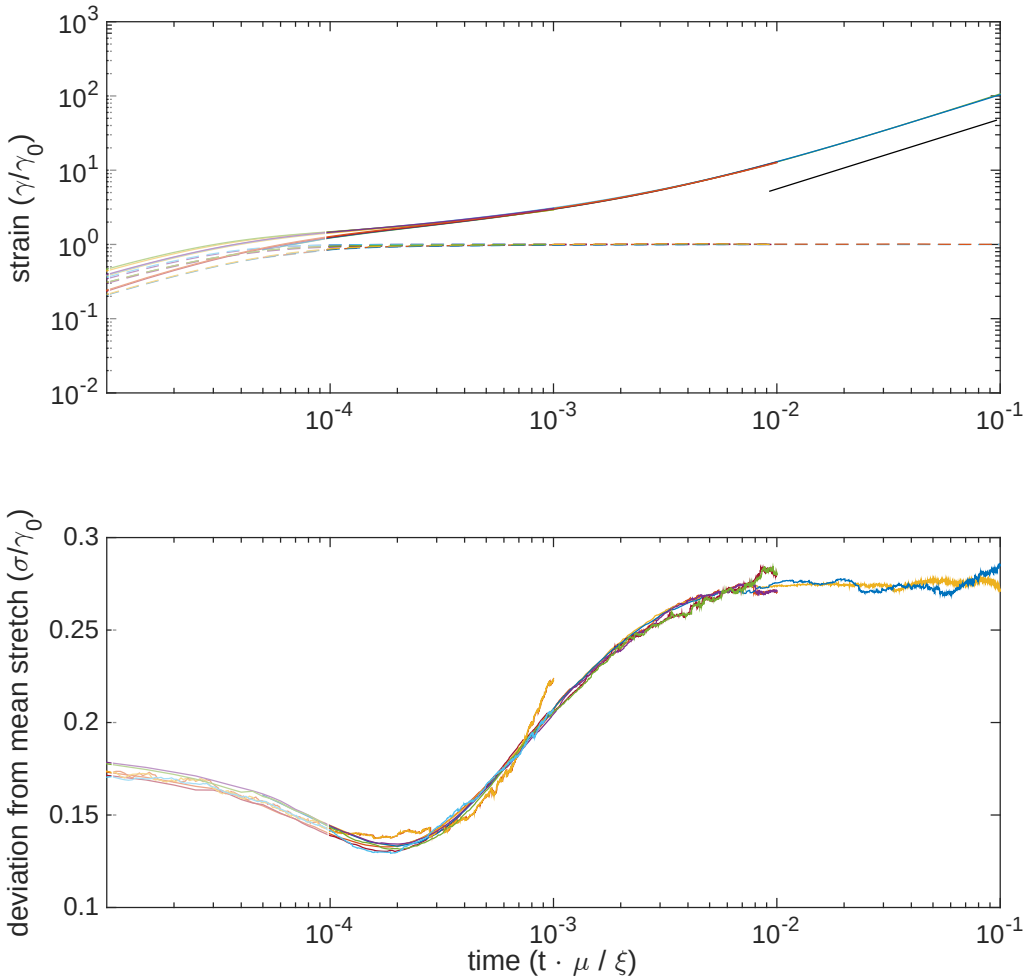


Figure 4.2: Network and filament strain for different filament drag coefficient parameters. (top) Plot of total strain normalized by the final mean filament strain, $\delta L/L$. Dashed lines show the amount of strain from affine mechanical stretching. (bottom) Standard deviation of filament extension for the networks in A. Note that the creep compliance in A becomes constant (slope 1) only after the spread in filament extension in B stops increasing. Colors indicate unique experimental conditions.

similar to that of cross-link slip and causes the effective viscosity to decay back toward the rigid limit. This gives rise to a less-than-linear creep response during times after the initial elastic relaxation but before full filament relaxation from cross-link slip. As shown in Figure 4.3, the transition begins to take place as network strain reaches

10 to 100 times the strain from pure mechanical stretching, $\gamma_0 = \delta L/L$, and this property is independent of the magnitude of the rate of strain.

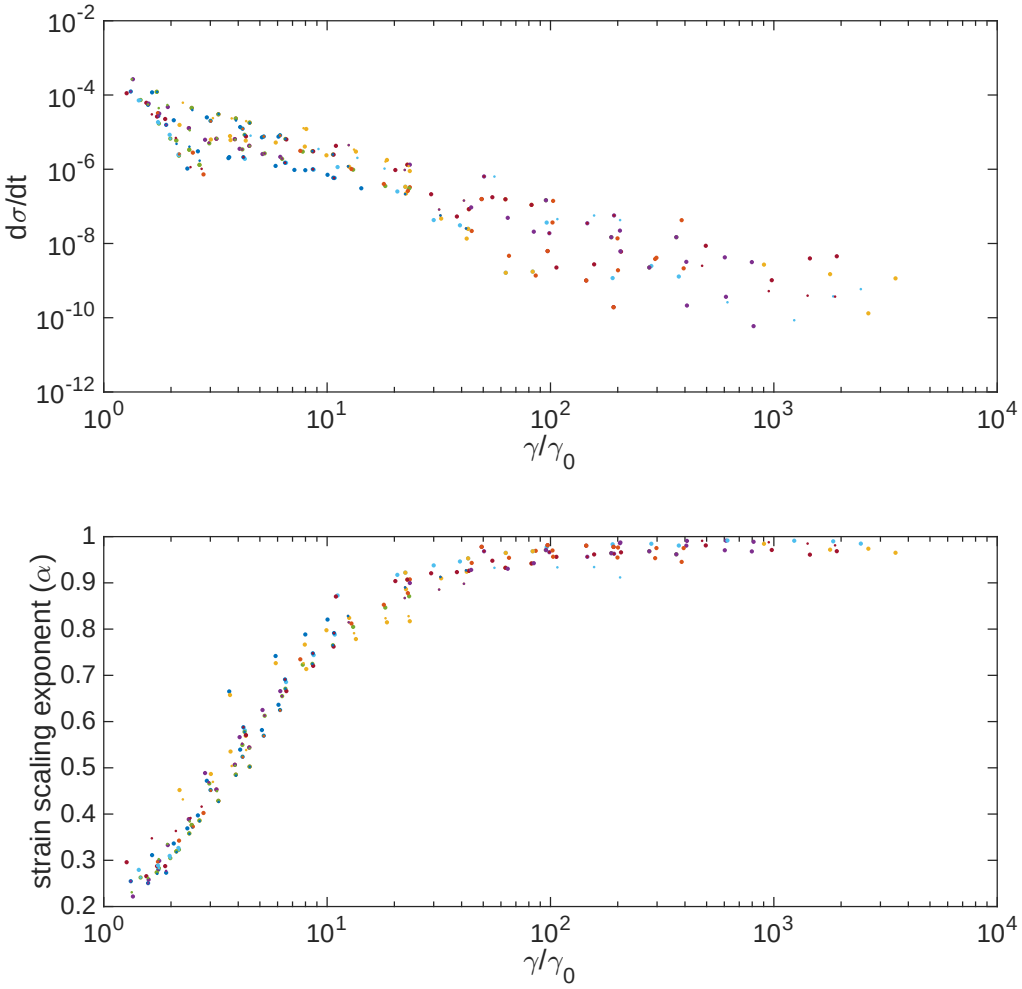


Figure 4.3: Sublinear network strain ends as change in filament strain decays. (top) Change in standard deviation of filament strain, σ, as a function of strain relative to pure mechanical strain. (bottom) Dependence of strain rate exponent as a function of strain relative to pure mechanical strain, γ_0. Colors indicate unique experimental conditions.

4.1.3 Alignment at High Strain and Network Tearing

Once the network is able to accumulate a large strain, the assumption of nearly uniform distributions of filament orientations begins to break down.

At this point the filament orientations become unevenly distributed $\langle \delta L/L \rangle \neq \gamma_{xy} sin(\theta) cos(\theta)$, with a larger number of filaments aligning in the direction of extension rather than compression. Filament alignment, conceptually, causes the formation of subdomains that no longer span the space of the network. To the authors' knowledge an exact derivation of the dependence of network connectedness on filament alignment has not been carried out, but Monte Carlo simulations have been used to show that alignment does indeed lead to lower connectedness[23].

We find that over time, the orientational distribution of the filaments begins to peak around 45 degrees as the large strain induces alignment. In Figure 4.4, we see that as the angular standard deviation falls, this reorientation eventually leads to fewer bonds bridging the network perpendicular to the line of strain. As this connectivity begins to noticeably decrease, the observed effective viscosity decreases as well, giving rise to greater than linear creep. From the inset of Figure 4.4 we can also see that the onset of phase D occurred before the network had completely reached phase C, leading to a rapid transition between sub-linear and super-linear creep. Finally, it should be noted that the end of this simulation resulted in the network tearing apart.

4.1.4 Phase Diagram of Dominant Behavior

In Figure 4.5, we illustrate the four stereotyped phases of the general mechanical behavior that we observed in our networks. A deforming network typically undergoes a rapid filament stretching, a slower relaxation of elastic constraints, a phase of purely viscous cross-link slippage, and an eventual alignment and breakdown of network connectivity.

Finally, to explore the transitions between the various phases, we measured the creep response for a computationally tractable network ($L/l_c = 25$), as we varied the filament extensional modulus, μ, and the cross-link friction coefficient, ξ. In Figure

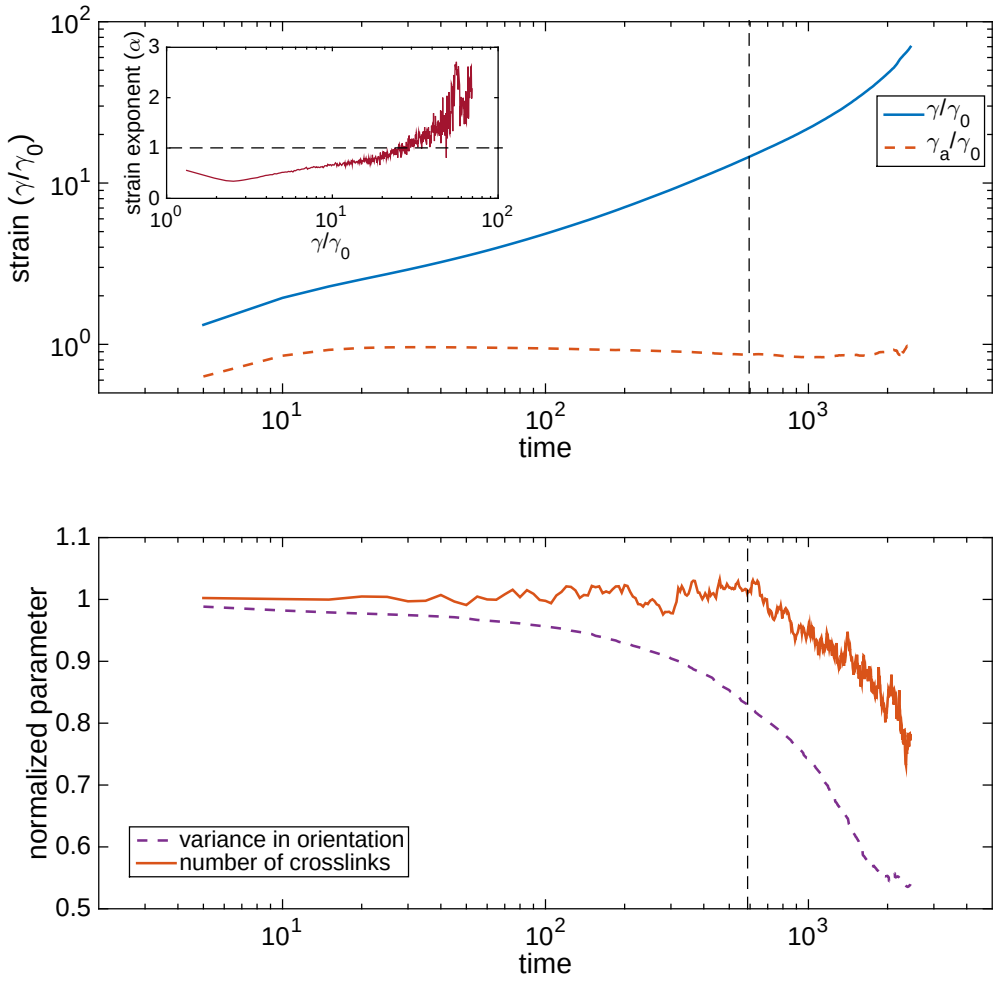


Figure 4.4: Creep response of a network transitioning to phase D. (top) Strain curves for a network undergoing large scale deformation. Inset shows strain exponent as a function of strain (exponent passes 1). (bottom) Traces for the variance in filament orientation and number of cross links. Vertical dashed line shows the point where the strain exponent becomes greater than one.

4.6, we classified parameter sets based on their strain exponent. We can see the trends for the transitions between phases A, B, and C. The line for the transition to D is still speculative at this time.

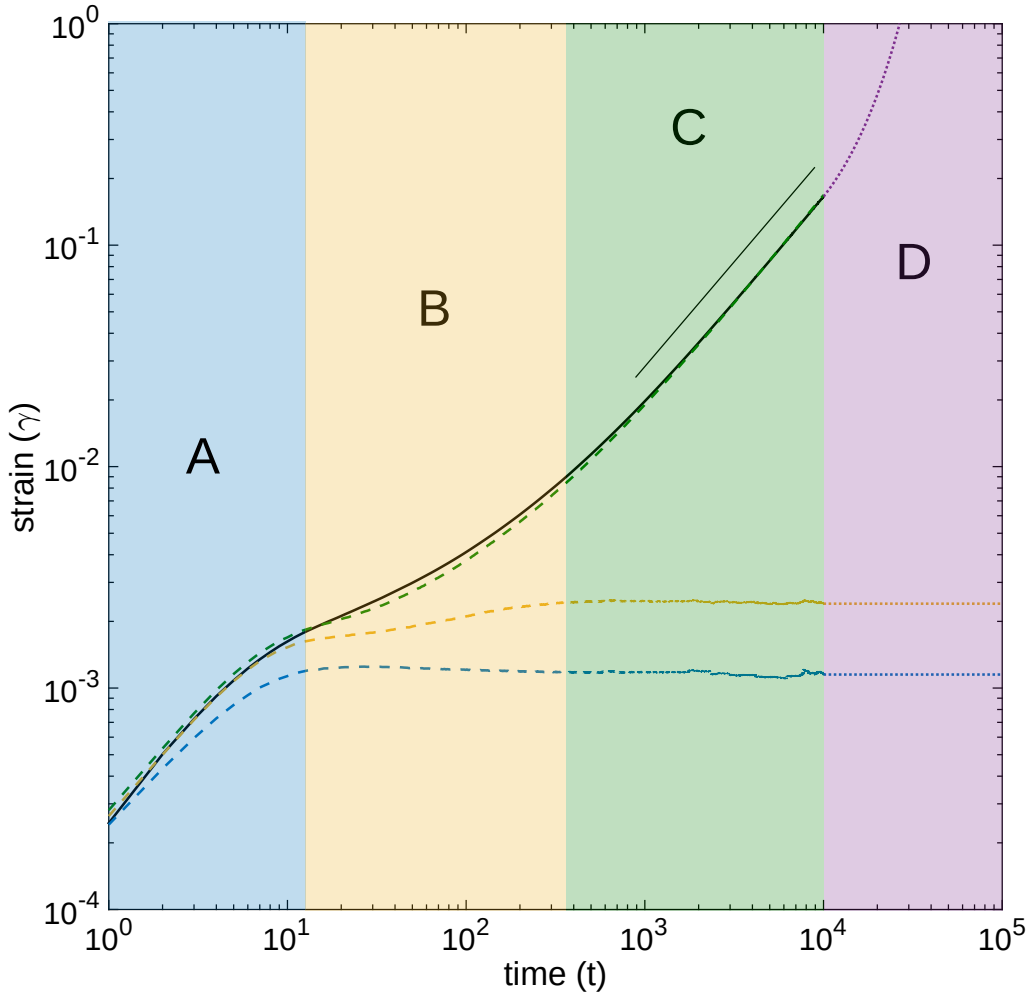


Figure 4.5: Schematic of the general creep response of compliant filament networks illustrating the 4 phases of deformation: A) rapid mechanical response, B) combination of slow filament stretching and cross-link slip, C) cross-link slip dominated (line indicates slope of one), D) network tearing from filament alignment. Note that the portion of the curve in section D is only a hypothetical continuation of the actual data.

4.1.5 Frequency dependent modulus

This section will include a figure on the frequency dependent modulus once I get those.

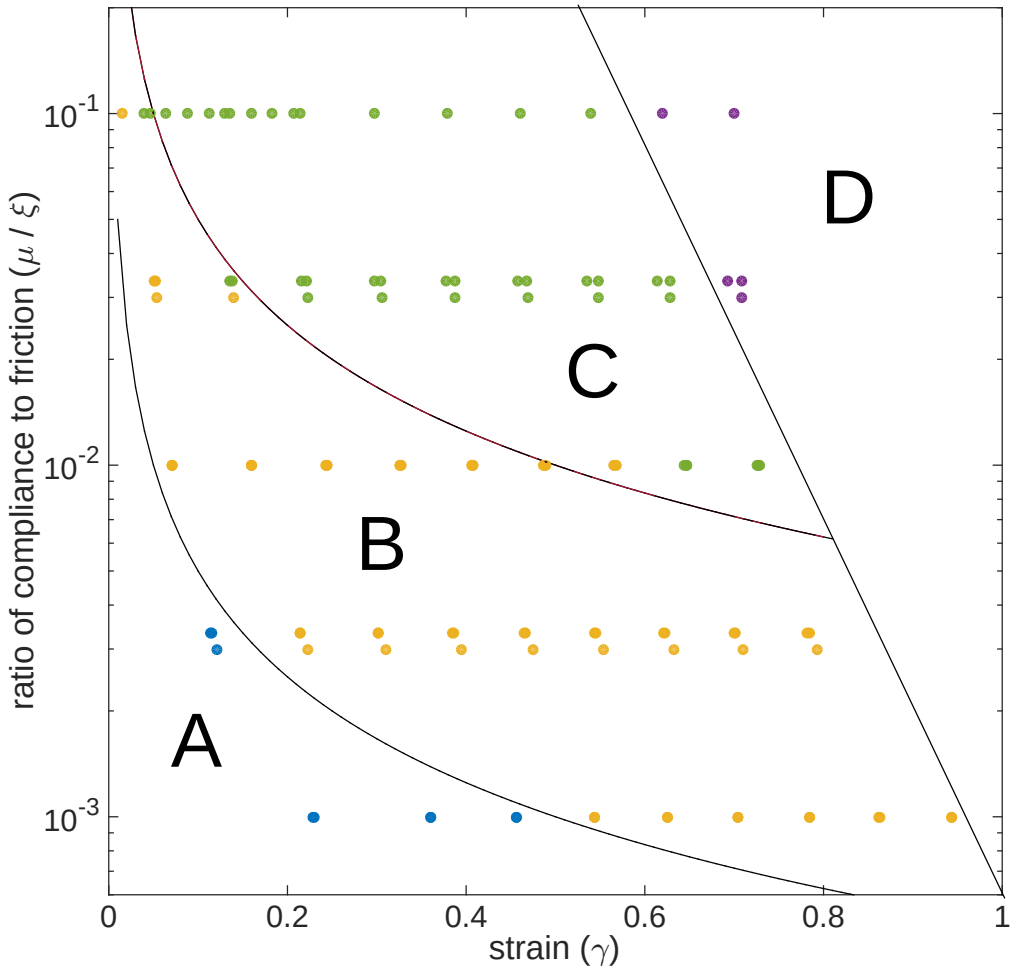


Figure 4.6: phase diagram of creep response for different filament extension, μ and cross-link friction, ξ. Yellow, green, and purple dots correspond to creep measurements $\gamma \sim t^{\alpha}$ with $\alpha < 0.92$, $0.92 < \alpha < 0.98$, or $\alpha > 0.98$ respectively. Blue dots represent creep measurements where $\gamma_{total} < 2\gamma_{mechanical}$

4.1.6 Strain Memory

Finally, we found an interesting behavior when we introduced non-linear extensional stiffness into our filaments. When the network is allowed to relax to it's unstrained state, there is generally a time comparable to the period of strain storage over which the energy in the network is relaxed away.

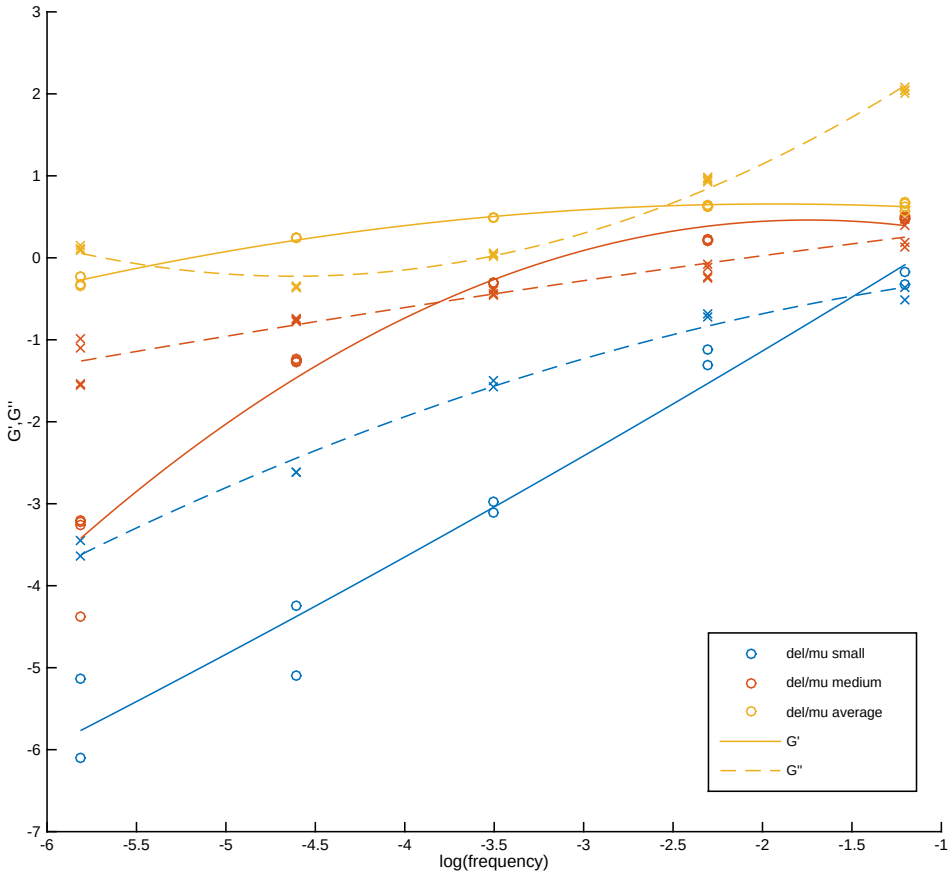


Figure 4.7: Frequency dependent moduli for networks.

We observed deviations from this behavior by applying stepwise stress pulses to simulated networks, and observing whether the network behaves identically upon reversal of the applied stress direction. If the network has no strain memory then each reversal will result in an identically shaped creep curve. However, when we include nonlinear filament extension in our model, we find that the mechanical strain can be stored for longer periods of time than it took to entrain the network.

This behavior mimics recent experiments in filamin cross-linked networks. Filamin provides a high level of compliance to a network ($\gamma_0 > 0.5$) without substantial cross-link unbinding. This allows large scale rearrangements to take place without driving

very much cross-link slip, similar to the conditions in section 4.1.2. However, if we force individual filaments to undergo a strongly nonlinear stiffening at strains above 5%, we find an interesting long term strain storage.

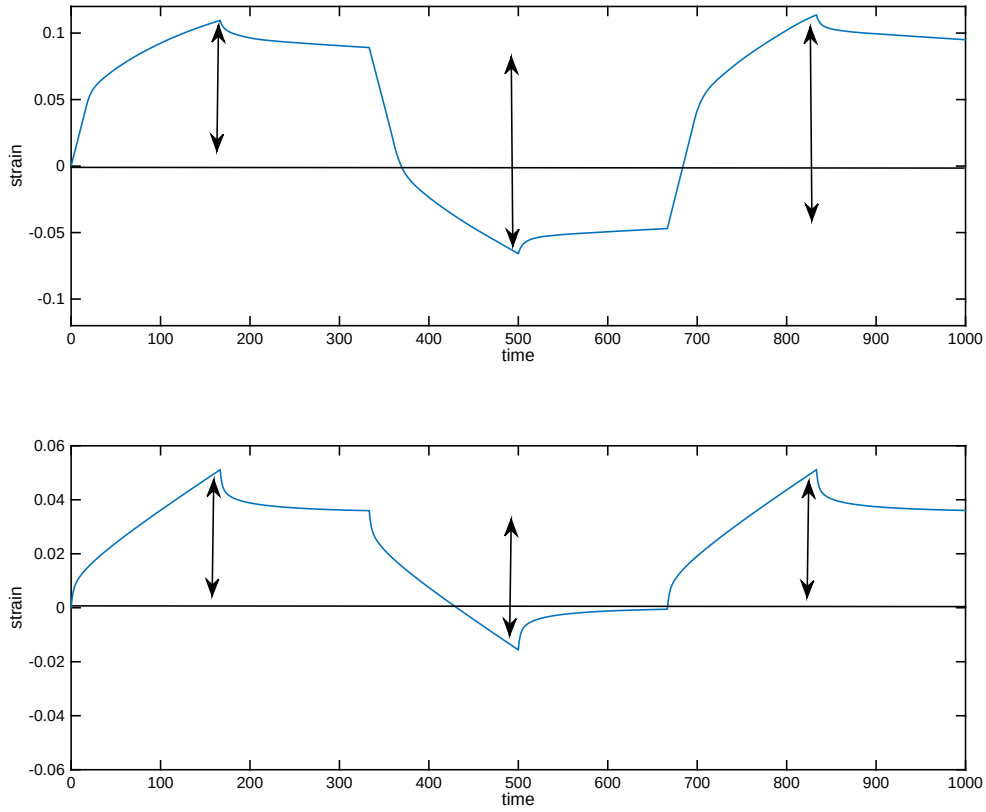


Figure 4.8: Creep curves in the presence of reversing applied stress for (a) nonlinear extension or (b) linear extension. Note that for linear filaments the induced strain returns to approximately 0 after a complete cycle, while in the nonlinear case the cycle is not completely reversible.

Figure 4.8 shows that the strain storage occurs, but I will need time for further study to build the full analytical picture of where when and why this happens in my model.

4.2 Summary and Conclusions

We have proposed a simplified effective friction model for understanding 2D cross-linked networks. Our model extends previous Mikado and lattice models to include effects of cross-link relaxation. We expect that our model can confer insights into mechanisms of network stress relaxation in quasi-2D networks such as those found in *in vitro* actin monolayer experiments[71] as well as in eukaryotic actomyosin cortices[64].

Our model is the first to address the plausible dependence of network effective viscosity on network structural properties. This led to a derivation of an estimate for the long timescale creep rate of networks under constant stress. Although this derivation neglects possible frequency dependence at short timescales, this finding offers a potential framework for addressing the dependence of network deformation rate on filament concentration and length.

Additionally, our simulations suggest that, in the presence of constant shear stress, cross-link friction will also produce a long-lived phase of sublinear creep as filaments relax from their affine stretched position. While this phase may transiently resemble more explicit 3D models such as [11], it is clear that our model differs by predicting that network will achieve a constant effective viscosity more rapidly. In particular, we predict that this relaxation will occur at a rate similar to that of rate of cross-link slip derived strain and will therefore be negligible after the network has slipped by roughly ten times the magnitude of the purely affine mechanical deformation.

In building our model we have neglected any other sources of potential mechanical relaxation in order to simplify out analysis. In the future, we hope to extend our model to include biochemically driven forms of relaxation such as filament turnover or regulated cross-link unbinding.

This model forms a basis for addressing 2D filament network deformation, and it proposes a simplified formulation of important qualitative properties. In this way we are able to address potentially general phases of network deformation and delineate what network properties may give rise to them. This may provide an important starting point for addressing the general importance of network structure in more complex networks containing active elements.

4.3 Network Tearing under Extensional Stress

4.3.1 Extensional Thinning and Network Tearing

For moderate extensional stresses, the rigid filament approximation of the effective viscosity simply picks up a different geometrical factor out front.

However, at higher stress and in the presence of different things happen.

$$\frac{\partial l_c}{dt} = l_c\dot{\gamma} = \frac{l_c\sigma}{\eta} \sim l_c^3\frac{\sigma}{L^2\xi} \tag{4.6}$$

We can see that the rate of network thinning accelerates as we would expect. When the network reaches some minimum connectivity we assume that it stops behaving as a continuum material and the network tears irreversibly.

$$\tau_{break} = \frac{\eta_{eff}}{2\sigma} \cdot \left(1 - \frac{l_c^2}{l_{break}^2}\right) \tag{4.7}$$

This provides us with an estimate of the timescale of catastrophic breakdown for a network with a given initial architecture and molecular drag.

4.3.2 Tearing Events During Extensional Strain

This behavior is caused primarily by the low density network undergoing tearing events that interrupt global connectedness.

4.4 Deriving Molecular Drag Coefficients

Thus far, the idea of a molecular drag coefficient was taken as a phenomenological, measured parameter for a given experimental setup. While this is a sufficient pragmatic justification, it's useful to try to motivate the quantitative value of this drag coefficient by connecting it to the underlying cross-link properties of binding affinity, concentration, and extensibility.

To do this we'll imagine the simplified case of two cross linkers sliding past each other in one dimension. In this case, assume that we have an equilibrium number

of bound cross-linkers, n_B, each of which is displaced from its equilibrium length by some distance x. Each cross linker unbinds with rate k_{off} and rebinds at it's relaxed position ($x = 0$) with rate k_{on}. At the same time, all the cross linkers are being pulled from their relaxed position at a rate, v, which is simply the rate at which the filaments are sliding past each other.

We can write the differential equation for the change in the density of cross-links, ρ, at displacement x as they are pulled upon, bind, and unbind.

$$\frac{\partial \rho}{\partial t} = -k_{off}\rho(x) - v\frac{\partial \rho}{\partial x} + k_{on}\delta(x) \tag{4.8}$$

Recognizing that $\int \rho(x) = n_B$ implies $k_{on} = k_{off}n_B$, we can find the steady state solution

$$\rho(x) = \frac{n_b k_{off}}{v} \cdot exp\left(-\frac{k_{off}}{v}x\right) \tag{4.9}$$

If each cross-link has a spring constant μ_c, then we can equate the force on all cross-links to the applied force that is sliding the filaments past each other. Realistically, the spring constant and binding affinity would be functions of the cross-link stretch, but here we are taking them as approximately constant.

$$\int_0^\infty \rho(x)\mu_c x dx = v\frac{\mu_c n_B}{k_{off}} = F_{app} \tag{4.10}$$

a

Therefore, the term next to v, (i.e. $\frac{\mu_c n_B}{k_{off}}$) would be equal to our molecular drag coefficient, ξ. Assuming approximately 1 cross link per filament overlap, and using parameter estimates culled from Ferrer et al., we build the following table of estimates for ξ.

cross-linker type	α-actinin	filamin-A
dissociation constant (s^{-1})	0.4	0.6
spring constant ($nN/\mu m$)	455	820
drag coefficient, ξ ($\frac{nN \cdot s}{\mu m}$)	182	492

This molecular description assumed both a constant off-rate and linear force ex-

tension of cross-links. In the event that binding kinetics are regulated by the state of extension, we would expect (based on Rf) to find a region that exhibits a stick-slip behavior instead of the smooth. Depending on the nature of any coupling between cross-links local stick-slip could either give rise to a global stick-slip behavior or a heterogenous mixture of stuck and sliding cross-links. It would be interesting to explore this topic further in the future, but in the present analysis, we choose to ignore complications from these nonlinear effects.

CHAPTER 5
FILAMENT RECYCLING AND SUSTAINED CONTRACTILE FLOWS IN AN ACTOMYOSIN NETWORK

5.1 Results

The goal of this study is to understand how cortical flow is shaped by the simultaneous dependencies of active stress and effective viscosity on filament turnover, crosslink drag and on "network parameters" that control filament density, elasticity and motor activity. We approach this in three steps: First, we analyze the passive deformation of a cross-linked network in response to an externally applied stress; we identify regimes in which the network response is effectively viscous and characterize the dependence of effective viscosity on network parameters and filament turnover. Second, we analyze the buildup and dissipation of active stress in cross-linked networks with active motors, as they contract against an external resistance; we identify conditions under which the network can produce sustained stress at steady state, and characterize how steady state stress depends on network parameters and filament turnover. Finally, we confirm that the dependencies of active stress and effective viscosity on network parameters and filament turnover are sufficient to predict the dynamics of networks undergoing steady state flow in response to spatial gradients of motor activity.

5.1.1 Filament turnover allows and tunes effectively viscous steady state flow.

Networks with passive cross-links and no filament turnover undergo three stages of deformation in response to an extensional force. To characterize the passive response of a cross-linked filament network without filament recycling,

we simulated a simple uniaxial strain experiment in which we pinned the network at one end, imposed an external stress at the opposite end, and then quantified network strain and internal stress as a function of time (Fig. 3.1E). The typical response occurred in three qualitatively distinct phases (Fig. 5.1A,C). At short times the response was viscoelastic, with a rapid buildup of internal stress and a rapid ∼exponential approach to a fixed strain (Fast viscoelastic response to extensional stress. Plots of normalized strain vs time during the elastic phase of deformation in passive networks under extensional stress. Measured strain is normalized by the equilibrium strain predicted for a network of elastic filaments without crosslink slip $\gamma_{eq} = \sigma/G_0 = \sigma/(2\mu/l_c)$. A), which represents the elastic limit in the absence of cross-link slip predicted by [37]. At intermediate times, the local stress and strain rate were approximately constant across the network (Fig. 5.1B), and the response was effectively viscous; internal stress remained constant while the network continued to deform slowly and continuously with nearly constant strain rate (shown as dashed line in Fig. 5.1C) as filaments slip past one another against the effective cross-link drag. In this regime, we can quantify effective viscosity, η_c, as the ratio of applied stress to the measured strain rate. Finally, as the network strain approached a critical value ($\sim$ 30% for the simulation in Fig. 5.1), strain thinning lead to decreased network connectivity, local tearing, and rapid acceleration of the network deformation (see inset in Fig. 5.1C).

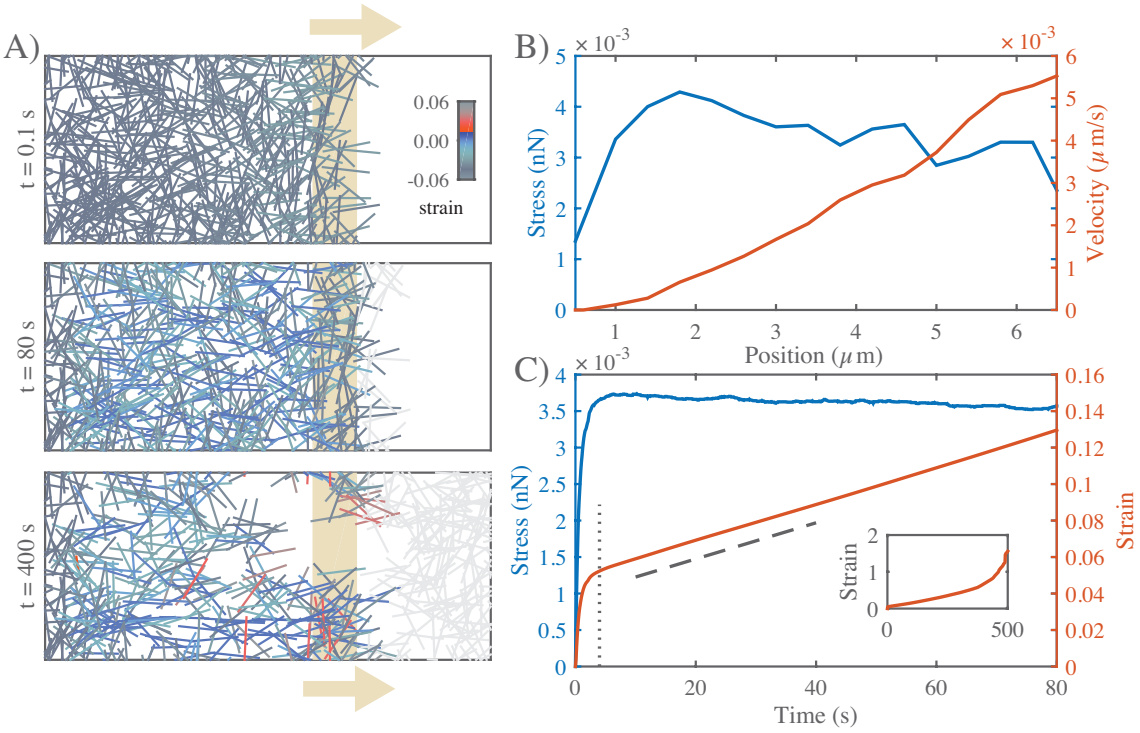


Figure 5.1: Networks with passive cross-links and no filament turnover undergo three stages of deformation in response to an extensional stress. **A)** Three successive time points from a simulation of a $4 \times 6.6\,\mu m$ network deforming under an applied stress of 0.005 $nN/\mu m$. Stress (tan arrows) is applied to filaments in the region indicated by the tan bar. In this and all subsequent figures, filaments are color-coded with respect to state of strain (blue = tension, red = compression). Network parameters: $L = 1\,\mu m$, $l_c = 0.3\,\mu m$, $\xi = 100\,nN \cdot s/\mu m$. **B)** Mean filament stress and velocity profiles for the network in (a) at t=88s. Note that the stress is nearly constant and the velocity is nearly linear as predicted for a viscous fluid under extension. **C)** Plots of the mean stress and strain vs time for the simulation in (a), illustrating the three stages of deformation: (i) A fast initial deformation accompanies rapid buildup of internal network stress; (ii) after a characteristic time τ_c (indicated by vertical dotted line) the network deforms at a constant rate, i.e. with a constant effective viscosity, η_c, given by the slope of the dashed line; (iii) at long times, the network undergoes strain thinning and tearing (see inset)

Network architecture sets the rate and timescales of deformation. To characterize how effective viscosity and the timescale for transition to effectively viscous behavior depend on network architecture and cross-link dynamics, we simulated a uniaxial stress test, holding the applied stress constant, while varying filament length L, density l_c, elastic modulus μ_e and cross link drag ξ (see S1 Table). We measured the elastic modulus, G_0, the effective viscosity, η_c, and the timescale τ_c for transition from viscoelastic to effectively viscous behavior, and compared these to theoretical predictions. We observed a transition from viscoelastic to effectively viscous deformation for the entire range of parameter values that we sampled. Our estimate of G_0 from simulation agreed well with the closed form solution $G_0 \sim \mu/l_c$ predicted by a previous theoretical model [37] for networks of semi-flexible filaments with irreversible cross-links (Fig. 5.2B).

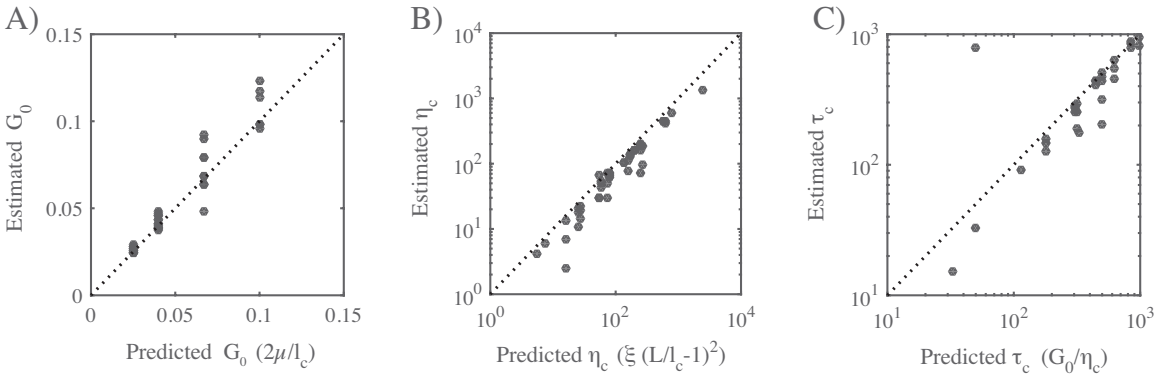


Figure 5.2: Network architecture sets the rate and timescales of deformation. **(a-c)** Comparison of predicted and simulated values for: **A)** the bulk elastic modulus G_0, **B)** the effective viscosity η_c and **C)** the timescale for transition from viscoelastic to viscous behavior τ_c, given by the ratio of the bulk elastic modulus G_0 to effective viscosity, η_c. Dotted lines indicates the relationships predicted by theory.

A simple theoretical analysis of filament networks with frictional cross link slip, operating in the intermediate viscous regime (see S1 Appendix A.2), predicted that the effective viscosity η_c should be proportional to the cross-link drag coefficient and to the square of the number of cross-links per filament:

$$\eta_c = 4\pi\xi \left(\frac{L}{l_c} - 1\right)^2 \tag{5.1}$$

As shown in Fig. 5.2B, our simulations agree well with this prediction for a large range of sampled network parameters. Finally, for many linear viscoelastic materials, the ratio of effective viscosity to the elastic modulus η_c/G_0 sets the timescale for transition from elastic to viscous behavior[65]. Combining our approximations for G_0 and η_c, we predict a transition time, $\tau_c \approx L^2\xi/l_c\mu$. Measuring the time at which the strain rate became nearly constant (i.e. $\gamma \sim t^n$ with $n > 0.8$) yields an estimate of τ_c that agrees well with this prediction over the entire range of sampled parameters (Fig. 5.2C). Thus the passive response of filament networks with frictional cross link drag is well-described on short (viscoelastic) to intermediate (viscous) timescales by an elastic modulus G_0, an effective viscosity η_c, and a transition timescale τ_c, with well-defined dependencies on network parameters. However, without filament turnover, strain thinning and network tearing limits the extent of viscous deformation to small strains.

Filament turnover allows sustained large-scale viscous flow and defines two distinct flow regimes. To characterize how filament turnover shapes the passive network response to an applied force, we introduced a simple form of turnover in which entire filaments disappear at a rate $k_{diss}\rho$, where ρ is the filament density, and new unstrained filaments appear with a fixed rate per unit area, k_{app}. In a non-deforming network, filament density will equilibrate to a steady state value, $\rho_0 = k_{ass}/k_{diss}$, with time constant $\tau_r = 1/k_{diss}$. However, in networks deforming under extensional stress, the density will be set by a competition between strain thinning and density equilibration via turnover.

We simulated a uniaxial stress test for different values of τ_r, while holding all other parameters fixed (Fig. 5.3A-C). For large τ_r, as described above, the network undergoes strain thinning and ultimately tears. Decreasing τ_r increases the rate at which the network equilibrates towards a steady state density ρ_0. However, it also increases the rate of deformation and thus the rate of strain thinning (Fig. 5.3B). We found that the former effect dominates, such that below a critical value $\tau_r = \tau_{crit}$, the network can achieve a steady state characterized by a fixed density and a constant strain rate (Filament turnover rescues strain thinning. **a)** Plots of strain vs time

for different turnover times (see inset in (b)). Note the increase in strain rates with decreasing turnover time. **b)** Plots of filament density vs strain for different turnover times τ_r. For intermediate τ_r, simulations predict progressive strain thinning, but at a lower rate than in the complete absence of recycling. For higher τ_r, densities approach steady state values at longer times.). Simple calculations (S1 Appendix A.3) show that the critical value of τ_r is approximately:

$$\tau_{crit} = \frac{\xi\left(\sqrt{\frac{L}{l_c}} - 1\right)^3}{\sigma}. \tag{5.2}$$

where σ is the applied stress, L/l_c the linear cross link density, and ξ is the effective crosslink drag.

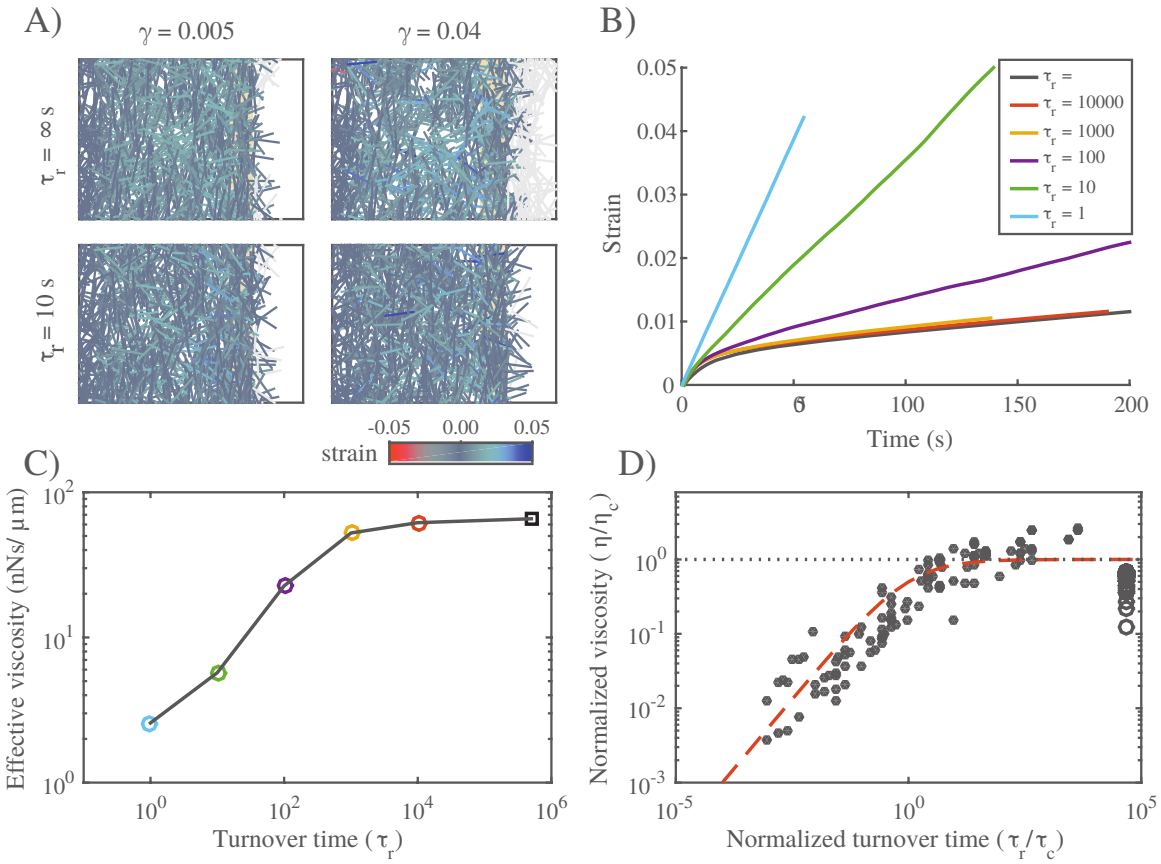


Figure 5.3: Filament recycling defines two regimes of effectively viscous flow. **A)** Comparison of $20 \times 12\mu m$ networks under 0.001 $nN/\mu m$ extensional stress without (top) and with (bottom) filament turnover. Both images are taken when the networks had reached a net strain of 0.04. For clarity, filaments that leave the domain of applied stress are greyed out. **B)** Plots of strain vs time for identical networks with different rates of filament turnover. Network parameters: $L = 5\ \mu m$, $l_c = 0.5\ \mu m$, $\xi = 10\ nN \cdot s/\mu m$. **C)** Plot of effective viscosity vs turnover time derived from the simulations shown in panel b. Square dot is the $\tau_r = \infty$ condition. **D)** Plot of normalized effective viscosity (η/η_c) vs normalized turnover time (τ_r/τ_c) for a large range of network parameters and turnover times. For $tau_r \ll \tau_c$, the viscosity of the network becomes dependent on recycling time. Red dashed line indicates the approximation given in equation 5.3 for $m = 3/4$.

For $\tau_r < \tau_{crit}$, we observed two distinct steady state flow regimes (Fig. 5.3B,C). For intermediate values of τ_r, effective viscosity remains constant with decreasing τ_r. However, below a certain value of τ_r ($\approx 10^3$ for the parameters used in Fig. 5.3C), effective viscosity decreased monotonically with further decreases in τ_r. To

understand what sets the timescale for transition between these two regimes, we measured effective viscosity at steady steady for a wide range of network parameters (L, μ, l_c), crosslink drags (ξ) and filament turnover times (Fig. 5.3D). Strikingly, when we plotted the normalized effective viscosity η_r/η_c vs a normalized recycling rate τ_r/τ_c for all parameter values, the data collapsed onto a single curve, with a transition at $\tau_r \approx \tau_c$ between an intermediate turnover regime in which effective viscosity is independent of τ_r and an high turnover regime in which effective viscosity falls monotonically with decreasing τ_r/τ_c (Fig. 5.3D).

This biphasic dependence of effective viscosity on filament turnover can be understood intuitively as follows: As new filaments are born, they become progressively stressed as they stretch and reorient under local influence of surrounding filaments, eventually reaching an elastic limit where their contribution to resisting network deformation is determined by effective crosslink drag. The time to reach this limit is about the same as the time, τ_c, for an entire network of initially unstrained filaments to reach an elastic limit during the initial viscoelastic response to uniaxial stress, as shown in Fig. 2b. For $\tau_r < \tau_c$, individual filaments do not have time, on average, to reach the elastic limit before turning over; thus the deformation rate is determined by the elastic resistance of partially strained filaments, which increases with lifetime up to $\tau_r = \tau_c$. For $\tau_r > \tau_c$, the deformation rate is largely determined by cross-link resistance to sliding of maximally strained filaments, and the effective viscosity is insensitive to further increase in τ_r.

These results complement and extend a previous computational study of irreversibly cross-linked networks of treadmilling filaments deforming under extensional stress[46]. Kim et al identified two regimes of effectively viscous deformation: a "stress-dependent" regime in which filaments turnover before they become strained to an elastic limit and deformation rate is proportional to both applied stress and turnover rate; and a "stress-independent" regime in which filaments reach an elastic limit before turning over and deformation rate depends only on the turnover rate. The fast and intermediate turnover regimes that we observe here correspond to the stress-dependent and independent regimes described by Kim et al, but with a key difference. Without filament turnover, Kim et al's model predicts that a network

cannot deform beyond its elastic limit. In contrast, our model predicts viscous flow at low turnover, governed by an effective viscosity that is set by cross-link density and effective drag. Thus our model provides a self-consistent framework for understanding how crosslink unbinding and filament turnover contribute separately to viscous flow and connects these contributions directly to previous theoretical descriptions of cross-linked networks of semi-flexible filaments.

In summary, our simulations predict that filament turnover allows networks to undergo viscous deformation indefinitely, without tearing, over a wide range of different effective viscosities and deformation rates. For $\tau_r < \tau_{crit}$, this behavior can be summarized by an equation of the form:

$$\eta = \frac{\eta_c}{1 + (\tau_c/\tau_r)^m} \tag{5.3}$$

For $\tau_r \gg \tau_c$, $\eta \approx \eta_c$: effective viscosity depends on crosslink density and effective crosslink drag, independent of changes in recycling rate. For $\tau_r \ll \tau_c$, effective viscosity is governed by the level of elastic stress on network filaments, and becomes strongly dependent on filament lifetime: $\eta \sim \eta_c(\tau_r/\tau_c)^m$. The origins of the $m = 3/4$ scaling remains unclear (see Discussion).

5.1.2 Filament turnover allows persistent stress buildup in active networks

In the absence of filament turnover, active networks with free boundaries contract and then stall against passive resistance to network compression. Previous work [51, 71, 48] identifies asymmetric filament compliance and spatial heterogeneity in motor activity as minimal requirements for macroscopic contraction of disordered networks. To confirm that our simple implementation of these two requirements (see Models section) is sufficient for macroscopic contraction, we simulated active networks that are unconstrained by external attachments, varying filament length, density, crosslink drag and motor activity. We observed qualitatively similar results for all choices of parameter values: Turning on motor activity in an initially unstrained network induced rapid initial contraction, followed by a slower buildup

of compressive stress (and strain) on individual filaments, and an ∼exponential approach to stall (Fig. 5.4). The time to stall, τ_s, scaled as $L\xi/\upsilon$ (Mechanical properties of active networks. **a)** Time for freely contracting networks to reach maximum strain, τ_s, scales with $L\xi/\upsilon$. **b)** Free contraction requires asymmetric filament compliance, and total network strain increases with the applied myosin force υ. Note that the maximum contraction approaches an asymptotic limit as the stiffness asymmetry approaches a ratio of ~ 100. **c)** Maximum stress achieved during isometric contraction, σ_m, scales approximately with $\sqrt{\mu_e \upsilon}/l_c$. **d)** Time to reach max stress during isometric contraction scales approximately with $L\xi/\sqrt{\mu_e \upsilon}$. Scalings for τ_s, σ_m and τ_m were determined empirically by trial and error, guided by dimensional analysis. A). On even longer timescales, polarity sorting of individual filaments, as previously described [76, 69, 74, 87] lead to network expansion (see S2 Video).

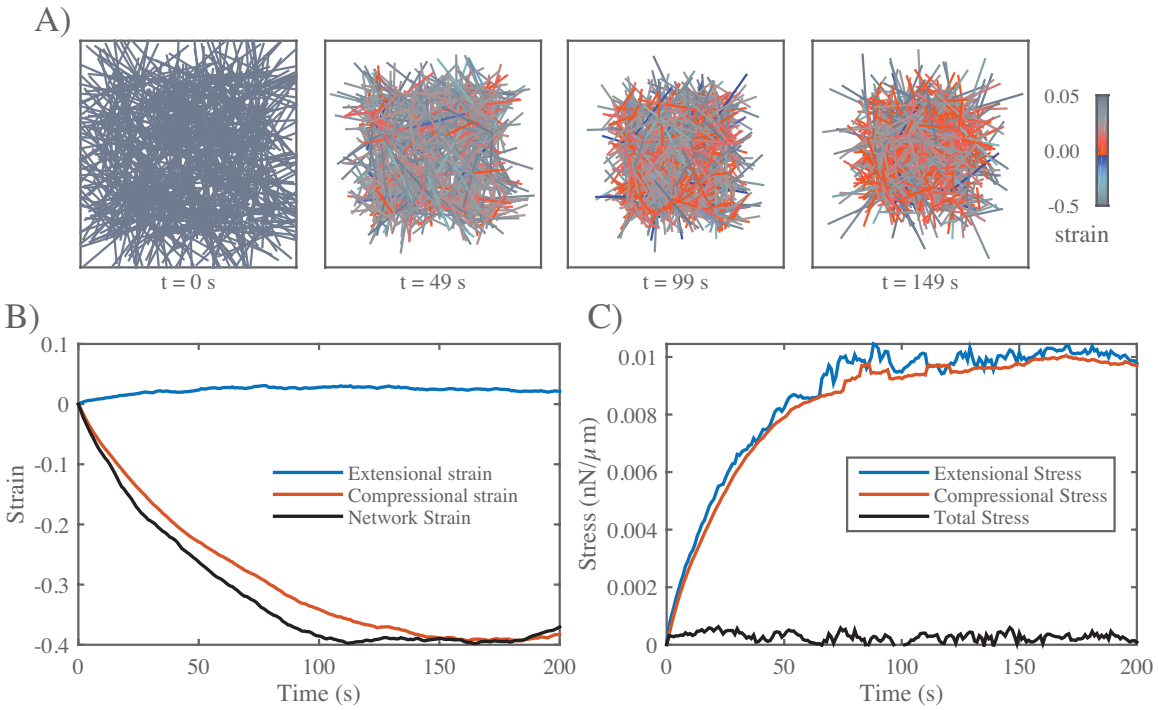


Figure 5.4: In the absence of filament turnover, active networks with free boundaries contract and then stall against passive resistance to network compression. **A)** Simulation of an active network with free boundaries. Colors represent strain on individual filaments as in previous figures. Note the buildup of compressive strain as contraction approaches stall between 100 s and 150 s. Network parameters: $L = 5\ \mu m$, $l_c = 0.3\ \mu m$, $\xi = 1\ nN \cdot s/\mu m$, $\upsilon = 0.1\ nN$. **B)** Plots showing time evolution of total network strain (black) and the average extensional (blue) or compressive (red) strain on individual filaments. **C)** Plots showing time evolution of total (black) extensional (blue) or compressive (red) stress. Note that extensional and compressive stress remain balanced as compressive resistance builds during network contraction.

During the rapid initial contraction, the increase in network strain closely matched the increase in mean compressive strain on individual filaments Fig. 5.4B, as predicted theoretically [51, 50] and observed experimentally[71]. Contraction required asymmetric filament compliance and spatial heterogeneity of motor activity ($\mu_e/\mu_c > 1$, $\phi < 1$, Mechanical properties of active networks. **a)** Time for freely contracting networks to reach maximum strain, τ_s, scales with $L\xi/\upsilon$. **b)** Free contraction requires asymmetric filament compliance, and total network strain increases with the applied myosin force υ. Note that the maximum contraction approaches an asymptotic limit

as the stiffness asymmetry approaches a ratio of ~ 100. **c)** Maximum stress achieved during isometric contraction, σ_m, scales approximately with $\sqrt{\mu_e \upsilon}/l_c$. **d)** Time to reach max stress during isometric contraction scales approximately with $L\xi/\sqrt{\mu_e \upsilon}$. Scalings for τ_s, σ_m and τ_m were determined empirically by trial and error, guided by dimensional analysis. B). Thus our model captures a minimal mechanism for bulk contractility in disordered networks through asymmetric filament compliance and dispersion of motor activity. However, in the absence of turnover, contraction is limited by internal buildup of compressive resistance and the dissipative effects of polarity sorting.

Active networks cannot sustain stress against a fixed boundary in the absence of filament turnover. During cortical flow, regions with high motor activity contract against a passive resistance from neighboring regions with lower motor activity. To understand how the active stresses that drive cortical flow are shaped by motor activity and network remodeling, we analyzed the buildup and maintenance of contractile stress in active networks contracting against a rigid boundary. We simulated active networks contracting from an initially unstressed state against a fixed boundary (Fig. 5.5A,B), and monitored the time evolution of mean extensional (blue), compressional (red) and total (black) stress on network filaments (Fig. 5.5C,D). We focused initially on the scenario in which there is no, or very slow, filament turnover, sampling a range of parameter values controlling filament length and density, motor activity, and crosslink drag.

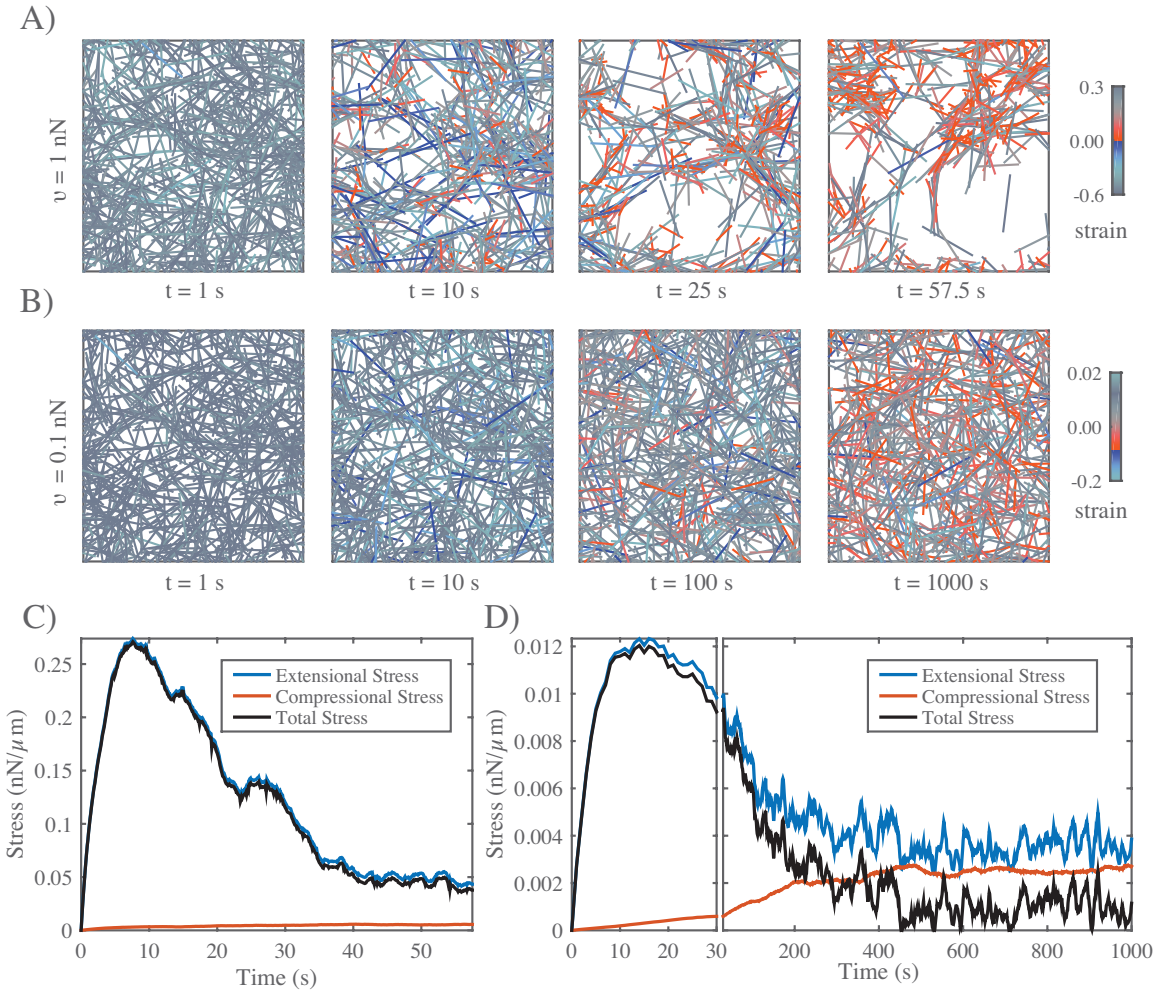


Figure 5.5: In the absence of filament turnover, active networks cannot sustain continuous stress against a fixed boundary. **A)** Simulation of an active network with fixed boundaries. Rearrangement of network filaments by motor activity leads to rapid loss of network connectivity. Network parameters: $L = 5\ \mu m$, $l_c = 0.3\ \mu m$, $\xi = 1\ nN \cdot s/\mu m$, $\upsilon = 1\ nN$. **B)** Simulation of the same network, with the same parameter values, except with ten-fold lower motor activity $\upsilon = 0.1\, nN$. In this case, connectivity is preserved, but there is a progressive buildup of compressive strain on individual filaments. **C)** Plots of total network stress and the average extensional (blue) and compressive (red) stress on individual filaments for the simulation shown in (a). Rapid buildup of extensional stress allows the network transiently to exert force on its boundary, but this force is rapidly dissipated as network connectivity breaks down. **D)** Plots of total network stress and the average extensional (blue) and compressive (red) stress on individual filaments for the simulation shown in (b). Rapid buildup of extensional stress allows the network transiently to exert force on its boundary, but this force is dissipated at longer times as decreasing extensional stress and increasing compressive stress approach balance. Note the different time scales used for plots and subplots in **C)** and **D)** to emphasize the similar timescales for force buildup, but very different timescales for force dissipation.

We observed a similar behavior in all cases: total stress built rapidly to a peak value σ_m, and then decayed towards zero (Fig. 5.5C,D). The rapid initial increase in total stress was determined largely by the rapid buildup of extensional stress (Fig. 5.5C,D) on a subset of network filaments (Fig. 5.5A,B $t = 10s$). The subsequent decay involved two different forms of local remodeling: under some conditions, e.g. for higher motor activity (e.g. Fig. 5.5A,C), the decay was associated with rapid local tearing and fragmentation, leading to global loss of network connectivity as described previously both in simulations[61] and *in vitro* experiments [1]. However, for many parameters, (e.g. for higher motor activity as in Fig. 5.5B,D), the decay in stress occurred with little or no loss of global connectivity. Instead, local filament rearrangements changed the balance of extensile vs compressive forces along individual filaments, leading to a slow decrease in the average extensional stress, and a correspondingly slow increase in the compressional stress, on individual filaments (see Fig. 5.5D).

Combining dimensional analysis with trial and error, we were able to find empirical scaling relationships describing the dependence of maximum stress σ_m and the time to reach maximum stress τ_m on network parameters and effective crosslink drag ($\sigma_m \sim \sqrt{\mu_e \upsilon}/l_c$, $\tau_m \sim L\xi/\sqrt{\mu_e \upsilon}$, Mechanical properties of active networks. **a)** Time for freely contracting networks to reach maximum strain, τ_s, scales with $L\xi/\upsilon$. **b)** Free contraction requires asymmetric filament compliance, and total network strain increases with the applied myosin force υ. Note that the maximum contraction approaches an asymptotic limit as the stiffness asymmetry approaches a ratio of ~ 100. **c)** Maximum stress achieved during isometric contraction, σ_m, scales approximately with $\sqrt{\mu_e \upsilon}/l_c$. **d)** Time to reach max stress during isometric contraction scales approximately with $L\xi/\sqrt{\mu_e \upsilon}$. Scalings for τ_s, σ_m and τ_m were determined empirically by trial and error, guided by dimensional analysis. C,D). Although these relationships should be taken with a grain of salt, they are reasonably consistent with our simple intuition that the peak stress should increase with motor force (υ), extensional modulus (μ_e) and filament density ($1/l_c$), and the time to reach peak stress should increase with crosslink drag (ξ) and decrease with motor force (υ) and extensional modulus (μ_e).

Filament turnover allows active networks to exert sustained stress on a fixed boundary. Regardless of the exact scaling dependencies of σ_m and τ_m on network parameters, these results reveal a fundamental limit on the ability of active networks to sustain force against an external resistance in the absence of filament turnover. To understand how this limit can be overcome by filament turnover, we simulated networks contracting against a fixed boundary from an initially unstressed state, for increasing rates of filament turnover (decreasing τ_r), while holding all other parameter values fixed (Fig. 5.6A-C). While the peak stress decreased monotonically with decreasing τ_r, the steady state stress showed a biphasic response, increasing initially with decreasing τ_r, and then falling off as $\tau_r \to 0$. We observed a biphasic response regardless of how stress decays in the absence of turnover, i.e. whether decay involves loss of network connectivity, or local remodeling without loss of connectivity, or both (Filament turnover prevents tearing of active networks. **a)** An active network undergoing large scale deformations due to active filament rearrangements. **b)** The same network as in (a) but with a shorter filament turnover time. **c)** Plots of internal stress vs time for the network in (a). **d)** Plots of internal stress vs time for the network in (b). and not shown). Significantly, when we plot normalized steady state stress (σ/σ_m) vs normalized recycling time (τ_r/τ_m) for a wide range of network parameters, the data collapse onto a single biphasic response curve, with a peak near $\tau_r/\tau_m = 1$ (Fig. 5.6D). In particular, for $\tau_r < \tau_m$, the scaled data collapsed tightly onto a single curve representing a linear increase in steady state stress with increasing τ_r. For $\tau_r > \tau_m$, the scaling was less consistent, although the trend towards a monotonic decrease with increasing τ_r was clear. These results reveal that filament turnover can rescue the dissipation of active stress during isometric contraction due to network remodeling, and they show that, for a given choice of network parameters, there is an optimal choice of filament lifetime that maximizes steady state stress.

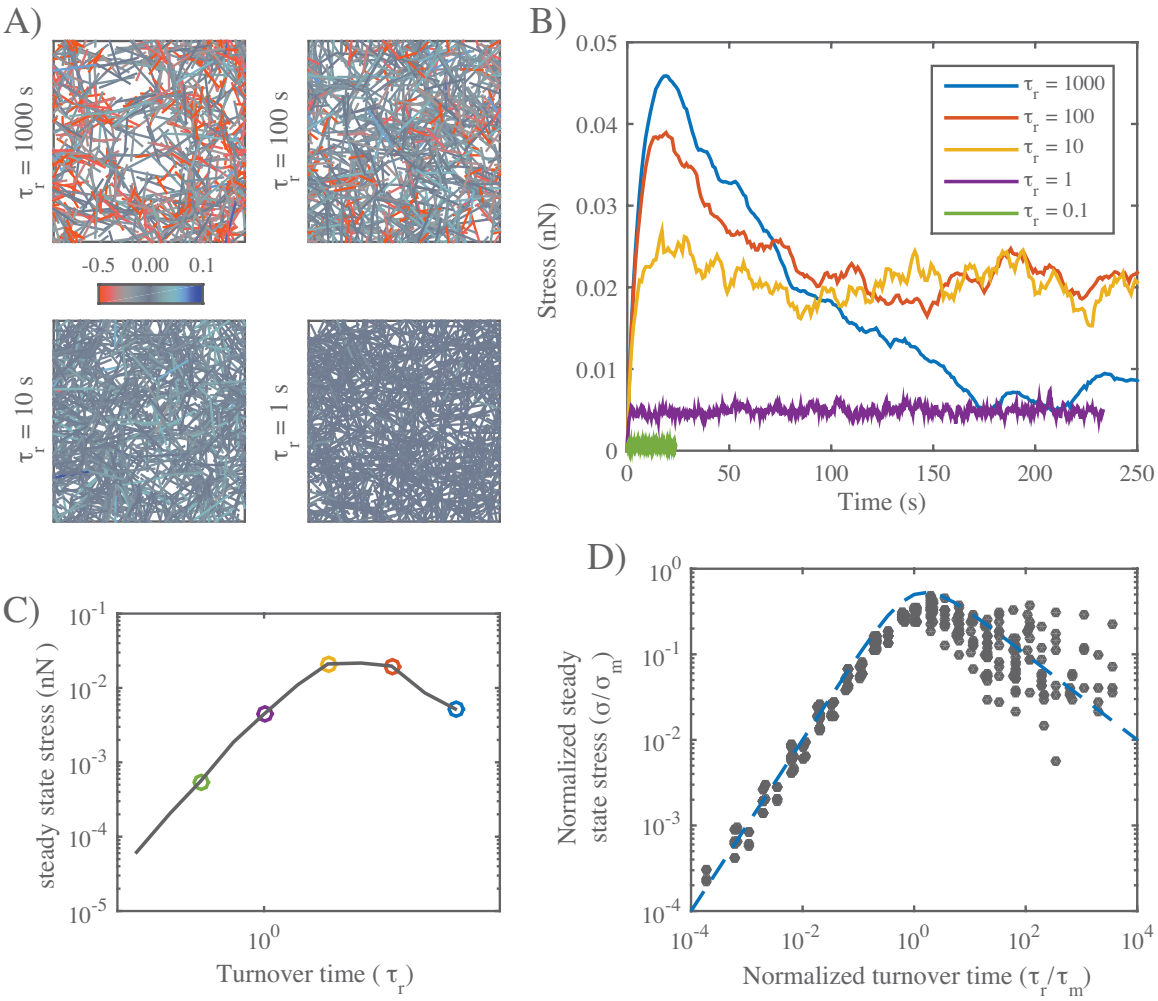


Figure 5.6: Filament turnover allows active networks to exert sustained stress on a fixed boundary. **A)** Snapshots from simulations of active networks with fixed boundaries and different rates of filament turnover. All other parameter values are the same as in Fig. 5.5A. Note the significant buildup of compressive strain and significant remodeling for longer, but not shorter, turnover times. **B)** Plots of net stress exerted by the network on its boundaries for different recycling times; for long-lived filaments, stress is built rapidly, but then dissipates. Decreasing filament lifetimes reduces stress dissipation by replacing compressed with uncompressed filaments, allowing higher levels of steady state stress; for very short lifetimes, stress is reduced, because individual filaments do not have time to build stress before turning over. **C)** Plots of ≈steady state stress estimated from the simulations in **B)** vs turnover time. **D)** Plot of normalized steady state stress vs normalized recycling time for a wide range of network parameters and turnover times. Steady state stress is normalized by the predicted maximum stress σ_m achieved in the absence of filament turnover. Turnover time is normalized by the predicted time to achieve maximum stress τ_m, in the absence of filament turnover. Predictions for σ_m and τ_m were obtained from the phenomenological scaling relations shown in (Fig. 5.12C,D). Dashed blue line indicates the approximation given in equation 5.4 for $n = 1$.

We can understand the biphasic dependence of steady state stress on filament lifetime using the same reasoning applied to the case of passive flow: During steady state contraction, the average filament should build and dissipate active stress on approximately the same schedule as an entire network contracting from an initially unstressed state (Fig. 5.6B). Therefore for $\tau_r < \tau_m$, increasing lifetime should increase the mean stress contributed by each filament. For $\tau_r > \tau_m$, further increases in lifetime should begin to reduce the mean stress contribution. Directly comparing the time-dependent buildup and dissipation of stress in the absence of turnover, with the dependence of steady state stress on τ_r, supports this interpretation (Bimodal dependence on turnover time matches bimodal buildup and dissipation of stress in the absence of turnover. **a)** Bimodal buildup of stress in a network with very slow turnover ($\tau_r = 1000s$). **b)** Steady state stress for networks with same parameters as in (a), but for a range of filament turnover times.)

As for the passive response (i.e. Equation 5.3), we can describe this biphasic dependence phenomenologically with an equation of the form:

$$\sigma_{ss} = \frac{\sigma_m}{(\tau_r/\tau_m)^n + \tau_m/\tau_r} \tag{5.4}$$

where the origins of the exponent n remain unclear.

5.1.3 Filament turnover tunes the balance between active stress buildup and viscous stress relaxation to generate flows

Thus far, we have considered independently how network remodeling controls the passive response to an external stress, or the steady state stress produced by active contraction against an external resistance. We now consider how these two forms of dependence will combine to shape steady state flow produced by spatial gradients of motor activity. We consider a simple scenario in which a network is pinned on either side and motor activity is continuously patterned such that the right half network has uniformly high levels of motor activity (controlled by υ, with $\psi = 0.5$), while the left half network has none. Under these conditions, the right half network will contract continuously against a passive resistance from the left half network. Because

of asymmetric filament compliance, the internal resistance of the right half network to active compression should be negligible compared to the external resistance of the left half network. Thus the steady state flow will be described by:

$$\dot{\gamma} = \frac{\sigma_{ss}}{\eta} \tag{5.5}$$

where σ_{ss} is the active stress generated by the right half-network (less the internal resistance to filament compression), η is the effective viscosity of the left half network and strain rate $\dot{\gamma}$ is measured in the left half-network. Note that strain rate can be related to the steady state flow velocity v at the boundary between right and left halves through $v = \dot{\gamma} D x$. Therefore, we can understand the dependence of flow speed on filament turnover and other parameters using the approximate relationships summarized by equations 5.3 and 5.4 for η and σ_{ss}. As shown in Fig. 5.7, there are two qualitatively distinct possibilities for the dependence of strain rate on τ_r, depending on the relative magnitudes of τ_m and τ_c. In both cases, for fast enough turnover ($\tau_r < \min(\tau_m, \tau_c)$), we expect weak dependence of strain rate on τ_r ($\dot{\gamma} \sim \tau_r^{1/4}$). For all parameter values that we sampled in this study (which were chosen to lie in a physiological range), $\tau_m > \tau_c$. Therefore we predict the dependence of steady state strain rate on τ_r shown in Fig. 5.7A.

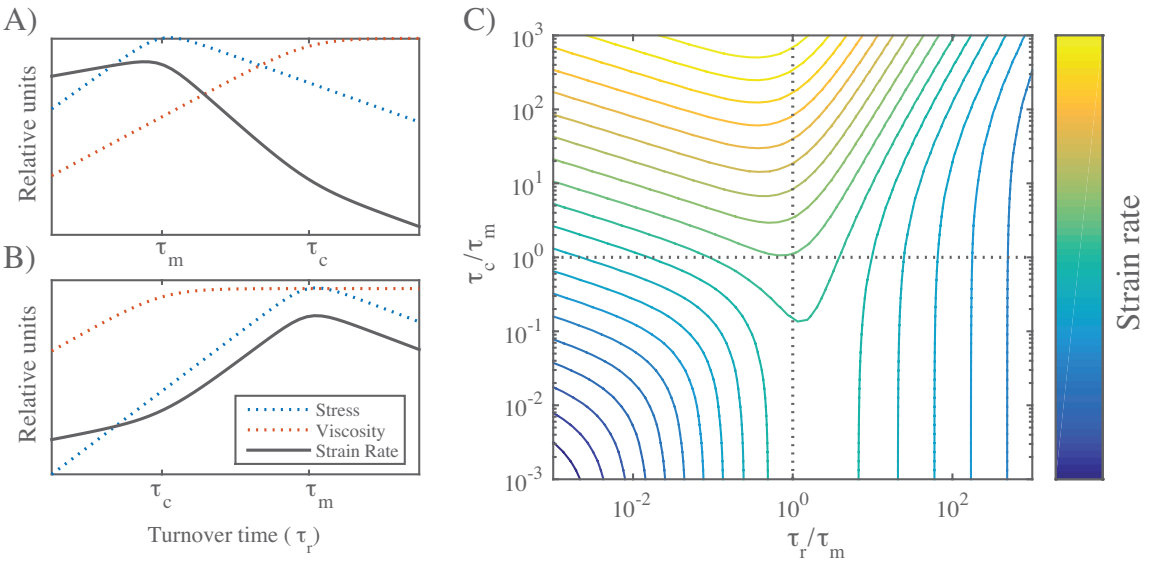


Figure 5.7: Filament recycling tunes the magnitudes of both effective viscosity and steady state stress. **A)** Dependence of steady state stress, effective viscosity, and resulting strain rate on recycling time τ_r under the condition $\tau_m < \tau_c$. **B)** Same as a) but for $\tau_c < \tau_m$. **C)** State space of flow rate dependence relative to the two relaxation timescales, τ_r and τ_c normalized by the stress buildup timescale, τ_m.

To confirm this prediction, we simulated the simple scenario described above for a range of values of τ_r, with all other parameter values initially fixed. As expected, we observed a sharp dependence of steady state flow speeds on filament recycling rate (Fig. 5.8B,C). For very long recycling times, ($\tau_r = 1000s$, dark blue line), there was a rapid initial deformation (contraction of the active domain and dilation of the passive domain), followed by a slow approach to a steady state flow characterized by slow contraction of the right half-domain and a matching dilation of the left half-domain (see Dynamics of steady state flow. Plots of stress and strain vs position for networks in which motor activity is limited to the right-half domain and filament turnover time is either **a)** $\tau_r = 10000$ or **b)** $\tau_r = 10s$. Blue indicates velocity while orange represents total stress, measured as described in the main text.). However, with decreasing values of τ_r, steady state flow speeds increased steadily, before reaching an approximate plateau on which flow speeds varied by less than 15 % over more than two decades of variation in τ_r (Fig. 5.8C).

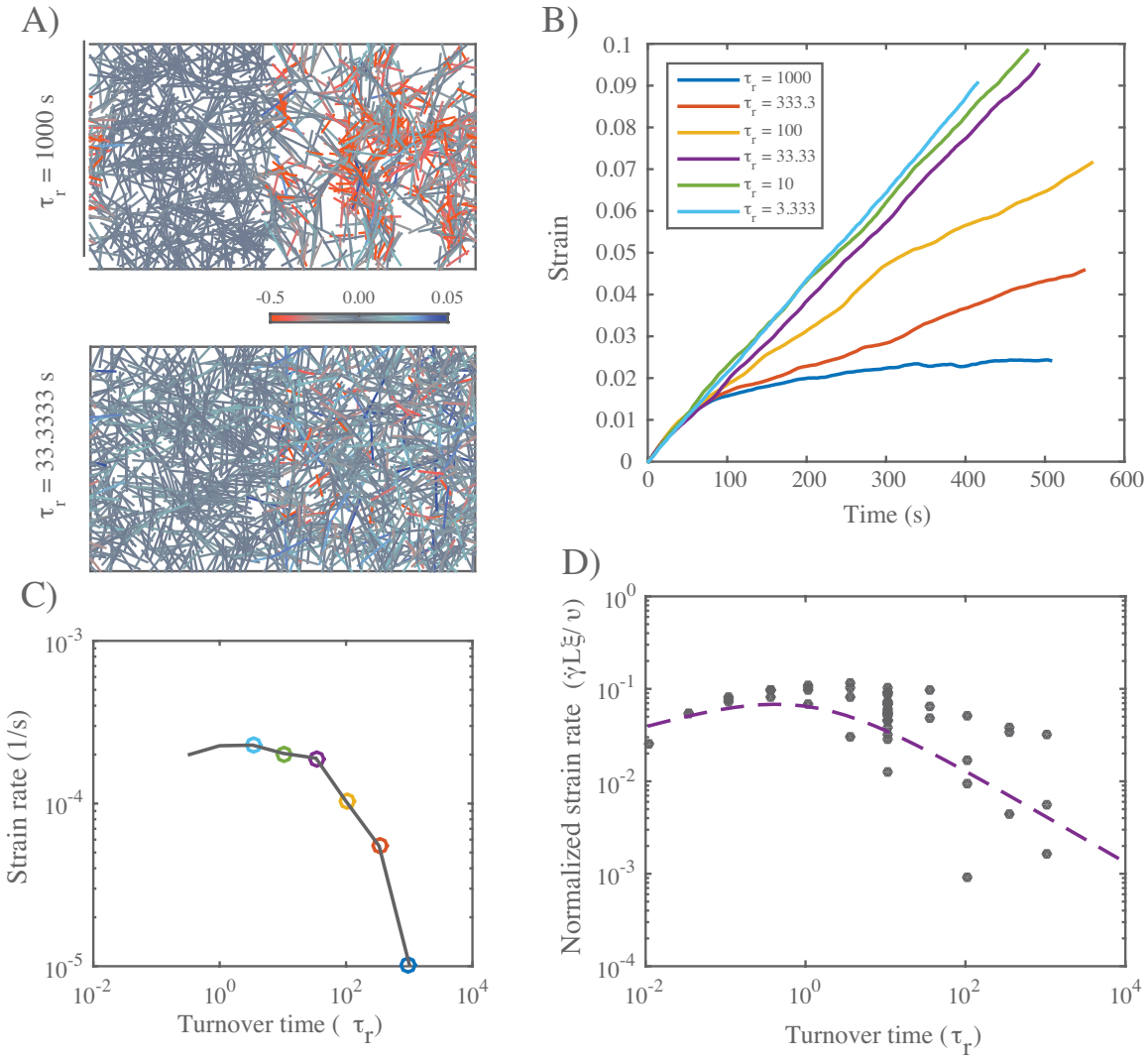


Figure 5.8: Filament recycling allows sustained flows in networks with non-isotropic activity. **A)** Example simulations of non-isotropic networks with long ($\tau_r = 1000$) and short ($\tau_r = 33$) recycling timescales. In these networks the left half of the network is passive while the right half is active. Network parameters are same as in Fig.s 5.5 and 5.6. Importantly, in all simulations $\tau_m < \tau_c$. **B)** Graph of strain for identical networks with varying recycling timescales. With long recycling times, the network stalls; reducing the recycling timescale allows the network to persist in its deformation. However, for the shortest recycling timescales, the steady state strain begins to slowly return to 0 net motion. Measurements are based on the passive side of the network. **C)** Steady state strain rates for the networks in (b). **D)** Graph of network's long-term strain rate as a function of recycling timescale. Dashed line is form of dependence predicted by the theoretical arguments shown in Fig. 5.7.

We repeated these simulations for a wider range of parameter values, and saw similar dependence of $\dot{\gamma}$ on τ_r in all cases. Using equation 5.3 with $\tau_r < \tau_c$ and equation 5.4 with $\tau_r < \tau_m$, and the theoretical or empirical scaling relationships found above for η_c, τ_c, σ_m and τ_m, we predict a simple scaling relationship for $\dot{\gamma}$ (for small τ_r):

$$\dot{\gamma} = \frac{\upsilon}{\xi L} (\tau_r)^{1/4} \tag{5.6}$$

Indeed, when we plot the steady state measurements of $\dot{\gamma}$, normalized by $\upsilon/\xi L$, for all parameter values, the data collapse onto a single curve for small τ_r. Thus. our simulations identify a flow regime, characterized by sufficiently fast filament turnover, in which the steady state flow speed is buffered against variation in turnover, and has a relatively simple dependence on other network parameters.

5.2 Discussion

Cortical flows arise through a dynamic interplay of force production and dissipation within cross-linked actomyosin networks. Here we combined computer simulations with simple theoretical analysis to explore how this interplay depends on motor activity, crosslink dynamics, network architecture and filament turnover. Our results reveal two essential requirements for filament turnover during cortical flow: (a) to allow the continuous relaxation of elastic resistance without catastrophic loss of network connectivity and (b) to prevent the dissipation of active stress through local network rearrangements. We find that biphasic dependencies of active stress and passive relaxation on filament lifetime define multiple modes of steady state flow with distinct dependencies on network parameters and filament turnover.

We identify two distinct modes of passive response to uniaxial stress: a low turnover mode in which filaments strain to an elastic limit before turning over, and effective viscosity depends on crosslink density and effective crosslink friction, and a high turnover mode in which filaments turn over before reaching an elastic limit and effective viscosity is proportional to elastic resistance and $\approx$ proportional to filament

lifetime. We note that the weakly sub-linear dependence of effective viscosity on filament lifetime that we observe in the high turnover regime may simply reflect a failure to capture very local modes of filament deformation, since a previous study [46] in which filaments were represented as connected chains of smaller segments predicted linear dependence of effective viscosity on filament lifetime. While previous studies have emphasized individual roles for cross-link unbinding or filament turnover in stress relaxation [19, 18, 81], here we have capture their distinct contributions within a single self-consistent modeling framework.

Our simulations confirm the theoretical prediction [51, 71, 48] that spatial heterogeneity of motor activity and asymmetric filament compliance are sufficient to support macroscopic contraction of unconstrained networks. However, under isometric conditions, and without filament turnover, our simulations predict that active stress cannot be sustained. On short timescales, motor forces drive local buildup of extensional stress, but on longer timescales, active local filament rearrangements drive local changes in connectivity that lead, invariably, to a decay in active stress. Under some conditions, contractile forces drive networks towards a critically connected state, leading to tearing and fragmentation, as previously described [1, 61]. However, we find that stress decay can also occur without any global loss of connectivity, through a gradual decrease in extensile force and a gradual increase in compressive force along individual filaments. Our results suggest that when filaments can slide relative to one another, the motor forces that produce active stress will inevitably lead to local changes in connectivity that decrease active stress. Thus for contractile networks to maintain isometric tension on long timescales, they must either form stable crosslinks to prevent filament rearrangements, or they must continuously recycle network filaments (or active motors) to renew the local potential for production of active stress.

Indeed, our simulations predict that filament turnover is sufficient for maintenance of active stress. As in the passive case, they predict biphasic dependence of steady state stress on filament turnover: For short-lived filaments ($\tau_r < \tau_m$), steady state

stress increases linearly with filament lifetime because filaments have more time to build towards peak extensional stress before turning over. For longer loved filaments ($\tau_r > \tau_m$), steady state stress decreases monotonically with filament lifetime because local rearrangements decrease the mean contributions of longer lived filaments. These findings imply that for cortical networks that sustain contractile stress under approximately isometric conditions, tuning filament turnover can control the level of active stress, and there will be an optimal turnover rate that maximizes the stress, all other things equal. This may be important, for example in early development, where contractile forces produced by cortical actomyosin networks maintain, or drive slow changes in, cell shape and tissue geometry [81, 35].

For cortical networks that undergo steady state flows driven by spatial gradients of motor activity, our simulations predict that the biphasic dependencies of steady state stress and effective viscosity on filament lifetime define multiple regimes of steady state flow, characterized by different dependencies on filament turnover (and other network parameters). In particular, the ∼linear dependencies of steady state stress and effective viscosity on filament lifetime for short-lived filaments define a fast turnover regime in which steady state flow speeds are buffered against variations in filament lifetime, and are predicted to depend in a simple way on motor activity and crosslink resistance. Measurements of F-actin turnover times in cells that undergo cortical flow [89, 72, 97, 36, 30, 77] suggests that they may indeed operate in this fast turnover regime, and recent studies in *C. elegans* embryos suggests that cortical flow speeds are surprisingly insensitive to depletion of factors (ADF/Cofilin) that govern filament turnover [64], consistent with our model's predictions. Stronger tests of our model's predictions will require more systematic analyses of how flow speeds vary with filament and crosslink densities, motor activities, and filament lifetimes.

5.2.1 Summary of Supplemental Materials

S1 Appendix. Code Reference and Supplementary Methods A.1) Reference to simulation and analysis code. **A.2)** Derivation of effective viscosity. **A.3)**

Derivation of critical turnover timescale for steady state flow

S1 Table. Parameter values. List of parameter values used for each set of simulations.

S1 Fig. Fast viscoelastic response to extensional stress. Plots of normalized strain vs time during the elastic phase of deformation in passive networks under extensional stress. Measured strain is normalized by the equilibrium strain predicted for a network of elastic filaments without crosslink slip $\gamma_{eq} = \sigma/G_0 = \sigma/(2\mu/l_c)$.

S2 Fig. Filament turnover rescues strain thinning. A) Plots of strain vs time for different turnover times (see inset in (b)). Note the increase in strain rates with decreasing turnover time. **B)** Plots of filament density vs strain for different turnover times τ_r. For intermediate τ_r, simulations predict progressive strain thinning, but at a lower rate than in the complete absence of recycling. For higher τ_r, densities approach steady state values at longer times.

S3 Fig. Mechanical properties of active networks. A) Time for freely contracting networks to reach maximum strain, τ_s, scales with $L\xi/\upsilon$. **B)** Free contraction requires asymmetric filament compliance, and total network strain increases with the applied myosin force υ. Note that the maximum contraction approaches an asymptotic limit as the stiffness asymmetry approaches a ratio of ~ 100. **C)** Maximum stress achieved during isometric contraction, σ_m, scales approximately with $\sqrt{\mu_e \upsilon}/l_c$. **D)** Time to reach max stress during isometric contraction scales approximately with $L\xi/\sqrt{\mu_e \upsilon}$. Scalings for τ_s, σ_m and τ_m were determined empirically by trial and error, guided by dimensional analysis.

S4 Fig. Filament turnover prevents tearing of active networks. Plots of normalized strain vs time during the elastic phase of deformation in passive networks under extensional stress. Measured strain is normalized by the equilibrium strain predicted for a network of elastic filaments without crosslink slip $\gamma_{eq} = \sigma/G_0 = \sigma/(2\mu/l_c)$.

S5 Fig. Bimodal dependence on turnover time matches bimodal buildup and dissipation of stress in the absence of turnover. A) Bimodal buildup of stress in a network with very slow turnover ($\tau_r = 1000s$). **B)** Steady state stress for networks with same parameters as in (a), but for a range of filament turnover times.

S6 Fig. Dynamics of steady state flow. Plots of stress and strain vs position for networks in which motor activity is limited to the right-half domain and filament turnover time is either **A)** $\tau_r = 10000$ or **B)** $\tau_r = 10s$. Blue indicates velocity while orange represents total stress, measured as described in the main text.

S1 Video. Extensional strain in passive networks. Movie of simulation setup shown in Fig. 5.1. Colors are the same as in figure.

S2 Video. Active networks contracting with free boundaries. Movie of simulation setup shown in Fig. 5.4. Colors are the same as in figure.

5.3 Supplemental Materials

5.3.1 Simulation and Analysis Code Available Online

All of the simulation and analysis code for generating the figures in this paper is available online. To find the source code please visit our Github repository at

https://github.com/wmcfadden/activnet

5.3.2 Steady-state Approximation of Effective Viscosity

We begin with a calculation of a strain rate estimate of the effective viscosity for a network described by our model in the limit of highly rigid filaments. We carry this out by assuming we have applied a constant stress along a transect of the network. With moderate stresses, we assume the network reaches a steady state affine creep. In this situation, we would find that the stress in the network exactly balances the

sum of the drag-like forces from cross-link slip. So for any transect of length D, we have a force balance equation.

$$\sigma = \frac{1}{D} \sum_{filaments} \sum_{crosslinks} \xi \cdot (\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x})) \tag{5.7}$$

where $\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x})$ is the difference between the velocity of a filament, i, and the velocity of the filament, j, to which it is attached at the cross-link location, $\mathbf{x}$. We can convert the sum over cross-links to an integral over the length using the average density of cross-links, $1/l_c$ and invoking the assumption of (linear order) affine strain rate, $\mathbf{v_i}(\mathbf{x}) - \mathbf{v_j}(\mathbf{x}) = \dot{\gamma} x$. This results in

$$\sigma = \frac{1}{D} \sum_{filaments} \int_0^L \xi \cdot \left(\mathbf{v_i}(\mathbf{s}) - \mathbf{v_j}(\mathbf{s})\right) \frac{ds \cos\theta}{l_c}$$
$$= \sum_{filaments} \frac{\xi \dot{\gamma} L}{l_c} \cos\theta \cdot (x_l + \frac{L}{2} \cos\theta) \tag{5.8}$$

Here we have introduced the variables x_l, and θ to describe the leftmost endpoint and the angular orientation of a given filament respectively. Next, to perform the sum over all filaments we convert this to an integral over all orientations and endpoints that intersect our line of stress. We assume for simplicity that filament stretch and filament alignment are negligible in this low strain approximation. Therefore, the max distance for the leftmost endpoint is the length of a filament, L, and the maximum angle as a function of endpoint is $\arccos(x_l/L)$. The linear density of endpoints is the constant $D/l_c L$ so our integrals can be rewritten as this density over x_l and θ between our maximum and minimum allowed bounds.

$$\sigma = \frac{1}{D} \int_0^L dx_l \int_{-\arccos(\frac{x_l}{L})}^{\arccos(\frac{x_l}{L})} \pi d\theta \frac{\xi \dot{\gamma} L}{l_c} \cdot \frac{D}{L l_c} \cdot (x_l \cos\theta + \frac{L}{2} cos^2\theta) \tag{5.9}$$

Carrying out the integrals and correcting for dangling filament ends leaves us with a relation between stress and strain rate.

$$\sigma = 4\pi \left(\frac{L}{l_c} - 1 \right)^2 \xi \dot{\gamma} \tag{5.10}$$

We recognize the constant of proportionality between stress and strain rate as a viscosity (in 2 dimensions). Therefore, our approximation for the effective viscosity, η_c, at steady state creep in this low strain limit is

$$\sigma = 4\pi \left(\frac{L}{l_c} - 1 \right)^2 \xi \tag{5.11}$$

5.3.3 Critical filament lifetime for steady state filament extension

We seek to determine a critical filament lifetime, τ_{crit} , below which the density of filaments approaches a stable steady state under constant extensional strain. To this end, let ρ be the filament density (i.e. number of filaments per unit area). We consider a simple coarse grained model for how ρ changes as a function of filament assembly k_{ass}, filament disassembly k_{diss}, ρ and strain thinning $\dot{\gamma}\rho$. Using $\rho_0 = \frac{k_{ass}}{k_{diss}}$, $\tau_r = \frac{1}{k_{diss}}$, and $\dot{\gamma} = \frac{\sigma}{\eta_c}$.

$$\frac{d\rho}{dt} = \frac{1}{\tau_r} \left(\rho_0 - \rho - \frac{\sigma \tau_r}{\eta_c(\rho)} \rho \right) \tag{5.12}$$

where $\eta_c = \eta_c(\rho)$ on the right hand side reflects the dependence of effective viscosity on network density. The strength of this dependence determines whether there exists a stable steady state, representing continuous flow. Using $\eta_c(\rho) \sim \xi \left(\frac{L}{l_c(\rho)} - 1 \right)^2$ from above (ignoring the numerical prefactor) and $\rho \sim \frac{2}{L l_c(\rho)}$, we obtain:

$$\frac{d\rho}{dt} = \frac{1}{\tau_r} \left(\rho_0 - \rho - \frac{\sigma \tau_r}{\xi (\rho L^2/2 - 1)^2} \rho \right) \tag{5.13}$$

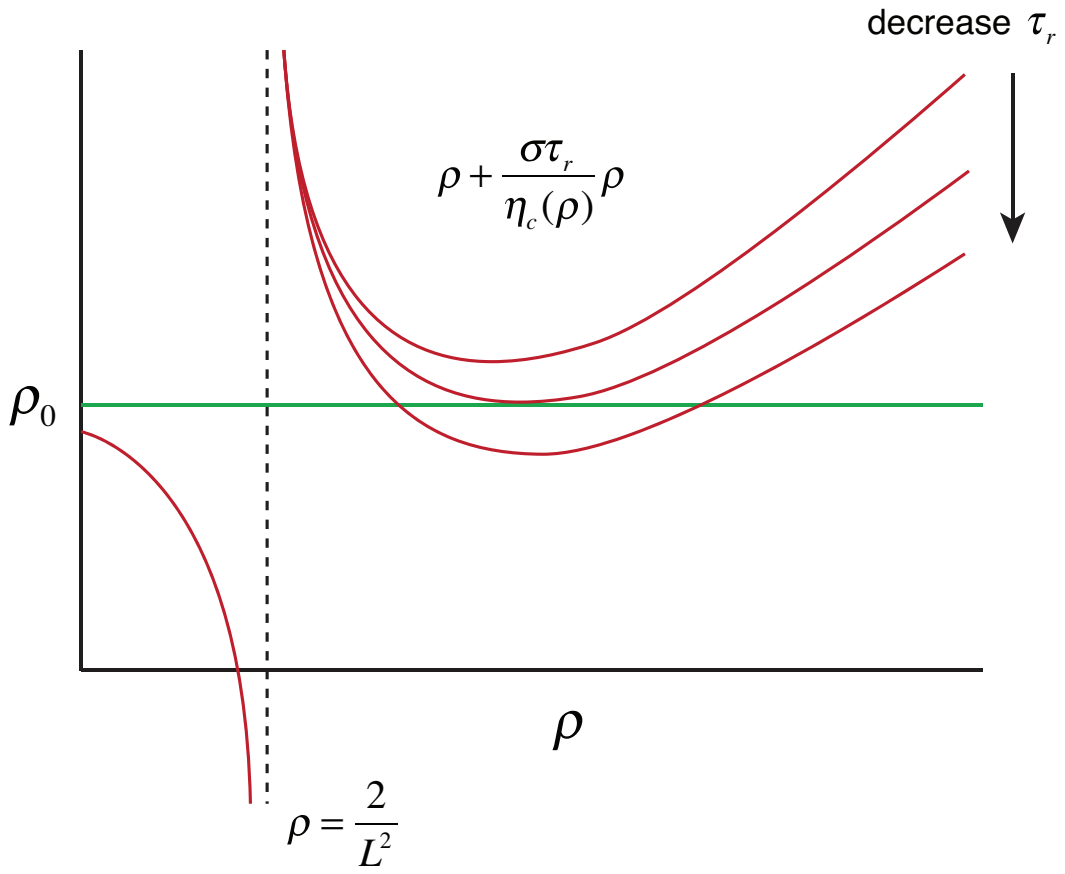


Figure 5.9: Flux balance analysis of network density. Qualitative plots of $\rho + \frac{\sigma\tau_r}{\eta_c(\rho)}\rho$ (red curves) vs ρ_0 (green line) for different values of τ_r. For sufficiently large τ_r, there are no crossings. For $\tau_r < \tau_{crit}$, there are two crossings: The rightmost crossing represents a stable steady state.

Figure 5.9 sketches the positive (ρ_0) and negative ($\rho + \frac{\sigma\tau_r}{\eta_c(\rho)}\rho$) contributions to the right hand side of Equation 6 for different values of τ_r. For sufficiently large τ_r, there is no stable state, i.e. strain thinning will occur. However, as τ_r decreases below a critical value τ_{crit}, a stable steady state appears. Note that when $\tau_r = \tau_{crit}$, $\rho + \frac{\sigma\tau_r}{\eta_c(\rho)}\rho$ passes through a minimum value ρ_0 at $\rho = \rho^*$. Accordingly, to determine τ_{crit}, we solve:

$$0 = \frac{d}{d\rho}\left(\rho + \frac{\sigma\tau_r}{\eta_c(\rho)}\rho\right) = 1 - \frac{\sigma\tau_r}{\xi(\rho L^2/2 - 1)^3} \tag{5.14}$$

From this, with some algebra, we infer that

$$\rho^* = \frac{2}{L^2}\left(1 + \left(\frac{\sigma\tau_r}{\xi}\right)^{1/3}\right) \tag{5.15}$$

and

$$\frac{\sigma\tau_r}{\eta_c(\rho^*)} = \left(\frac{\sigma\tau_r}{\xi}\right)^{1/3} \tag{5.16}$$

We seek a value for $\tau_r = \tau_{crit}$ at which

$$\rho^* + \frac{\sigma\tau_{crit}}{\eta_c(\rho^*)}\rho^* = \rho_0 \tag{5.17}$$

Substituting from above, and using $\rho_0 = \frac{2}{Ll_c}$, we have:

$$\frac{2}{L^2}\left(1 + \left(\frac{\sigma\tau_{crit}}{\xi}\right)^{1/3}\right)\left(1 + \left(\frac{\sigma\tau_{crit}}{\xi}\right)^{1/3}\right) = \frac{2}{Ll_c} \tag{5.18}$$

Finally, rearranging terms, we obtain

$$\tau_{crit} = \frac{\xi}{\sigma}\left(\sqrt{\frac{L}{l_c}} - 1\right)^3 \tag{5.19}$$

Table 5.1: Simulation Parameter Values

Parameter	**Figure 3**	**Figure 4**	**Figure S2a,b**	**Figure S2c,d**	**Figure 7**	**Figure 9**
L	$1, 3, 5, 7, 10$	3	$3, 5$	$3, 5$	5	$3, 5, 8$
l_c	$0.2, 0.3, 0.5, 0.8$	$0.3, 0.5$	0.3	$0.15, 0.2, 0.3, 0.4$	$0.2, 0.3$	$0.15, 0.2, 0.3, 0.4$
μ_e/μ_c	100	100	$3 - 300$	100	100	100
μ_c	0.01	0.01	$0.01 - 0.3$	$0.001 - 0.03$	0.01	0.01
ξ	$0.1, 1$	$0.05, 0.1, 1$	$0.01, 0.1, 1$	$0.1, 1$	$0.1, 1, 3.3$	$0.1, 1$
υ			$0.1, 0.3, 1$	$0.1, 1$	$0.1, 1, 3$	0.1
ϕ			0.25	0.5	$0.25, 0.75$	0.25
τ_r		$0.1 - 10^4$			$0.01 - 10^3$	$0.01 - 10^3$
σ	$0.0002 - 0.01$	$0.00003 - 0.005$				

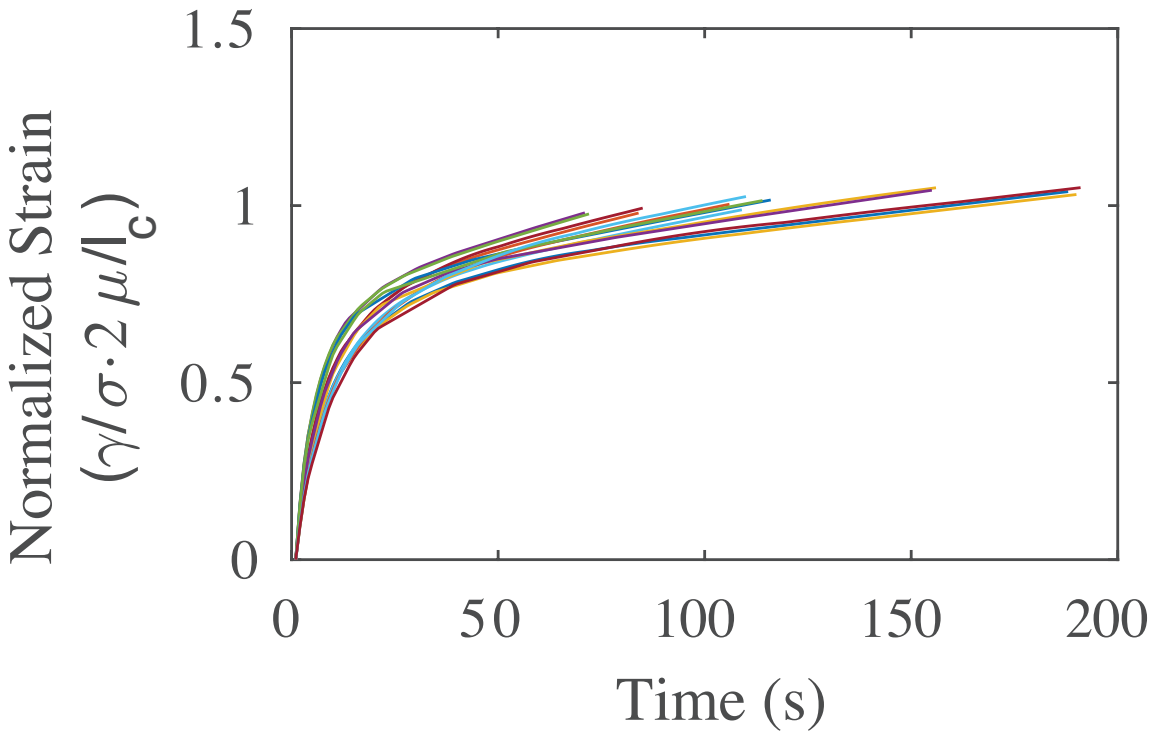


Figure 5.10: Fast viscoelastic response to extensional stress. Plots of normalized strain vs time during the elastic phase of deformation in passive networks under extensional stress. Measured strain is normalized by the equilibrium strain predicted for a network of elastic filaments without crosslink slip $\gamma_{eq} = \sigma/G_0 = \sigma/(2\mu/l_c)$.

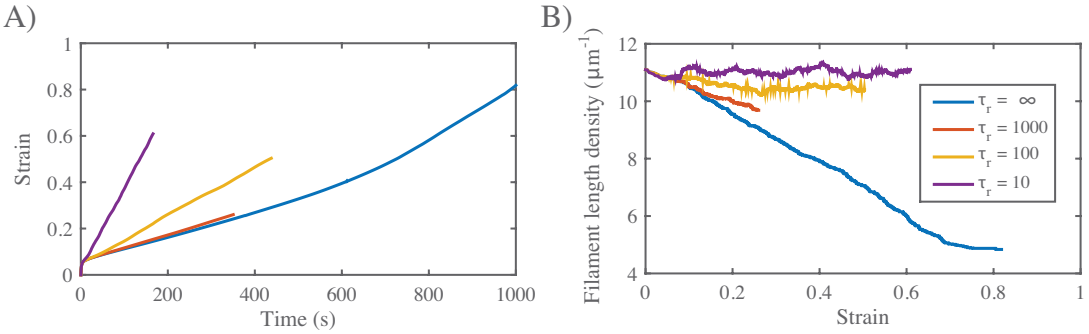


Figure 5.11: Filament turnover rescues strain thinning. **a)** Plots of strain vs time for different turnover times (see inset in (b)). Note the increase in strain rates with decreasing turnover time. **b)** Plots of filament density vs strain for different turnover times τ_r. For intermediate τ_r, simulations predict progressive strain thinning, but at a lower rate than in the complete absence of recycling. For higher τ_r, densities approach steady state values at longer times.

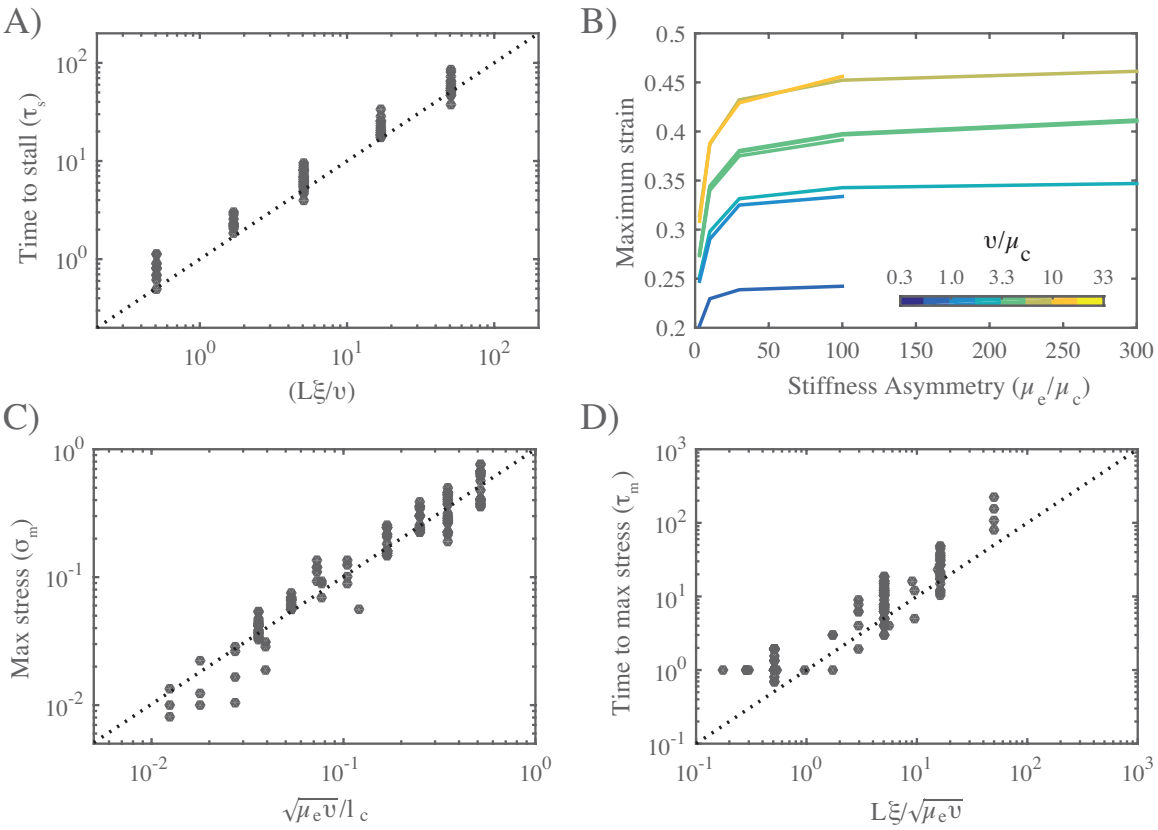


Figure 5.12: Mechanical properties of active networks. **a)** Time for freely contracting networks to reach maximum strain, τ_s, scales with $L\xi/\upsilon$. **b)** Free contraction requires asymmetric filament compliance, and total network strain increases with the applied myosin force υ. Note that the maximum contraction approaches an asymptotic limit as the stiffness asymmetry approaches a ratio of ~ 100. **c)** Maximum stress achieved during isometric contraction, σ_m, scales approximately with $\sqrt{\mu_e \upsilon}/l_c$. **d)** Time to reach max stress during isometric contraction scales approximately with $L\xi/\sqrt{\mu_e \upsilon}$. Scalings for τ_s, σ_m and τ_m were determined empirically by trial and error, guided by dimensional analysis.

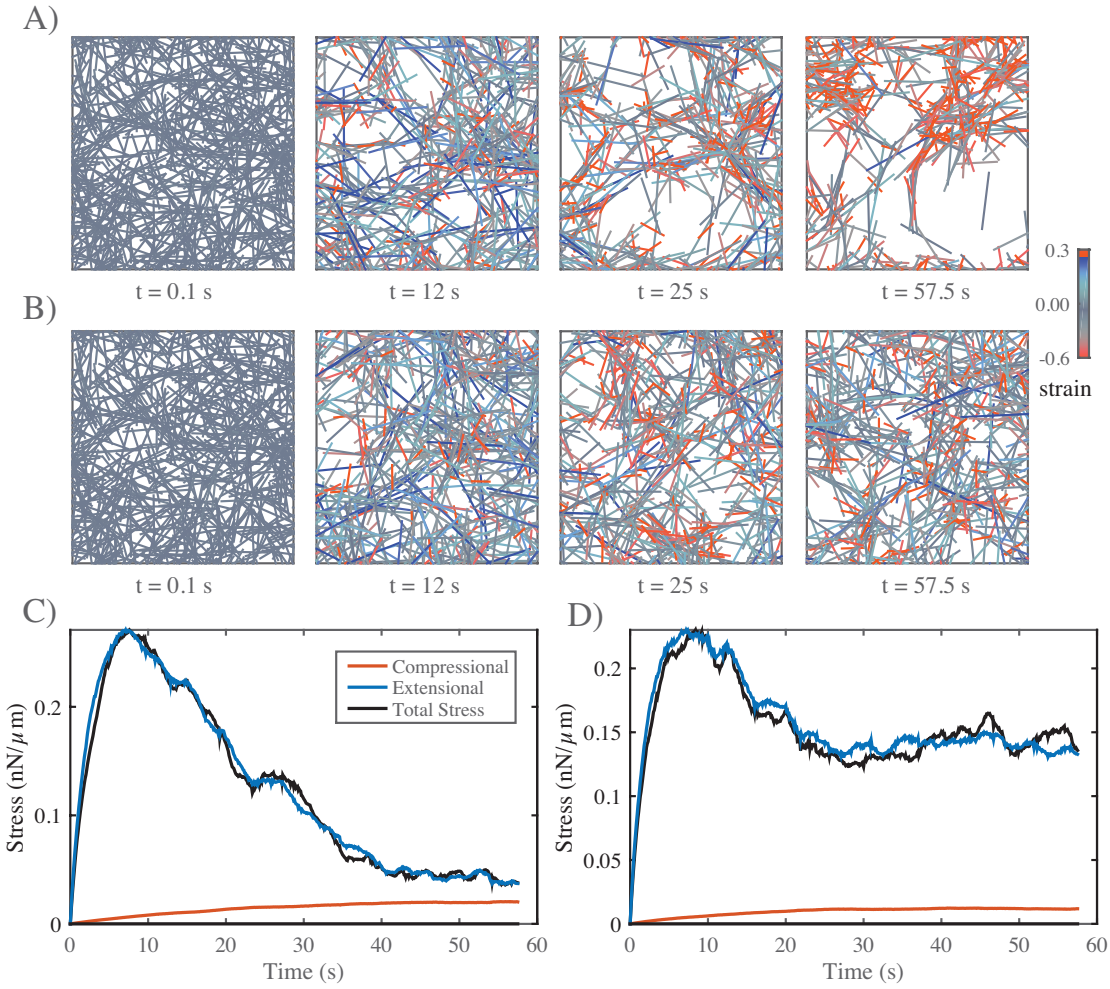


Figure 5.13: Filament turnover prevents tearing of active networks. **a)** An active network undergoing large scale deformations due to active filament rearrangements. **b)** The same network as in (a) but with a shorter filament turnover time. **c)** Plots of internal stress vs time for the network in (a). **d)** Plots of internal stress vs time for the network in (b).

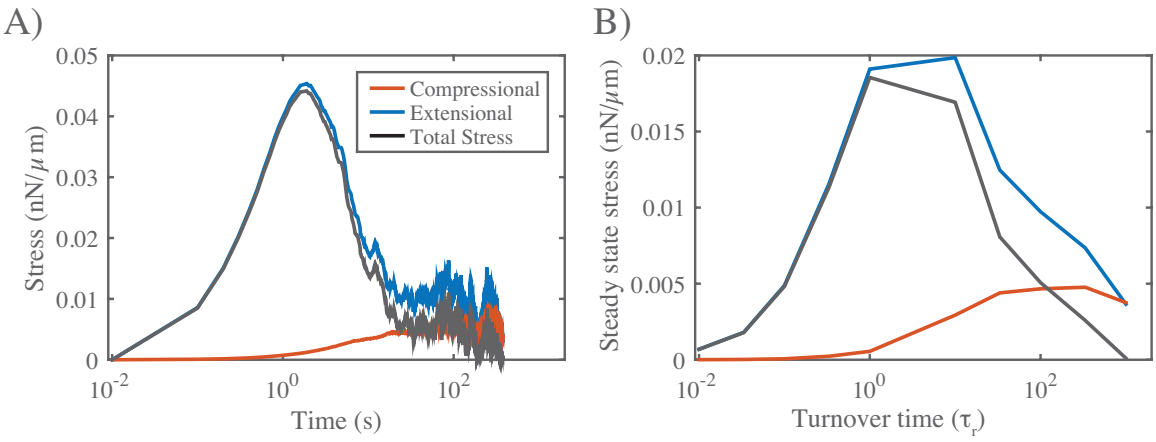


Figure 5.14: Bimodal dependence on turnover time matches bimodal buildup and dissipation of stress in the absence of turnover. **a)** Bimodal buildup of stress in a network with very slow turnover ($\tau_r = 1000s$). **b)** Steady state stress for networks with same parameters as in (a), but for a range of filament turnover times.

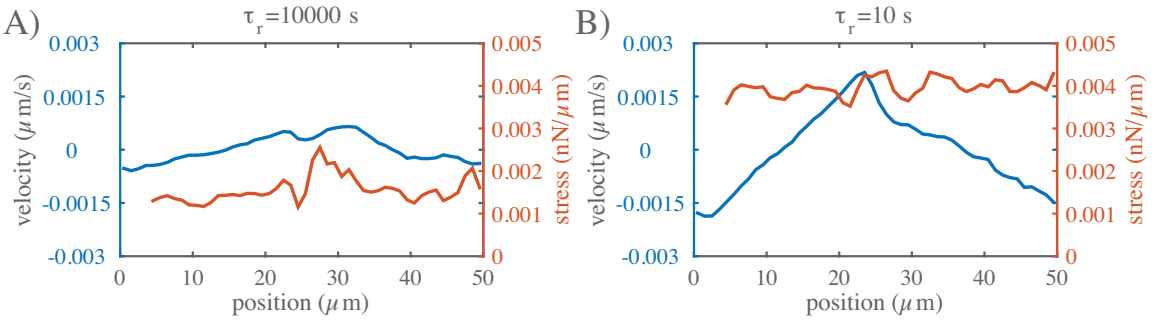


Figure 5.15: Dynamics of steady state flow. Plots of stress and strain vs position for networks in which motor activity is limited to the right-half domain and filament turnover time is either **a)** $\tau_r = 10000$ or **b)** $\tau_r = 10s$. Blue indicates velocity while orange represents total stress, measured as described in the main text.

CHAPTER 6
A MODEL OF UPSTREAM ACTOMYOSIN REGULATORS IN PULSED CONTRACTIONS

6.1 Motivation and experimental context

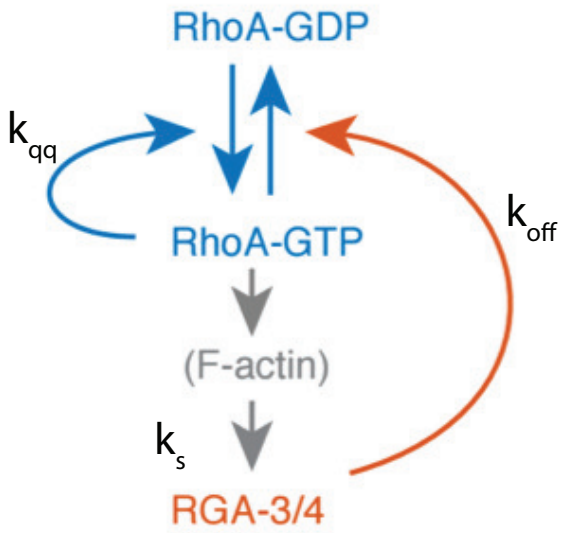


Figure 6.1: Reaction pathway with labels for coefficients associated with feedback stregths.

In many systems, cortical flows are driven not by continuous contraction of active material, but by repeated rounds of pulsatile contraction. While it was once believed that these pulsatile behaviors could be an emergent property of actomyosin contractility itself, it is now more probably suspected that

It is still unclear why so many systems exhibit this type of behavior, it is nonetheless important to understand the origins of these behaviors and what differences may arise due to their presence or absence in a contractile system. Therefore, we have

begun exploring how to model both the upstream regulators that govern contractile systems as well as the downstream effects of coupling these to regulators to contractile flows themselves.

The majority of our data comes from the work of Francois Robin and Jon Michaux (cite). They have shown that upstream of both actin and myosin is a separate pulsatile biochemical circuit consisting of a combination of positive and negative feedback between a pair of proteins called Rho and RGA as depicted in Figure 6.1. For more information on the biochemistry of this circuit please see their paper (cite again).

To explain the dynamics we attempted to build ordinary differential equation models of local variation in active Rho and RGA concentrations. We found that many models are effectively equivalent at producing the qualitative results. However, this model in all of its forms is inconsistent with producing robust pulsatile behavior once the models were constrained by parameter fitting to the data.

6.2 A model for pulsatile actomyosin accumulation in *C. elegans*

My first attempt at modeling involved a fair

I defined ρ as the concentration of Rho and r as the concentration of RGA, and generated association, dissociation and feedback parameters for the model.

$$\frac{d\rho}{dt} = k_{on}^{\rho}\left(1 + k_{on}^{\rho\rho}\frac{\rho^n}{\rho_0^n + \rho^n}\right) - (k_{off}^{\rho} + k_{off}^{\rho r}r)p \tag{6.1}$$

$$\frac{dr}{dt} = k_{on}^{r\rho}\rho - k_{off}^{r}r \tag{6.2}$$

Next we can nondimensionalize the equation with $q = \rho/\rho_0$, $s = k_{off}^{\rho r}/k_{off}^{r}r$, and $\tau = k_{off}^{r}t$, and rename parameters for simplicity.

$$\frac{dq}{d\tau} = k_q\left(1 + k_{qq}\frac{q^n}{1 + q^n}\right) - (k_{off} + s)q \tag{6.3}$$

$$\frac{ds}{d\tau} = k_s q - s \tag{6.4}$$

6.2.1 Analyzing parameter space of the model

Just from analyzing the null-clines of this model we can gather a lot of information about the qualitative behavior of this model.

$$s_q = \frac{k_q}{q}\left(1 + k_{qq}\frac{q^n}{1+q^n}\right) - k_{off} \tag{6.5}$$

$$s_s = k_s q \tag{6.6}$$

The s_q null-cline appears as an inverse relation between s and q, with a hump around $q = 1$ whose height is dictated by the strength of the q positive feedback, k_{qq}. The s_s null-cline is simply a straight line whose slope is governed by k_s. The dynamical system is only able to attain pulsatile behavior when the s_s null-cline runs parallel and to the left of the hump in the s_q null-cline. This gives two effective conditions on pulsatility: 1) If k_{qq} isn't much larger than 1, there will not be any hump, and so there cannot be any pulsatility. 2) If k_s and k_off

To test this prediction, I implemented an automated search of the model's parameter space. The test worked by adding perturbations of fixed strength to the equilibrium state, and observing whether the response of the system exceeded the perturbation. As shown in Figure 6.2 for a number of simulation parameters, the response function could vary from having just a stable fixed point ($k_{off} = 10$ and $k_s = 25$) to having stable oscillations ($k_{off} = 10$ and $k_s = 63$) to having pulses ($k_{off} = 20$ and $k_s = 63$). Figure 6.2 also shows the null-clines for each case, which can be used to interpret whether pulsation is possible.

Using this automated system, I was able to generate 1600 simulations and classify them automatically based on the ratio of the magnitude of the perturbation to maximum response (to determine if positive feedback drove excitation) and whether the maximum was attained multiple times (to differentiate pulses and stable oscillations).

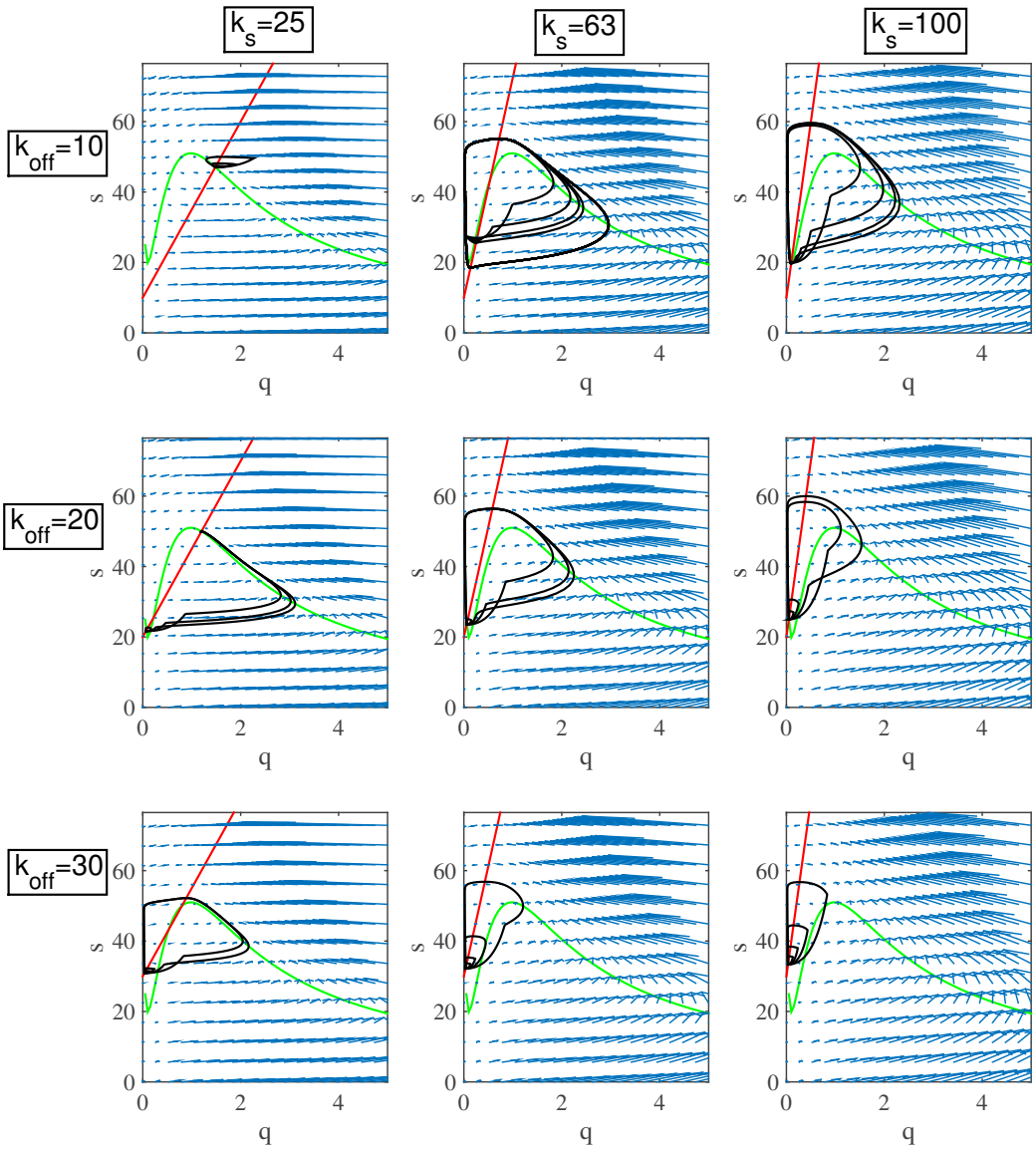


Figure 6.2: Perturbation simulation results plotted on phase planes for a variety of parameters.

This resulted in a phase diagram of the behavior as shown in Figure 6.3, with the three colors indicating the behavior of dynamical systems in that regime. The size and shape of the pulsing region of phase space was a direct result of the strength of positive feedback (k_{qq}) with stronger feedback allowing more of phase space to permit

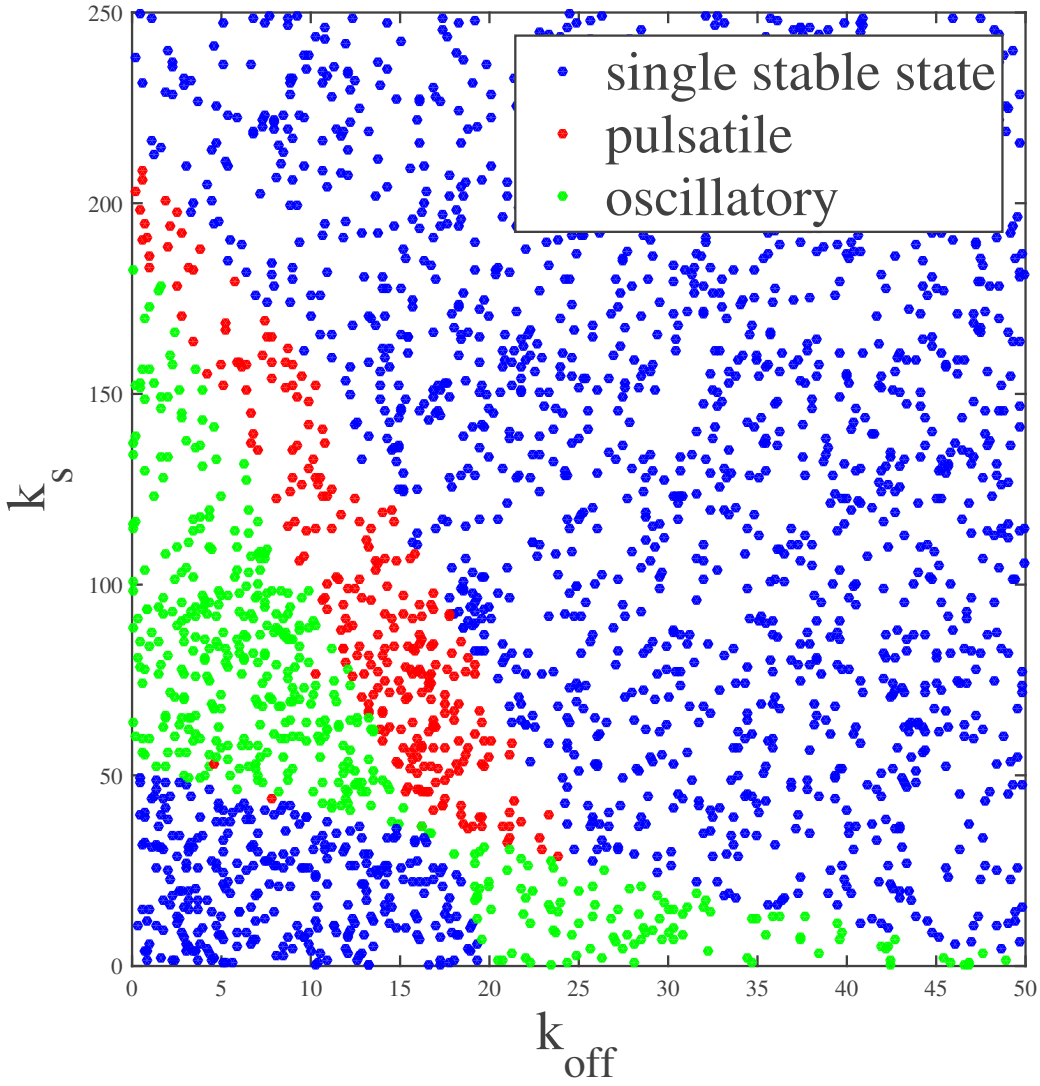


Figure 6.3: Phase diagram of for $k_q = 1$ and $k_q q = 100$.

pulsing and oscillations.

Finally, using this understanding of the phases behavior of the system, I implemented a stochastic dynamical equation to test the response to noise. As indicated in Figure 6.4, a basal level of noise was able to trigger robust pulses in both Rho and RGA for certain parameters. Taken together this analysis indicated that our biochemical reaction circuit for Rho and RGA feedback could in principle account for

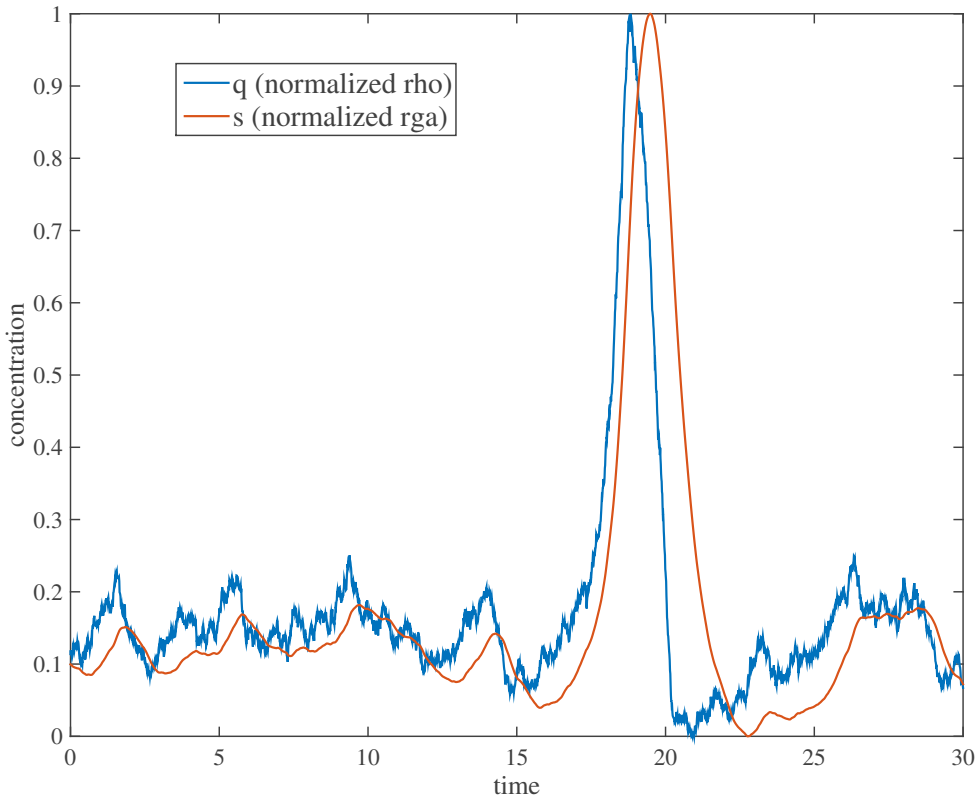


Figure 6.4

the excitable dynamics exhibited in the system.

6.3 Simplified model and data fitting

6.3.1 Fitting techniques

Although this model accounted for the qualitative behavior, we wished to determine whether it could be corroborated by fitting to the data presented in the paper. To experiment with fitting the normalized data, I created a reduced model, where the equilibrium concentration had been renormalized to 0 for both Rho and RGA, and the feedbacks between Rho and RGA are allowed to be non-linear.

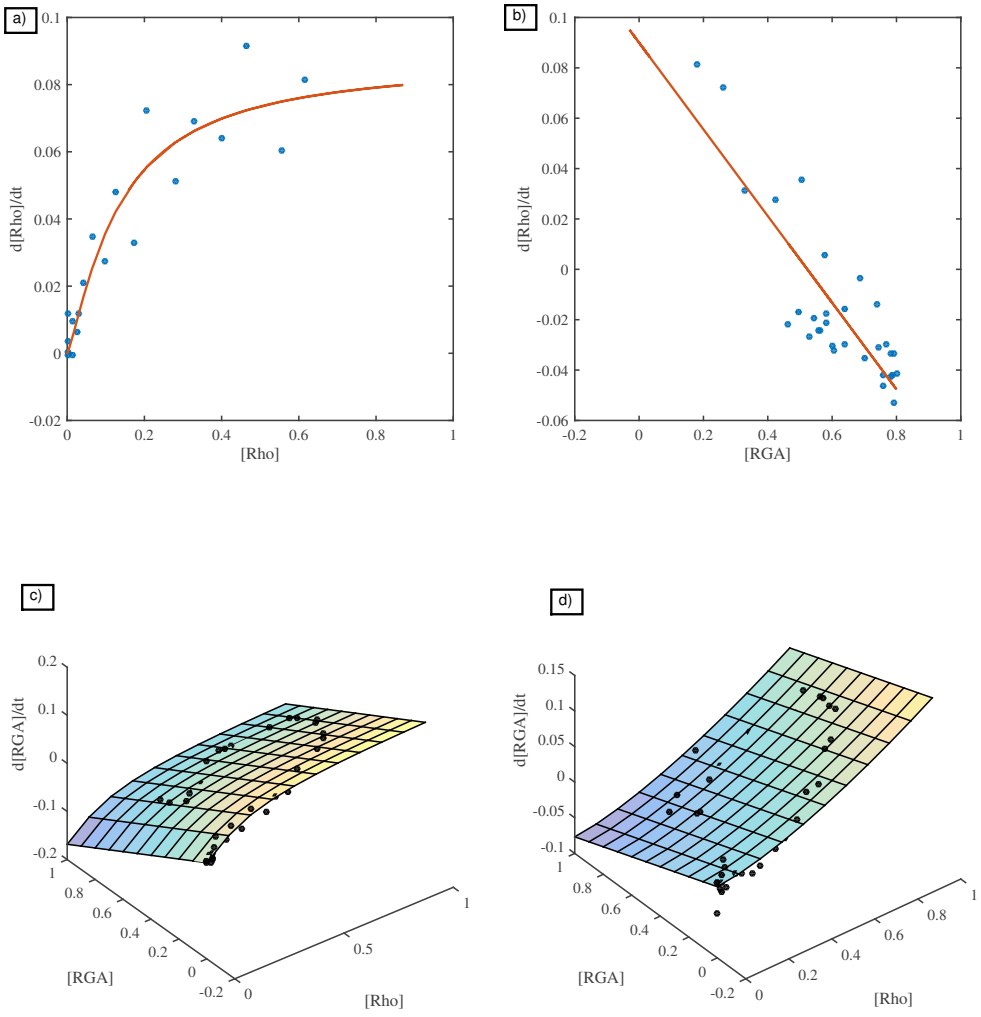


Figure 6.5: Multiple methods of fitting.

$$\frac{dq}{d\tau} = \alpha \frac{q^n}{1+q^n} - sq^k \tag{6.7}$$

$$\frac{ds}{d\tau} = \beta q^{(m-k)} - s \tag{6.8}$$

These equations resulted in a family of models with different levels of feedback,

depending on the values of n, m, and k. I fit models with a variety of exponents and found that any model with $k \approx 0$, $n > 1$, and $m \approx 2$ gave the best overall agreement with the greatest robustness. I used two different methods to fit the Rho and

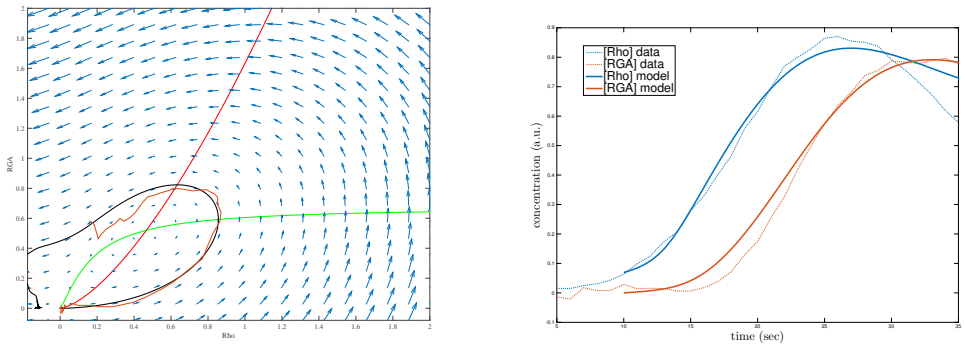


Figure 6.6: Simulation results and fitted data for model with $m = 2$ and $k = 0$. Displaying simulation results and the data used to fit the simulation on a phase plane (left) and as pairs of time plots (right).

RGA data to determine the most appropriate model parameters. The first required subsecting the data to fit only datapoints where some terms in the equation were presumably close to constant (Figure 6.5a,b). For example, as long as RGA concentration remained relatively small, the equation for Rho could be taken to consist of only the Rho self-feedback function. The second method allowed fitting all of the data the equations at once (Figure 6.5c,d). The second method is clearly more accurate, but it comes at the cost of being slightly more difficult to explain to the lay-biologist.

The resulting fitted models could then be used to run simulations. As shown in Figure 6.6, the models and fitted data were largely indistinguishable relative to the error in the data. Although the model was quite successful at recapitulating the original data, I found that the resulting models were not necessarily. For example by changing the value of the parameter α by 25% and rerunning the fits, I could tune the system from a stable system to pulsatility and into an oscillatory regime (Figure 6.7). Therefore, it appears that this model is not incredibly robust to minor variations in parameters.

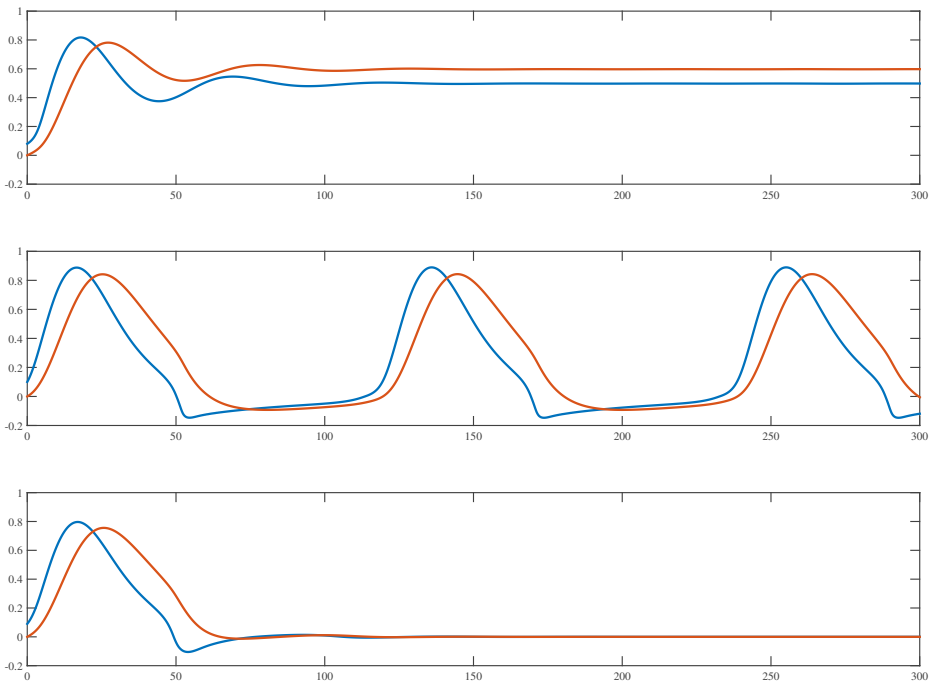


Figure 6.7: Small variation in simulation parameters can have large qualitative effects.

6.4 Conclusion

Ultimately, a model similar to 6.1 and 6.2 was utilized in the final paper in combination with at the 2D fitting routine outlined in Figure 6.5c,d. In short, all models performed fairlt well at describing the qualitative features of the data, but none were very effective at generating a quantitative match with great robustness. We believe this is indicative of the need to perhaps extend to incorporating spatial effects in driving the dynamics of active material pulses. Based on the theoretical results of [bois and the other one], we attempted to incorporate our upstream regulatory model into a 1D active fluid. This ongoing work will hopefully tie together the spatial and temporal interdependencies driving this system into its interesting non-equilibrium state.

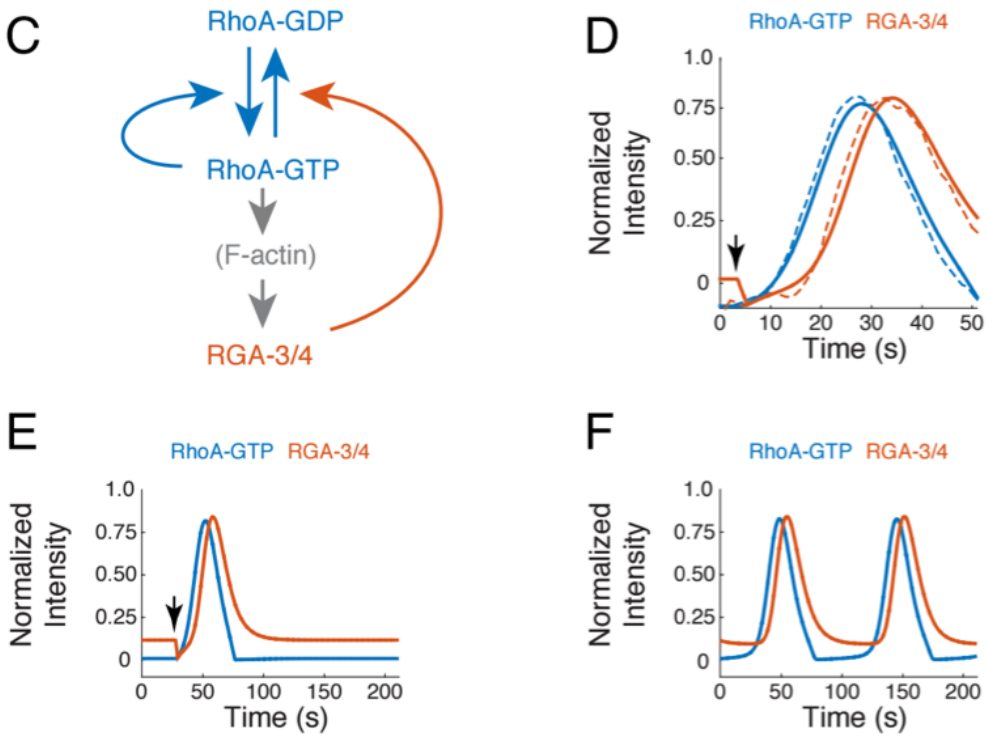


Figure 6.8: Final figure as it appears in Robin et al.

CHAPTER 7
OPEN ISSUES & FUTURE DIRECTIONS

7.1 Incorporating multi-segment filaments and bending degrees of freedom

For simplicity, notable aspects of semi-polymer mechanics have been ignored throughout the entirety of this thesis. In developing this work, I chose to limit my analysis to single springlike filaments in order to focus attention on the most prominent properties of semi-flexible polymers. In effect, I took the minimal number of model elements (and accompanying free parameters) that would suffice to produce the 2D network flows of interest.

While this choice greatly simplified the analyses performed and allowed me to focus my results, it does ignore aspects of filament mechanics that may play an observable role in macroscopic cell mechanics. In particular, there are two clear oversimplifications that are introduced by using single springs: uniform strain along filaments and absence of bending.

Uniform strain along filaments results from the model's current insistence that forces add to produce a net strain in the filament. Because all forces are transmitted merely to the ends of each spring, there can be no internal regions of variable strain anywhere else along the filament. This necessarily overlooks the local deformations that could be driven by internal motor forces. The net result will be that deformations on small length scales. As such, certain measurements that were made in the above analysis are most probably over-averaged and not indicative of what would be found in a real system. It is still unclear what impact this will have on the macroscopic dynamics of the system, but nevertheless, this is one of the largest missing pieces that could invalidate some of the model predictions.

The absence of bending degrees of freedom is probably of less concern than the

imposition of uniform filament strain. To begin, the fact that bending causes filament stiffness asymmetries has already been incorporated in the asymmetric extensional stiffness imposed on filaments in this model. Thus, adding bending will only serve to double count this asymmetry and will probably not provide much benefit. A second aspect of filament network mechanics is more problematic. It has been shown previously that the mechanical picture of 2D networks can transition from extension dominated to bending dominated when network densities are sufficiently sparse. The net result of this is that at low enough densities, the main mechanical resistance will be dependent on filaments resisting bending. My model will neglect this transition to bending dominated mechanics, and therefore, the model network should become completely pliable despite the fact that the real network will still maintain some resistance to deformation. However, analyzing networks at such sparse densities will present numerous challenges to the continuum models presented in this work so worrying about the specifics of bending dominated mechanics may be of lesser concern.

Nevertheless, it is important to note that the current implementation can easily allow the introduction of multisegment bending elements. If one uses segment sizes that are shorter than the total filament length, joints will automatically be introduced that separate the filament into multiple regions that are free to deform on their own. However, with $\kappa = 0$, these joints will be free to rotate, which will cause the model to create effectively separated springs that are merely forced to share one attached end. However, $\kappa > 0$, the model will introduce a bending spring that tries to keep individual filaments straight. The magnitude of the bending modulus can then be varied to change the bending stiffness of the filament.

7.2 Probing more complex mechanisms of filament turnover

Another notable simplification in this work is the method of incorporating filament turnover by causing entire filaments to be reset at random. This is actually not the microscopic means by which filaments are depolymerized and repolymerized in the cell. In fact, both filament depolymerization and repolymerization are governed by more complex and intricate processes that have been studied to painstaking detail.

The net result of these events does cause rapid and complete strain resetting on long enough timescales, but there can be subtleties that change the exact form of the strain and orientation resetting of filaments.

The actual mechanism for filament depolymerization relies on a balance of slow filament treadmilling, accompanied by faster timescale filament severing events. The combined action of these two mechanisms should cause a quite rapid removal of the filament from the physically connected network and thus cause an effectively immediate stress dissipation much like the one demonstrated in our model. Nevertheless, there may be all kinds of regulatory factors which act to make the stress disspiation less idealized than in the simplified model. One such complicating factor would be stress dependent severing rates, which would cause non-uniform dissipation of stress or preservation of stress depending on whether filaments were preferentially severed based on being high stress or low stress state, respectively. Any mechanism could in principle be added to this model, however, it would add to the difficulty of interpreting results and therefore should be incorporated only if some aspect of the model is found to be deficient in its explanatory ability of a specific result.

In addition, to regulation causing non-uniformity in depolymerization, there are molecular details that impact our assumptions about repolymerization as well. In cells, much of the structural actin cytoskeleton is aided in polymerization by formins, which recruit actins to rapidly accelerate the polymerization process. While many aspects of the polymerization are still under active study there is at least some preliminary evidence that the specific nature of formin polymerization provides biases in the orientation of newly polymerized filaments. Specifically, formins appear to follow existing actin filaments preferentially, thereby serving to lay down new actin along a template of existing actin. In addition, crosslinking proteins can preferentially align newly polymerized actin as well. This may be an important part of regulating stress assymetries and as such may need to be incorporated into some aspects of active network models in the future.

The end result of these complex processes allows stress resetting to occur independently from orientational resetting. In this sense, the simplifications in the current implementation tend to conflate the processes of stress resetting and orientation re-

setting. In effect, there may be different timescales over which different aspects of network memory relax. As such this may be an important avenue of future work.

7.3 A novel cell squishing technique to measure timescales of relaxation *in vivo*

At the moment, there are very few possible experiments to probe the stress relaxation possible from filament turnover. By squishing the cell into a hot dog shape and then letting it freely relax to a sphere, one can approximate the viscosity of the cell's surface. This is because the timescale of relaxation to spherical for a purely viscous droplet embedded in a medium with much lower viscosity is $\tau \sim T/\eta$ where T is the surface tension and η is the droplet viscosity.

Some of my preliminary experiments showed that this technique was highly reproducible. In addition, by carrying out the experiments on days with different barometric pressures and in rooms with different temperatures, I was able to get a consistent shift in the relaxation timescale between sets of samples. Finally, in an experiment with Latrunculin A, a factor that largely depolymerizes the entire actin cytoskeleton, I could determine that the timescale of relaxation for cells in the absence of cortical structure was effectively instantaneous. Thus, changes in cortical viscosity should be easily assessed.

I made several attempts to perform the experiment myself, but, alas, my experimental chops were not up to the task. When treating with jasplakinolide to tabilie the cortex, the embryos underwent a global irreversible contraction. This is to be expected from our earlier observations where myosin acting on a network without turnover causes network contraction and tearing of the cortex. Therefore, in order to perform the experiments properly, one needs to first knock down myosin in the cortex. This presents some experimental difficulty, but can easily be overcome by anyone with sufficient training in *C. elegans* embryonic training.

APPENDIX A
ARTISTIC INTERPRETATIONS OF FILAMENT RECYCLING

As part of a collaborative *Arts, Science and Culture* Grant sponsored by the University of Chicago's Institute for Molecular Engineering, Divisions of the Biological and Physical Sciences, the Humanities, and the Office of the Vice President for Research and for National Laboratories, I undertook a .

A.1 The ship of Theseus as a metaphor for life

> The ship wherein Theseus and the youth of Athens returned from Crete had thirty oars, and was preserved by the Athenians down even to the time of Demetrius Phalereus, for they took away the old planks as they decayed, putting in new and stronger timber in their places, in so much that this ship became a standing example among the philosophers, for the logical question of things that grow; one side holding that the ship remained the same, and the other contending that it was not the same. —Plutarch's Life of Theseus, as translated by John Dryden

A.2 Experiments with plastic filament sculpture

Figure A.1

Figure A.2

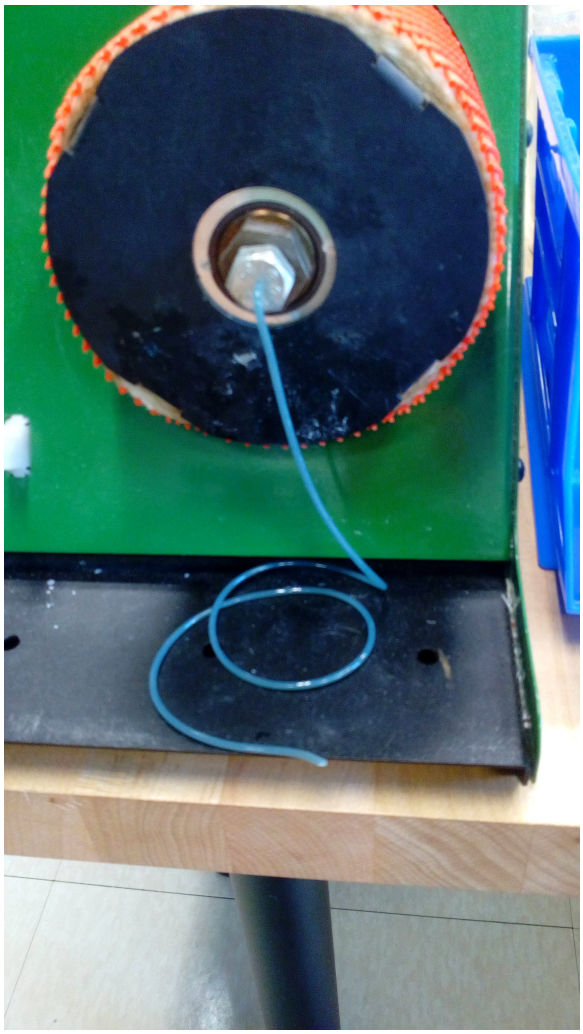


Figure A.3

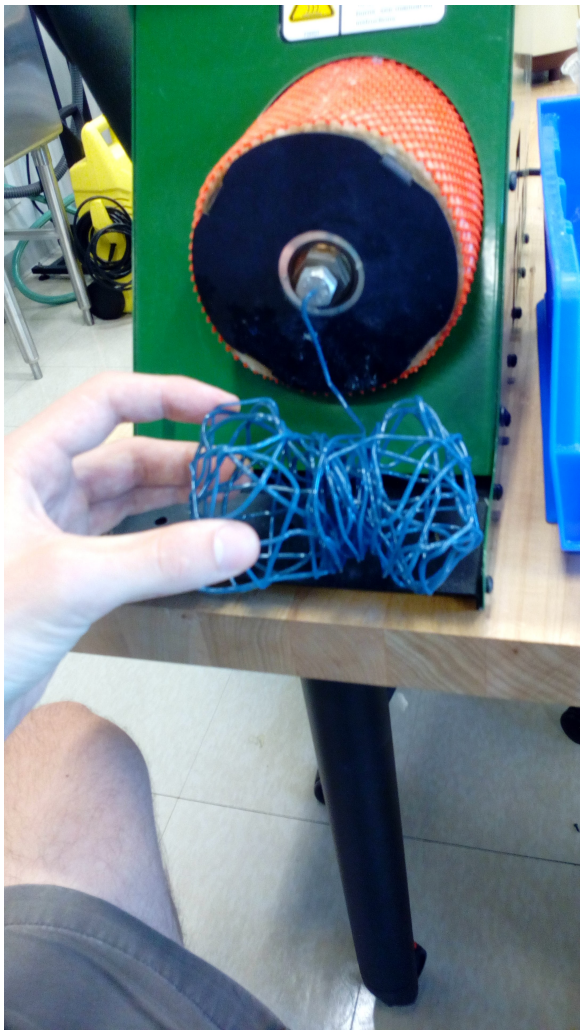


Figure A.4

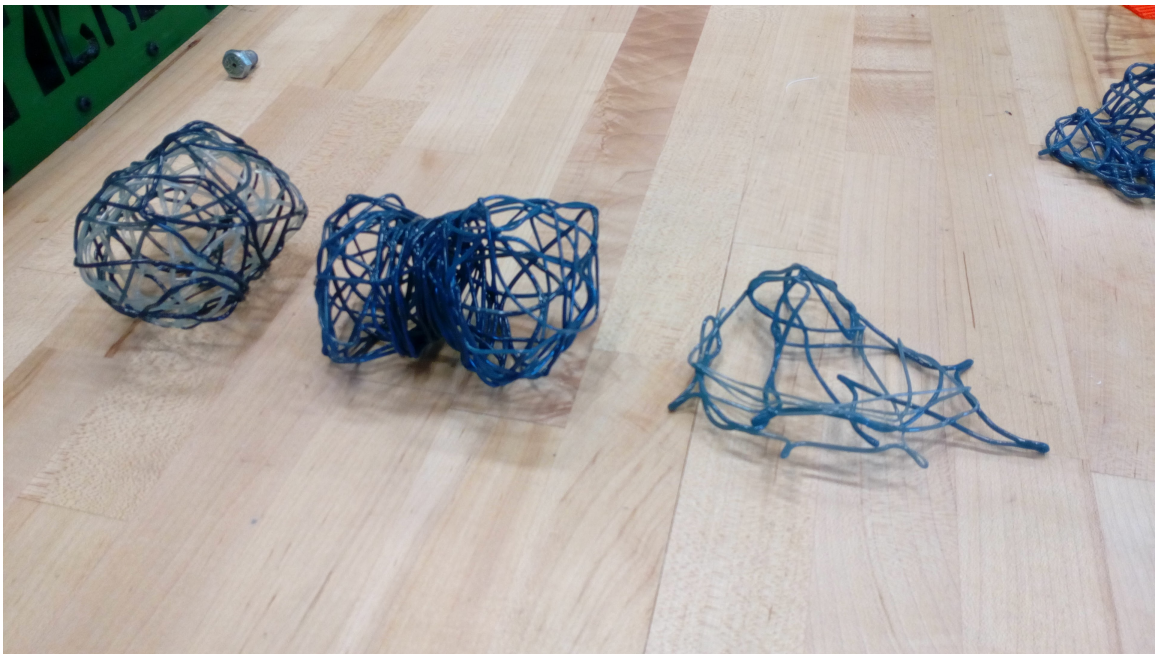


Figure A.5

Figure A.6

Figure A.7

Figure A.8

APPENDIX B
WORKSHOP ON MODELING IN BIOLOGY

B.1 Course syllabus

B.1.1 Course Objective

This course is designed to introduce a student with a reasonable understanding of biology to the basic techniques of mathematical modeling. Specifically, the student will be able to take a conceptual model of some biological system and transform it into an appropriate mathematical framework. The student will also be able to develop intuition based on their model and to interpret simulation results to draw conclusions.

B.1.2 Course Design

Content, Converse, Convey For each lesson in the course, the assignments will proceed in three parts. First, students will familiarize themselves with the content of that weeks lesson and take a short quiz to ensure that they have done the necessary reading. Second, the students will meet in class or use the online forum to converse on a problem related to the lesson. Finally, ever student will be given their own related problem to solve and convey their answer on our class wiki for other students to evaluate.

Programming Projects In addition to the weekly assignments, the students will also be synthesizing their knowledge of mathematical modeling concepts to build their own simple modeling projects. By the third week the students will be able to pick a biological system to study. They will work during a few of the hour long discussions to put together their own model of the system and explain it to the rest of the class. Finally, they will present an in-depth description of their project on the class wiki.

B.1.3 Assignments

Readings and Quizzes There are readings and lectures online for you to become introduced to the material before you come to class. There will be online quizzes on Chalk which will be due by Tuesday at midnight the day before new material is covered in class. These quizzes should resemble simple multiple choice exam questions; they

merely check whether you have viewed the material. However, in total these quizzes count towards 25your overall grade so take them seriously.

Class Forum Discussions Every week, the class will meet together to tackle a problem related to the lesson. The problems are frequently posed based on questions left unanswered by studentsin the previous year. The class should come to a reasonable consensus by the end of the session or they should follow up online to come to an agreement. Ive designed the discussion sections so that you dont need to be the most boisterous student to participate in the discussion meaningfully. There will be an equal weight to valuable comments given in person as well as those on our online forum. In addition, our class discussions will often contain smaller discussions that allow for one-on-one interaction. Nevertheless, class discussion accounts for 25% of your grade so, whether in the forums or in person, your participation is required.

Wiki Articles Each student will be given a more in-depth problem for each lesson to be written up and posted to the group wiki. Here we are not just looking for a solution to the problem. We need an explanation of your approach and why it is right. Your peers will be evaluating your work to make sure that it makes sense to them. The writing and evaluation of these assignments will count for another 25% of your grade.

Final Project By the third week of this course, the student should start to have an understanding of what types of systems can be modeled with a differential equation. At this point in time, the students will select a biological system to model as their final project. The class during 4th week will involve every student briefly stating their problem area how they will model it. Please note that the system you are modeling does not have to be extremely complex, and the model you generate does not have to be perfectly accurate. The goal of this project is simply to show that you understand all of the steps in moving from an abstract idea to a mathematical model. By the end of the session you should have a reasonably involved computational model.

B.2 Post-class Student Survey

I asked the students to give me some feedback on their experience in the course and what could be improved. Feedback was generally positive and instructive in looking for ways to improve.

	A lot	A little	Not at all	Did more harm than good
This class clarified what biological models are good for.	100.00% 2	0.00% 0	0.00% 0	0.00% 0
The class examples were at the right pace for my level of understanding.	50.00% 1	50.00% 1	0.00% 0	0.00% 0
The information presented was well-motivated.	100.00% 2	0.00% 0	0.00% 0	0.00% 0
Expectations were clearly stated.	0.00% 0	100.00% 2	0.00% 0	0.00% 0
Class discussions were useful.	50.00% 1	50.00% 1	0.00% 0	0.00% 0

Figure B.1: Responses from student survey. Only 2 of 5 students replied.

What are some questions you have about biological modeling that this class should have addressed. I think it might be good to emphasize presentation and interpretation of modeling results, both from the perspective of the experimenter and the reader. In other words, I think it would be good for people to become comfortable both reading theoretical papers and generating simple modeling figures. I know this wasn't the focus of the workshop, but it might be nice to briefly cover agent-based modeling to the point where someone could be conformable interpreting experiments.

It was a good foundation. Maybe how to interpret a model in a paper and evaluate it, if we did an example in class that would be helpful.

Describe how you would like to see this workshop structured to best promote your learning of the material It might be a good idea to have a list of things to model, so that people can still progress through the workshop even if the system they work on isn't amenable to modeling or isn't interesting to model. I liked the emphasis on students giving short presentations throughout.

A little more background on the basic calculus we needed. Also talking more about the reasoning behind what you're doing and less algebra on the board, it's kinda hard to follow but we can all probably do it.

B.3 Reflections and Future Ideas

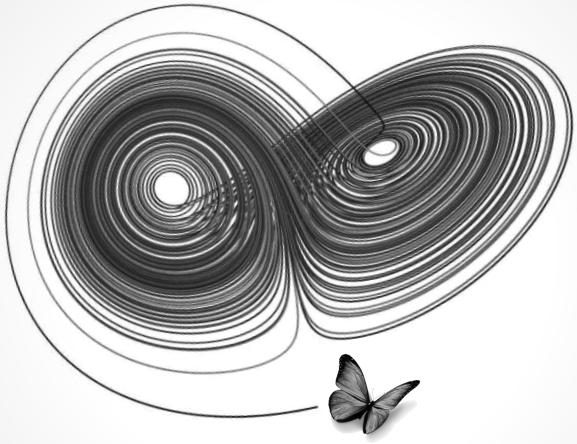
In general, the students reacted positively to the framework that we established with it's emphasis on the three C's outlined above. The student survey responses showed that the focus on classroom participation was welcome. In practice, the students were not required to complete any homework, and the course was not given for a grade. This led to many of the assignments being neglected entirely. However, despite all of the work being completely optional, many of the students completed a significant portion of the work voluntarily.

In the future, I will need to do a bit more preparation work to select a larger corpus of example systems for students to work with. And having more examples to work through together will also probably improve outcomes. On the other hand, I don't want to lose the student driven focus that I was hoping to attain. I may also want to establish a bit more authority so that students attend class with their assignments completed.

B.4 Course Materials

B.4.1 How mathematical models make sense of complex processes

Biology and Dynamical Systems



How mathematical models make sense of complex biological processes.

Class Structure: This workshop is broken up to focus on each of ”the three Cs”

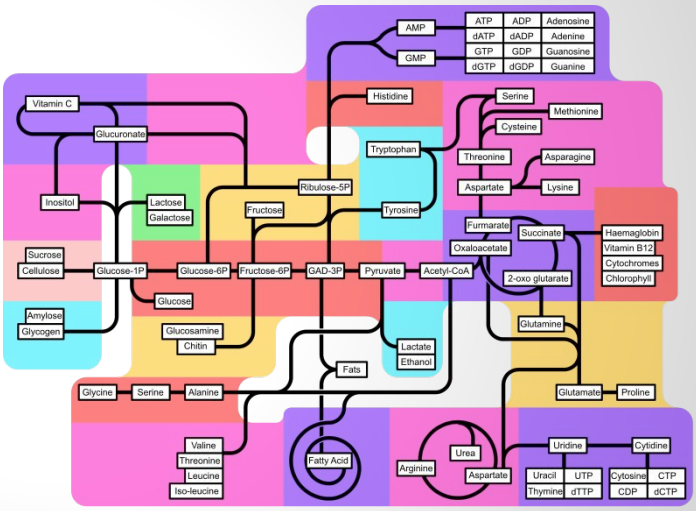
Collect: Part of what we are doing here is learning how to collect information. Every class starts with a simple content quiz over the short reading, followed by a 15 minute review of the day’s focus content.

Collaborate: The majority of the class time will focus on an in depth conversation between peers while solving a sample problem. The exercises are evaluated for their persuasiveness, teamwork, and communication.

Convey: The final portion of each class will involve an individual project assignment. This will involve both developing an appropriate answer to a question as well as clearly explaining the logical process.

Why is mathematical modeling important to biologists?

- Biological systems can be highly complex
- We explain complex systems with cartoon models of reaction pathways
- Most pathway models are built up by finding many pairwise interactions
- Predicting complex behaviors is only possible through quantitative modeling

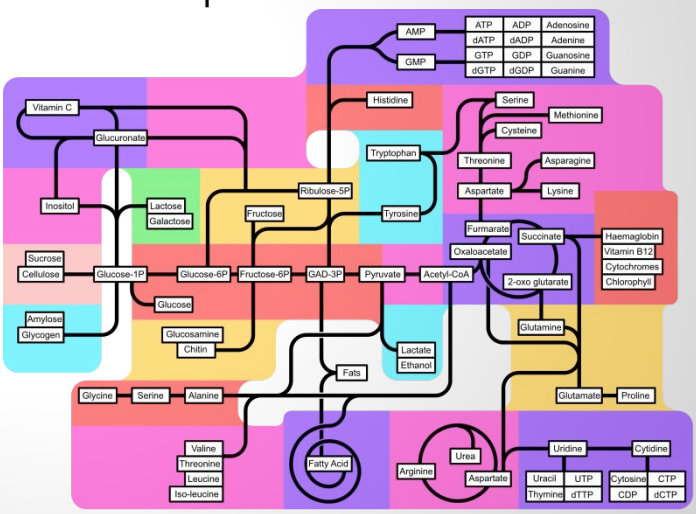


Biology is hard enough on its own. Why study mathematical modeling too?

1. Modeling isn't as hard as you think
2. Math models make predictions that cartoon models can't
3. Models clarify the key components of a system
4. Math skills make you more marketable after graduation

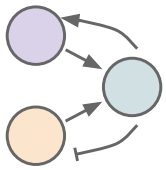
3. Models can clarify key components of reaction pathways

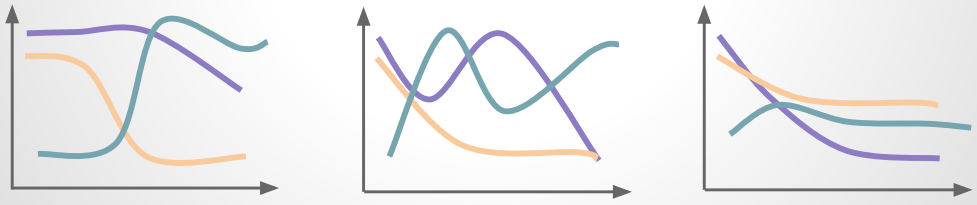
Wouldn't you like to tease apart a parasite's core metabolic pathway to find drug targets using just a system of differential equations?



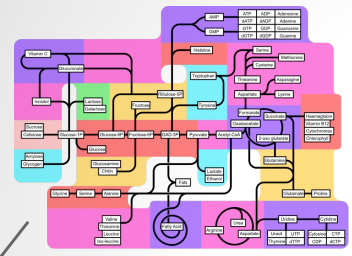
2. Mathematical models make predictions that cartoon models can't

Cartoons really aren't specific enough for surprisingly simple questions.

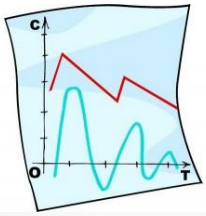




1. Mathematical models aren't that hard



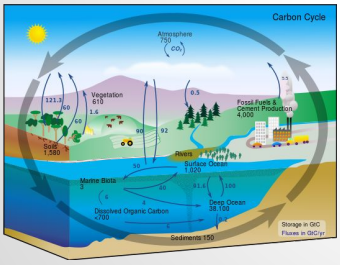
$$\frac{dc}{dt} = k \cdot a$$



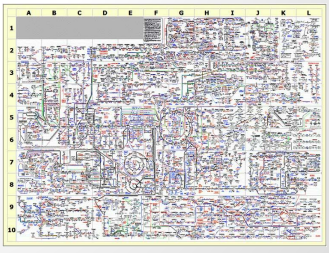
The balancing act behind scientific models

Complex Problem

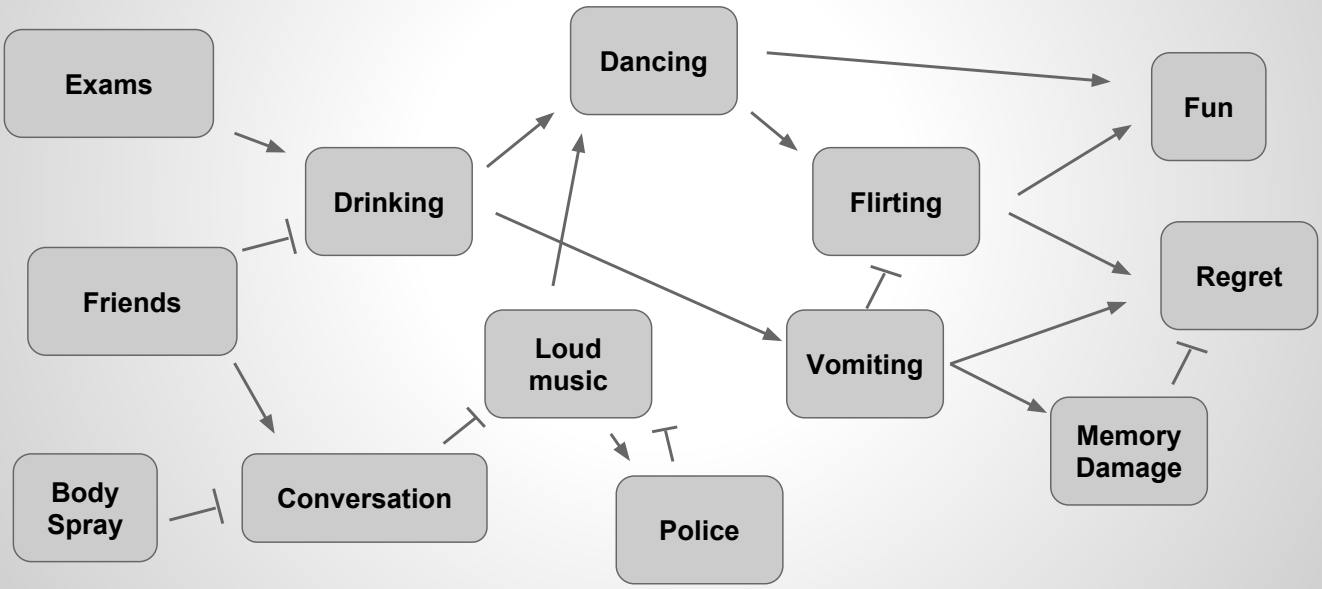
Simplification



Completeness



Scientific Model of Anything: party



Class activity: Build a model of ANYTHING

- Break up into groups (starting NOW)
- You'll have 8 minutes to figure out a model of anything you want
- Brainstorm a system quickly. You'll need to pick one after 1 minute.
- Define the most important things that happen in your system
  - Remember the party example
- Diagram how each important thing relates to each other
  - Use arrows for *upregulation* and crossbars for *downregulation*
- Clearly draw your model diagram for 5 points (put your names on it!)
- Groups can volunteer to present their model for 5 extra points
  - Each group gets 1 minute to present their system and how they are modeling it
  - We'll vote on the winners in 4 categories:
    - Complexity, Simplification, Completeness, and Creativity

Winning categories

1. Complexity (of the system)
   - Challenge yourself to break apart something that isn't trivial to understand
     - 👎👎: making a sandwich from cold cuts and bread,
     - 👍👍: making a sandwich from sunlight, water and CO2,
2. Simplicity (of the model)
   - Try to boil the system down to its most important components.
     - 👎👎: an exact 1:1 replica of every chemical that goes into the sandwich
     - 👍👍: summarizing multiple simple steps that all depend on the same inputs
3. Completeness
   - Include as much detail as you need to explain the process
     - 👎👎: oversimplifying a critical step (someone says "Hey what about….", defend yourself!)
4. Creativity
   - Entertain our brains! Make it funny, intelligent, provocative. Get your audience interested.

It's impossible to win in all categories. You have to make choices.

Recap: What did we learn?

Discussion: Let's generate a list of biological systems that have been modeled mathematically (40 pts)

1. Break into groups
2. Search the web for any mathematically modeled system you can find
3. Narrow in on 1 or 2 you like in particular
4. Get some of the details of the system
5. Be prepared to explain how the model helped
6. We'll discuss as a class in 15-20 minutes

Project: Find a particular biological system that you would like to model for the class project

1. You can use any of those discussed today, but everyone needs their own
2. Write 3 paragraphs of background on the system (10 pts)
3. Write 1 paragraph on a recent finding that needs modeling (5 pts)
4. Draw a summary figure that explains the process (20 pts)
    a. If applicable, draw a reaction diagram
    b. Otherwise, just draw a general illustrative figure
5. Download and install MATLAB or OCTAVE on a laptop (5 pts)

Reading: Read the one-page handout and be ready for a simple quiz at the start of next week

1. There will be a 5 minute, 4 question (10 pts each) quiz at the start of next class.
2. It is just designed to determine if you read the handout.
3. You probably won't even have to think if you do the reading.

B.4.2 Modeling biological systems with differential equations

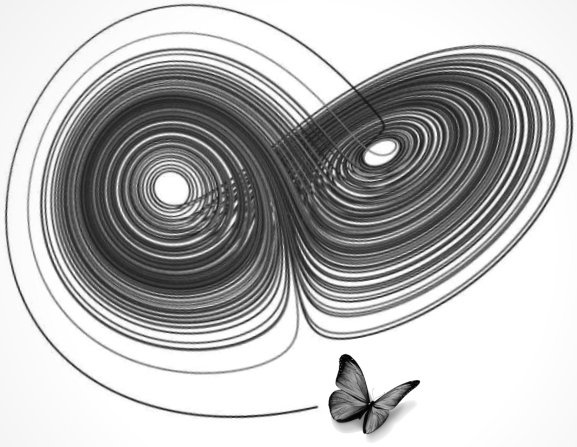
Modelling biological systems is a significant task of systems biology and mathematical biology. It involves the use of computer simulations of biological systems, including cellular subsystems (such as the networks of metabolites and enzymes which comprise metabolism, signal transduction pathways and gene regulatory networks), to both analyze and visualize the complex connections of these cellular processes." –Wikipedia

Many types of models Biological models can work in many different ways: In general, models are made to focus on systems at a certain scale. We can choose from a wide variety of model types. Some work by modeling continuous values, such as concentration of a protein. Others work by modeling the discrete state of a system, such as a neuron being either on or off. Some models are deterministic, meaning that there is no randomness, while others add stochastic noise.

Focusing on Differential equations In our class we?re going to focus on probably the simplest and most widely used kind of mathematical model, differential equations. Differential equations are a continuous, deterministic model, they model real values with rules that describe exactly how those values change. Differential equations are very similar to the equations we learned about in our high school and college calculus courses.

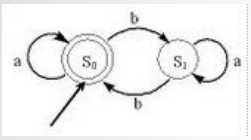
What is a differential equation good for? A differential equation is a simple formula that describes how a system will change in time based on the state of the system right now. You can think of it like a very formal protocol for how concentrations of substances will change over time. For example, a differential equation

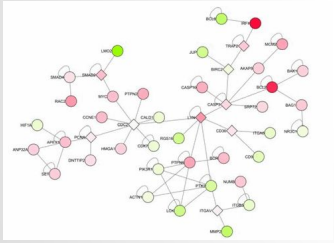
Biology and Dynamical Systems



Week 2: Modeling biological systems with differential equations

Mathematical modeling of biological systems can mean many different things



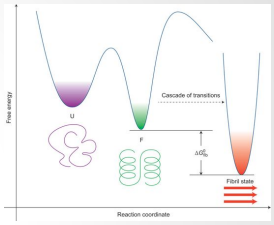


Discrete stochastic

Continuous stochastic

Discrete deterministic

Continuous deterministic



In this introductory class, we will focus on the simplest and most widely used type of continuous deterministic model: differential equations (diff E Q)

Today's aim is to show what differential equations are and how to create them from reaction diagrams

Outline:

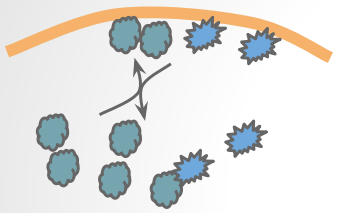
Describing rates of change

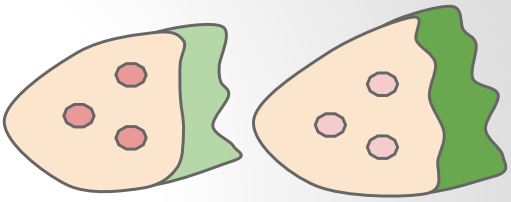
How to interpret reaction diagrams

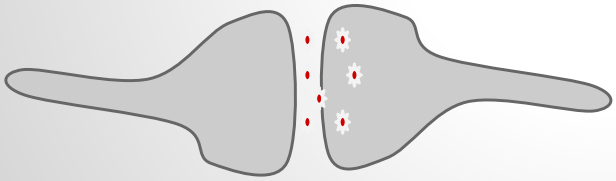
Converting reaction diagrams to differential equations

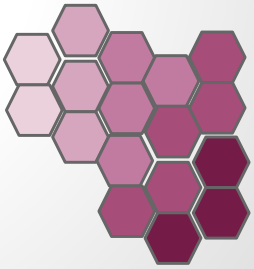
Examples

Biological models frequently aim to describe how proteins levels change over time (and space)

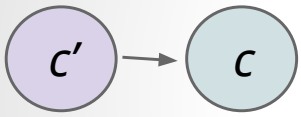




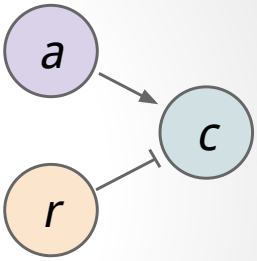




We use reaction diagrams to illustrate how protein levels change based on other cellular components



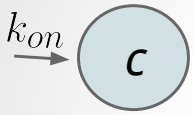
c' is converted into c



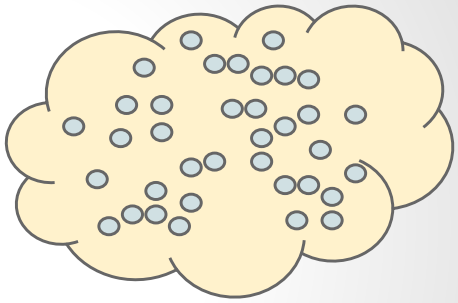
c is upregulated by a
c is downregulated by r

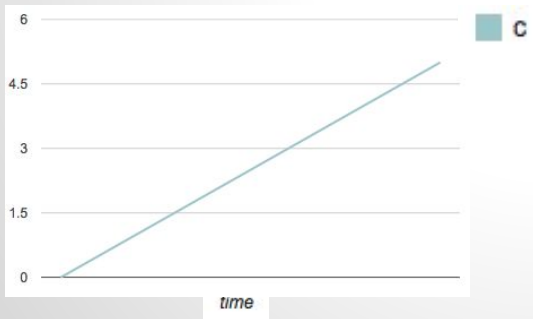
Differential equations formalize these relationships by defining protein level *rates of change*.

Differential equations state how protein levels change at every moment in time



Example: synthesis of a gene





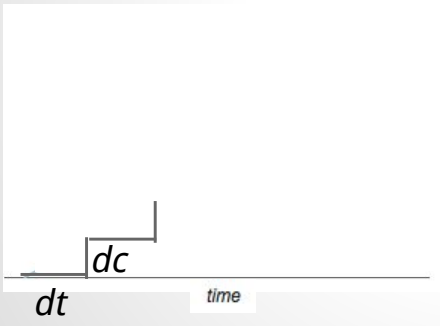
$$\frac{dc}{dt} = k_{on}$$

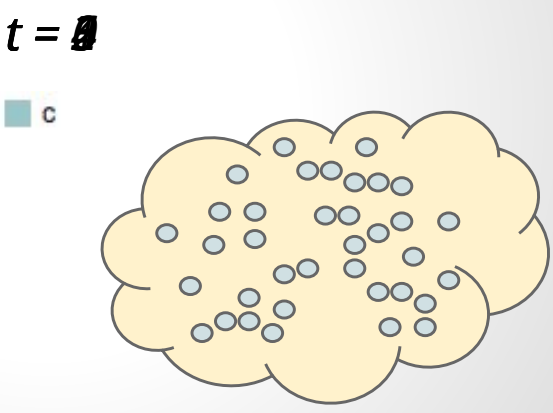
change in concentration → $\frac{dc}{dt} = k_{on}$ ← is a constant

over small time period →

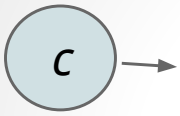
change in concentration → $\frac{dc}{dt} = k_{on}$ ← is a constant (in c/second)

over small time period →

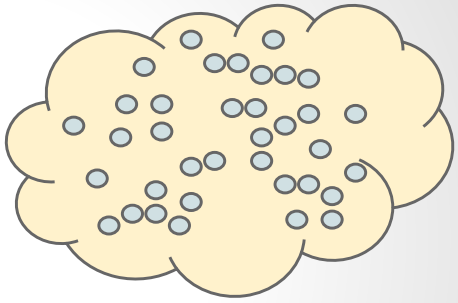




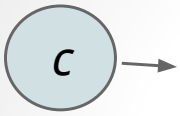
In most situations, a protein's rate of change will depend on how much is around



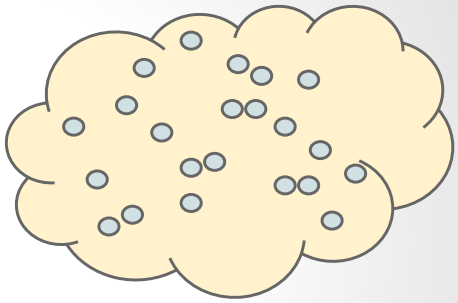
Example: degradation of a gene



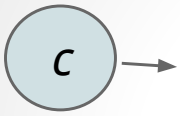
In most situations, a protein's rate of change will depend on how much is around



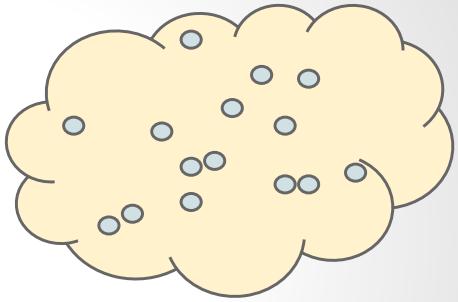
Example: degradation of a gene



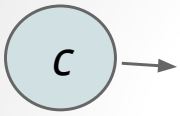
In most situations, a protein's rate of change will depend on how much is around



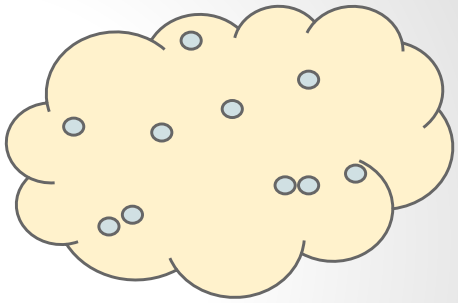
Example: degradation of a gene



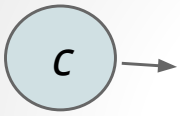
In most situations, a protein's rate of change will depend on how much is around



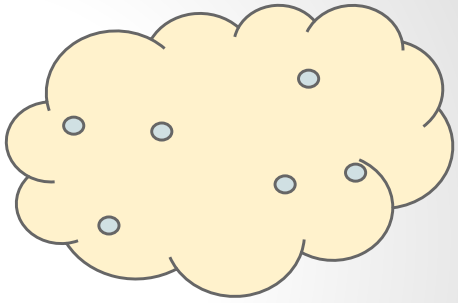
Example: degradation of a gene



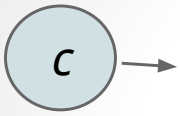
In most situations, a protein's rate of change will depend on how much is around



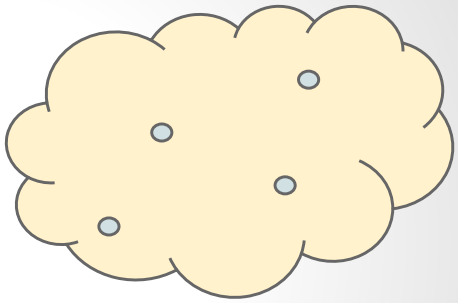
Example: degradation of a gene



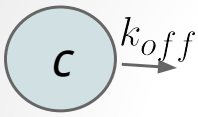
In most situations, a protein's rate of change will depend on how much is around



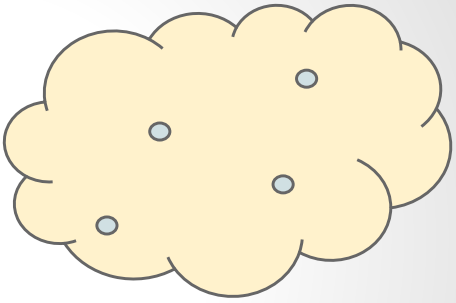
Example: degradation of a gene

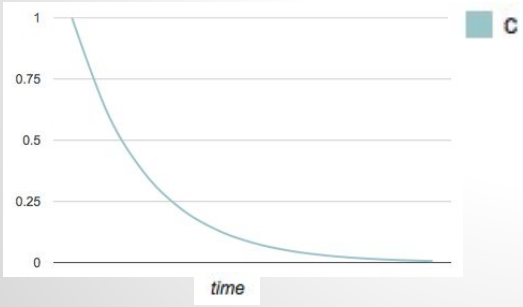


In most situations, a protein's rate of change will depend on how much is around



Example: degradation of a gene



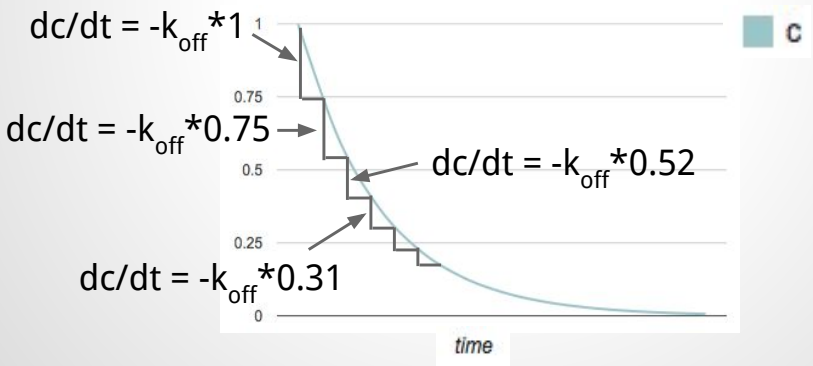


Note that rate depends on c

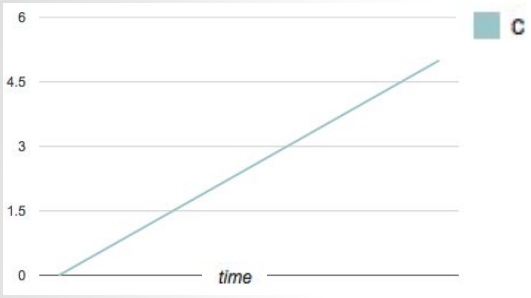
$$\frac{dc}{dt} = -k_{off}c$$

As the amount of *c* gets smaller, the rate of change gets smaller too

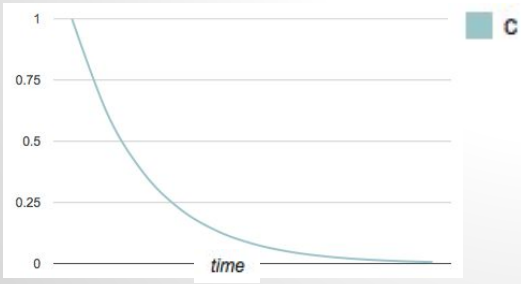
$$\frac{dc}{dt} = -k_{off}c$$



Discussion break:
What do k_on and k_off do?

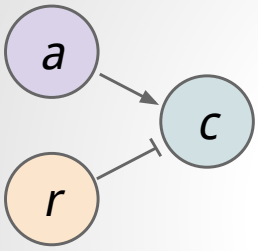


$$\frac{dc}{dt} = k_{on}$$

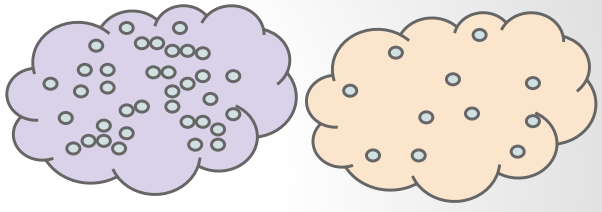


$$\frac{dc}{dt} = -k_{off}c$$

Most of the value in these models comes from describing how one factor affects another

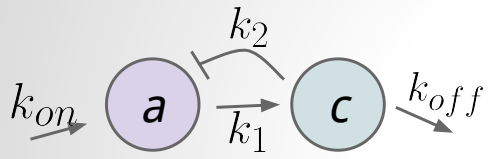


Example: gene regulation



$$\frac{dc}{dt} = a \cdot k_{on} - r \cdot k_{off}c$$

But the real beauty of diff eq is seeing how all the parts of the whole system work together



Example: gene feedback

$$\frac{dc}{dt} = k_1 a - k_{off} c = 0$$

$$\frac{da}{dt} = k_{on} - k_2 c \cdot a = 0$$

Next week we'll learn more about how to analyze these complex systems

Discussion: Let's convert a few reaction diagrams into differential equations (40 pts)

1. Break into groups
2. Each group will get a reaction diagram with a description of a system
3. Work for a few minutes to try to jot down a differential equation (10 pts)
4. We'll regroup after 15-20 minutes and go through each one
5. 10 pts if your group gets it right
6. 10 pts for explaining your reasoning
7. 10 pts for giving good feedback to others

Project: Convert your biological system into a differential equation

1. Continue with the system you wrote about in the last project assignment
2. Write a diff equation describing how your system changes in time (20 pts)
3. Write a few paragraphs on the logic that went into your equation (15pts)
4. Using program of choice, plot a fake result of a possible simulation (5pts)

Reading: Read the one-page handout and be ready for a simple quiz at the start of next week

1. There will be a 5 minute, 4 question (10 pts each) quiz at the start of next class.
2. It is just designed to determine if you read the handout.
3. You probably won't even have to think if you do the reading.

B.4.3 Visualizing equations with graphs

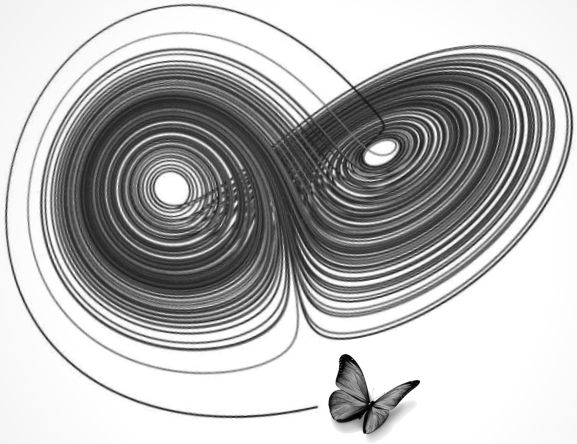
Why is it called differential? One could argue that differential equations could actually be called derivative equations. That?s because the equation itself is really just a way of explicitly writing out the derivative of a variable. For those that don?t remember a derivative is just ?a way to represent rate of change, that is - the amount by which a function is changing at one given point...it is the slope of the tangent line at a point on a graph.?

Equilibrium? Another important concept in diffeq is that of equilibrium. The meaning of equilibrium is simple once you have a rate equation to look at. When the net rates of change of all variable are equal to 0, then nothing can change anymore. The rates of production and destruction are balanced, and the concentrations of everything will remain constant forever. Not all systems reach equilibrium, but most do, and finding equilibria is the first step to understanding how many systems work. We will use equilibrium and steady state more or less interchangeably though there are subtle, pedantic differences.

Initial conditions and transient behaviors Even though most systems reach equilibrium after a long time, the system?s initial conditions set the early ?transient? behavior. The initial condition, or seed value, is simply the state of the system where we choose to start. The transient behavior is the way in which the system changes from the initial conditions to the steady state.

What are nullclines? If our rate equation has two variables that can change in it (i.e. not constants), then there is no longer a single value at which the rate equation will equal 0. Instead there will be a set of values that make the rate 0. This set of values defines a line, and this line is called a nullcline.

Biology and Dynamical Systems



Week 3: Visualizing equations with graphs

Today's aim is to show what we can do to visualize our equations so we can think about them concretely

Outline:

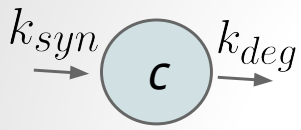
What to do when equations get tricky

Graphically visualizing rates of change

Visualizing the meaning of equilibrium

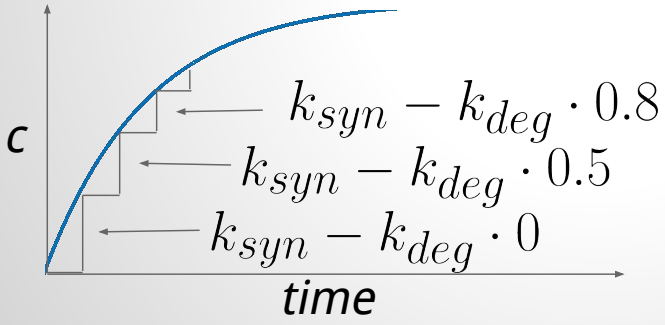
More complicated examples

What happens if we look at a protein that's being synthesized and degraded



synthesis and degradation of gene

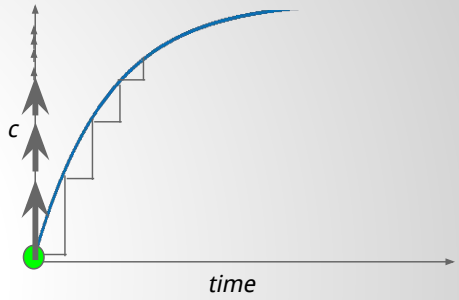
$$\frac{dc}{dt} = k_{syn} - k_{deg}c$$

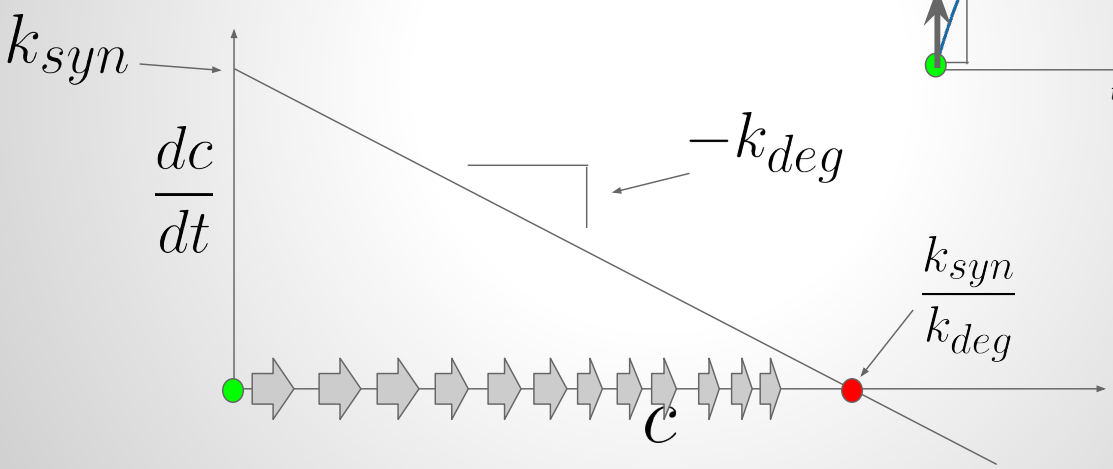


Does it ever stop? When?

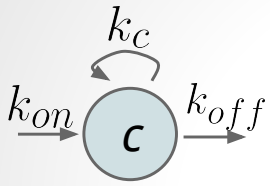
We graph dc/dt to see the protocol for how *c* changes.

$$\frac{dc}{dt} = k_{syn} - k_{deg}c$$



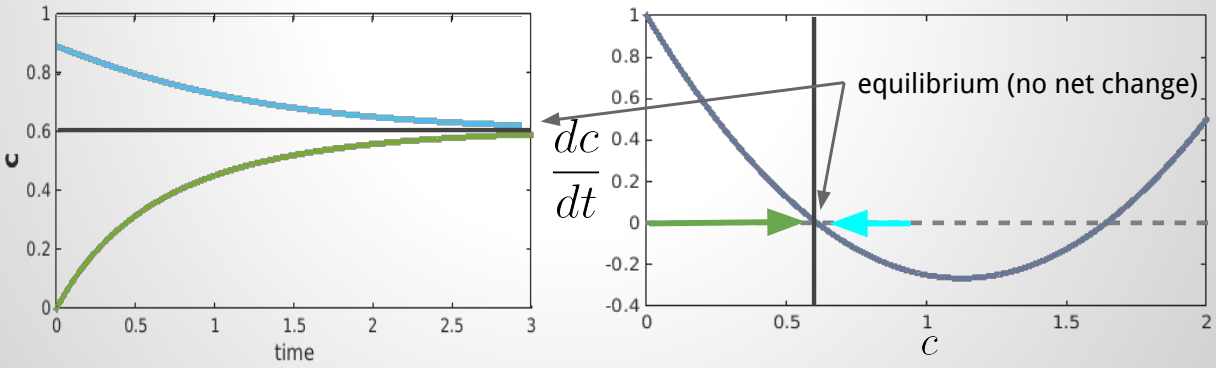


This same graphical presentation can be used to analyze more and more complicated systems

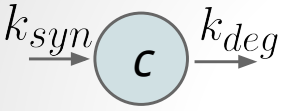


Example: gene auto-regulation

$$\frac{dc}{dt} = k_{on} + k_c c^2 - k_{off} c$$

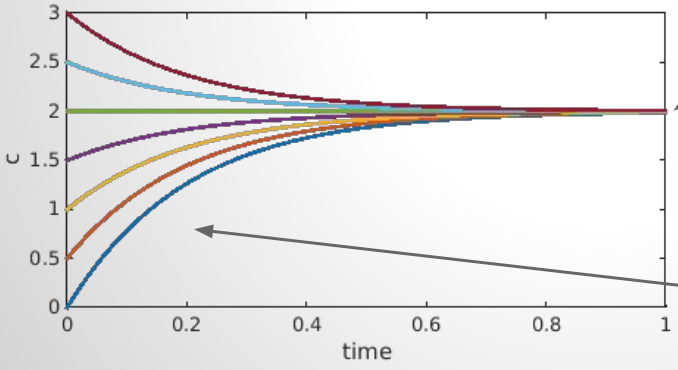


We can find equilibrium points by looking for concentrations at which the net change equals zero



Example: gene auto-regulation

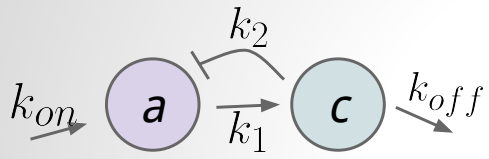
$$\frac{dc}{dt} = k_{syn} - k_{deg} \cdot c = 0$$



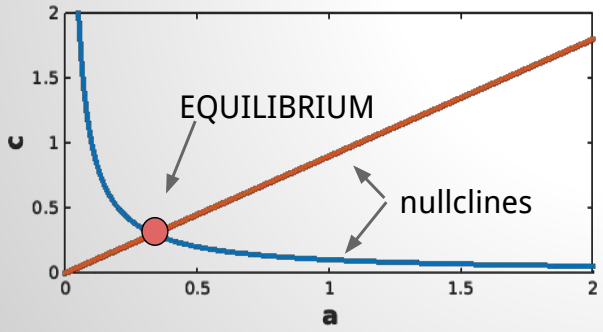
EQUILIBRIUM is important: it is where the system ends up after you wait long enough

Different initial conditions only give rise to different “transient“ behaviors

Graphing equilibria makes it much easier to understand two-component systems



Example: gene feedback



$$\frac{dc}{dt} = 0 = k_1 a^* - k_{off} c^*$$

$$\frac{da}{dt} = 0 = k_{on} - k_2 c^* \cdot a^*$$

$$c^* = \frac{k_1 a^*}{k_{off}}$$

$$c^* = \frac{k_{on}}{k_2 a^*}$$

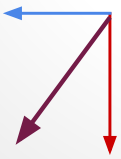
The "protocol" describes how *c* and *a* change depending on their current values

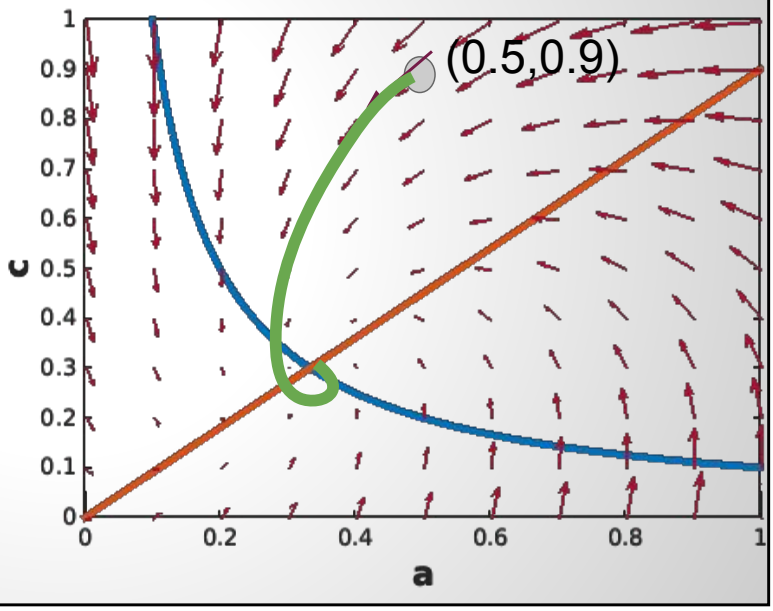
$$\frac{dc}{dt} = \overset{9}{k_1} a - \overset{10}{k_{off}} c$$

$$\frac{da}{dt} = \overset{1}{k_{on}} - \overset{10}{k_2} c \cdot a$$

$$\frac{dc}{dt} = -4.5$$

$$\frac{da}{dt} = -3.5$$





Question: What would happen if we multipied all of the constants (k) by 10?

$$\frac{dc}{dt} = \overset{9}{k_1}a - \overset{10}{k_{off}}c$$

$$\frac{da}{dt} = \overset{1}{k_{on}} - \overset{10}{k_2}c \cdot a$$

vs

$$\frac{dc}{dt} = \overset{90}{k_1}a - \overset{100}{k_{off}}c$$

$$\frac{da}{dt} = \overset{10}{k_{on}} - \overset{100}{k_2}c \cdot a$$

Discussion: Let's convert a few reaction diagrams into differential equations (40 pts)

1. Break into groups
2. Each group will get a differential equation with a description of a system
3. Work for a few minutes to plot the nullclines and the arrows (10 pts)
4. We'll regroup after 15-20 minutes and go through each one
5. 10 pts if your group get both the nullcline and the arrows plotted
6. 10 pts for explaining your reasoning
7. 10 pts for giving good feedback to others

Project: Simplify and graph your system

1. Continue with the system you wrote about in the last project assignment
2. Try to simplify your equations to have two components (5 pts)
    a. You can assume that everything else is magically held constant
3. Write the nullcline equations and graph them for the system(10pts)
4. Find any relevant equilibrium points (5 pts)
5. Draw the “protocol” arrows on your graph and trace a sample trajectory
    a. 5 pts for hand drawn, 5 pts for MATLAB plots
6. Write a few paragraphs describing your logic (10pts)

Reading: Read the one-page handout and be ready for a simple quiz at the start of next week

1. There will be a 5 minute, 4 question (10 pts each) quiz at the start of next class.
2. It is just designed to determine if you read the handout.
3. You probably won’t even have to think if you do the reading.

B.4.4 Simplifying models and starting simulations

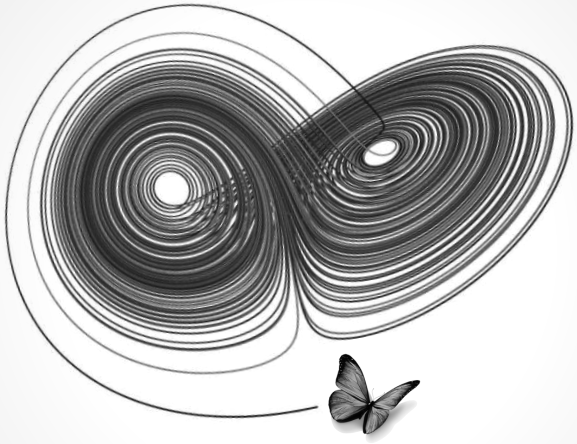
Why simplify before simulating? There are a number of reasons to simplify your equations before you ever start a simulation. Here are my top three. 1) You won?t be able to remember or adequately describe all the parameters that went into a given simulation when somebody asks you. If you have 20 parameters in a simulation there will always be a huge risk that you screwed one of them up and you will never be able to notice. 2) If you want to understand how your model works under many different conditions, then every time you add a parameter you have to do exponentially more work. In other words, when you are systematically checking how 3 parameters work together you may have to do 100^3 different simulations and analyze your results. If you add one more parameter you will have to do 100^4 simulation or 100 times more work. 3) Nobody wants to look at a disorganized giant equation. If you have a complicated equation most people will decide it isn?t worth their time to pay attention to the modeling so none of your modeling work will count for anything anyway.

The parts of a differential equation model A differential equation model can be broken down into 3 parts: variables, parameters, and the form of the equation. Variables are the values (like concentration) that are changing in time. This is the physically ?real? stuff like the number of proteins or the size of your cell. Parameters are things like rate constants and diffusion coefficients. These are descriptive numbers that set the magnitudes of rates of change. Finally, the form of the equation determines how the variables and parameters fit together. This is the level at which we can see how each variable affects the other qualitatively, but it takes parameters to know by how much.

How do simulations work? The applied math behind solving differential equations with computers is rich and we certainly won?t understand everything that has been done over the past 70 years to solve these problems. However we can get some intuition by looking at the simplest way of solving, called the Euler method. In this method you ?step? through time and change the value of your variables by a small amount using the rules defined in your differential equation (See wikipedia page for

more info). If you ever try this, you will get bored very fast because it is really annoying. But when computers do this, they can take very small steps and therefore ?solve? the system with arbitrary accuracy. There have been a bunch of modifications of this technique that significantly improve the accuracy vs timestep tradeoff.

Biology and Dynamical Systems



Week 4: Simplifying models and starting simulations

Today's aim is to show what we can do to visualize our equations so we can think about them concretely

Outline:

Why you do we simplify our models?

Reducing variables and equations

Running Simple Simulations

Why we have to simplify our models (before we start plugging away at simulations)

We really can't keep more than a handful of things in our head at once

7 7 3 2 6 3 0 1 8 1
vs
(773) 263-0181

Every free parameter in a model means you have to do **exponentially** more simulation and analysis

5 params vs 6 params
x^5 x^6

No one wants to look at a totally disorganized mess of equations

$$\frac{dc}{dt} = \frac{k_1}{k_2}c + k_{ph}c - k_r c + \frac{K^2}{k_b T}$$

vs

$$\frac{dc}{dt} = k_{on} - k_{off}c$$

There are really 3 parts to a model

$$\frac{dc}{dt} = k_{on} - k_{off}c$$

Variables: the values (like concentration) that are changing,

a,b,c,x,y,z

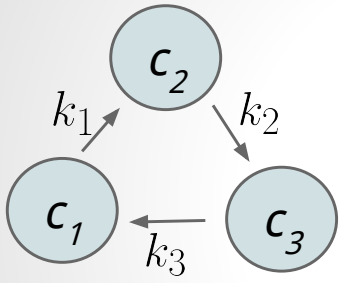
Parameters: the constants that define the system

k_on, k_off, D, α

The form of the equation: where do the variables show up in the equation

___dc/dt = ___ - ___c

The most important simplification is definitely limiting the number of variables/equations



cyclical reaction

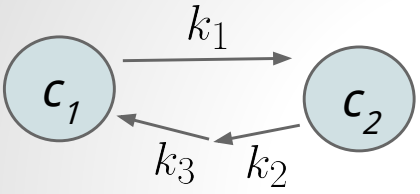
$$c_1 + c_2 + c_3 = C$$

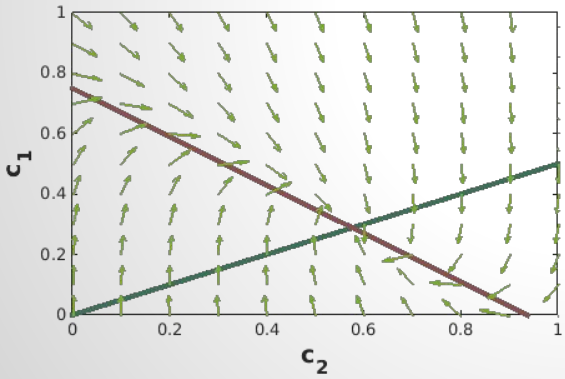
$$\frac{dc_1}{dt} = k_3 c_3 - k_1 c_1$$

$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

$$\frac{dc_3}{dt} = k_2 c_2 - k_3 c_3$$

The equation got slightly more complicated, but now we no longer have to worry about c_3 at all





$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1$$

$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

$$c_1^* = \frac{k_3 C}{k_1 + k_3} - \frac{k_3 c_2^*}{k_1 + k_3}$$

$$c_1^* = \frac{k_2 c_2^*}{k_1}$$

We can use a numerical differential equation solver to simulate the output of a simplified system

$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1 \qquad \frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

we must pick explicit values for each of our constants in the equations

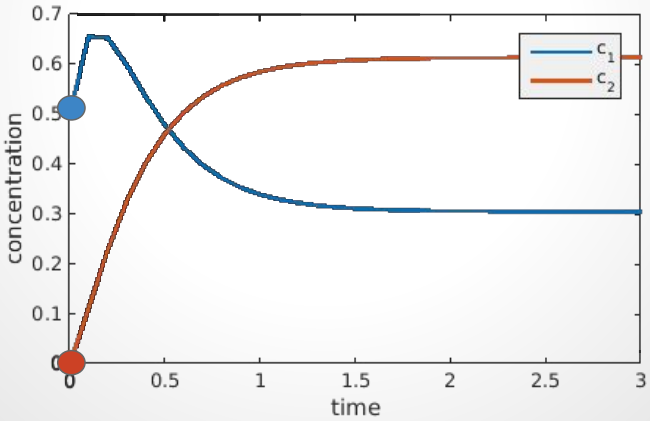
$$\frac{dc_1}{dt} = 1 * 8 - 8 * c_2 - (2 + 8) * c_1 \qquad \frac{dc_2}{dt} = 2 * c_1 - 1 * c_2$$

we must pick explicit starting values for both c_1 and c_2

$$c_1 = 0.5 \qquad c_2 = 0$$

The computer simulates by simply following the "protocol" over lots and lots of small steps

$$\frac{dc_1}{dt} = 1 * 8 - 8 * c_2 - (2 + 8) * c_1 \qquad \frac{dc_2}{dt} = 2 * c_1 - 1 * c_2$$



In MATLAB, all we have to do is define the equation and then set our variables and initial conditions

$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1$$

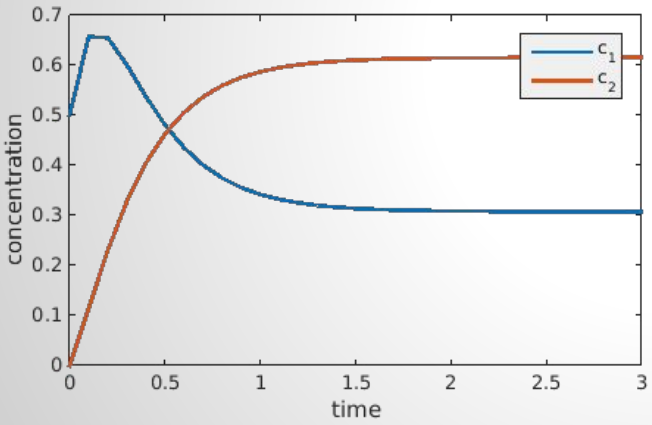
$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

```
function dc = myode(t,c,C,k1,k2,k3)
        c1 = c(1);
        c2 = c(2);
        dc1 = k3*C - k3*c2 - (k3+k1)*c1;
        dc2 = k1*c1 - k2*c2;
        dc = [dc1;dc2];
end
```

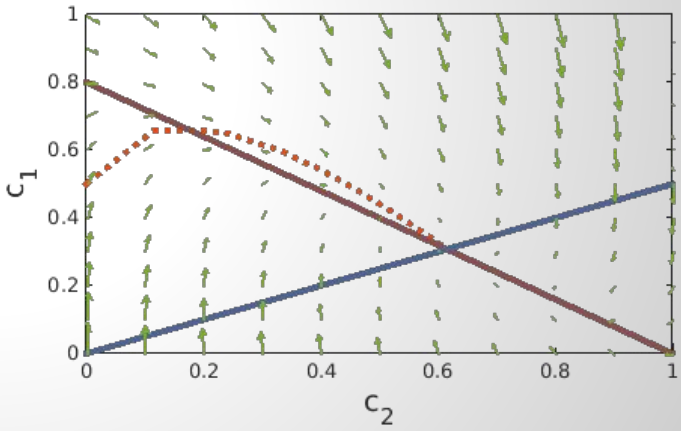
```
t = 0:0.1:10;
c0 = [1,0];

C = 1; k1=2; k2=1; k3=8;

[t, c] = ode45(@myode,t,c0,[],C,k1,k2,k3);
```

We can plot the output as "concentrations vs time" or on top of the nullcline and arrows as "c_1 vs c_2"

plot(t,c(:,1),t,c(:,2))



plot(c(:,2),c(:,1))



Discussion: Let's do some simple simulations (40 pts)

1. Break into groups
2. Each group will get a differential equation with a description of a system
3. Work for a few minutes to implement the simulations
    a. First convert the 3 component system into 2 components (5 pts)
    b. Change the sample code to implement the new equation (5 pts)
    c. Plot both types of output graphs (10 pts)
4. We'll regroup after 15-20 minutes and go through each one
5. 10 pts for explaining your reasoning
6. 10 pts for giving good feedback to others

Project: Simplify and graph your system

1. Continue with the system you wrote about in the last project assignment
2. Try to simplify your equations again (5 pts)
    a. Try to use tricks like the ones we came up with today
3. Pick constant values and initial conditions that makes sense (5 pts)
4. Simulate your simplified system and graph the outputs (10pts)
5. Change the constants and document how the output changes (10 pts)
6. Explain what your simulation taught you about your system (10 pts)
    a. How did the behavior depend on your choice of constants

Reading: Read the one-page handout and be ready for a simple quiz at the start of next week

1. There will be a 5 minute, 4 question (10 pts each) quiz at the start of next class.
2. It is just designed to determine if you read the handout.
3. You probably won't even have to think if you do the reading.

B.4.5 Analyzing equations and understanding simulation output

(from wikipedia) Nondimensionalization is the partial or full removal of units from an equation involving physical quantities by a suitable substitution of variables. This technique can simplify and parameterize problems where measured units are involved. It is closely related to dimensional analysis. In some physical systems, the term scaling is used interchangeably with nondimensionalization, in order to suggest that certain quantities are better measured relative to some appropriate unit. These units refer to quantities intrinsic to the system, rather than units such as SI units. Nondimensionalization is not the same as converting extensive quantities in an equation to intensive quantities, since the latter procedure results in variables that still carry units.

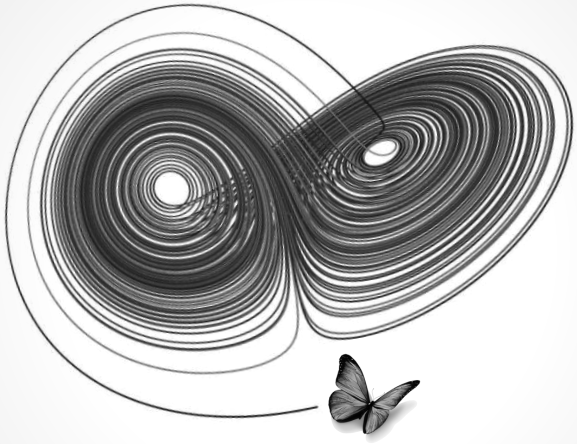
Running simulations : The following two snippets of code is all you need to solve differential equations in MATLAB.

```
function dc = myode(t,p,a,b)
    p1 = p(1);
    p2 = p(2);
    dp1 = 1 - a*p2 - p1;
    dp2 =p1 - b*p2;
    dp = [dp1;dp2];
end
```

```
t = 0:0.1:10;      #timepoints for solution (1 to 10 increment of 0.1)
p0 = [1,0];         #initial conditions for the concentration of p1 and p2
a=1; b=1;          #optional constants a and b which will change behavior
[t, pt] = ode45(@myode,t,p0,[],a,b);        #this runs the simulaton
plot(t,p(:,1)); hold on; plot(t,p(:,2));           #this plots the output
```

So now you can put whatever you want into the equation and change your parameters and run many simulations.

Biology and Dynamical Systems

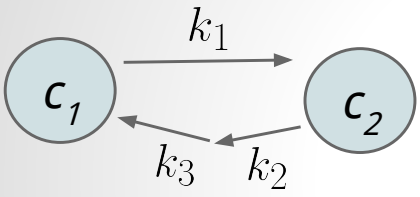


Week 5: Analyzing equations and understanding simulation output

Today's aim is to show how our models are a way to provide clarity on the complex systems

1. We've seen that raw simulation output can be unwieldy
2. Looking for qualitative differences in simulations
3. Reducing the number of extraneous constants
4. Finding a phase diagram

How many parameters do we have to vary to get a good idea of how our model works?



$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1$$

$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

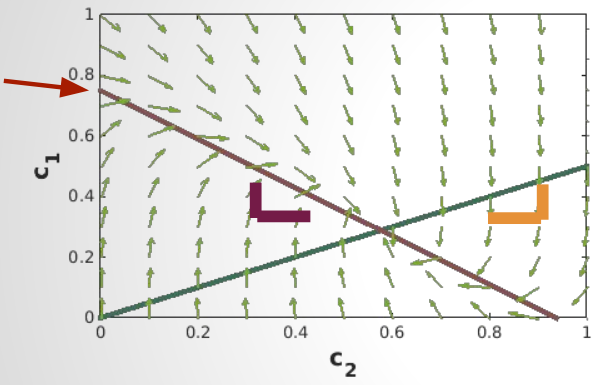
What happens if you double k_3 or k_1?

What about doubling C and halving k_3?

It is nice to know that the computer understands the problem. But I would like to understand it too.

--Eugene Wigner

Constants set quantitative values of a model, but it's often better to look for qualitative differences



$$c_1^* = \frac{k_3 C}{k_1 + k_3} - \frac{k_3 c_2^*}{k_1 + k_3}$$

$$c_1^* = \frac{k_2 c_2^*}{k_1}$$

We can get the same equilibrium value for different parameter values

Can equilibrium become impossible? How?

Limiting the number of parameters

$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1$$

$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

```
function dc = myode(t,c,C,k1,k2,k3)
        c1 = c(1);
        c2 = c(2);
        dc1 = k3*C - k3*c2 - (k3+k1)*c1;
        dc2 = k1*c1 - k2*c2;
        dc = [dc1;dc2];
end
```

At first glance, it appears that we have to understand how the 4 parameters come together and how each one affects the others. But based on the form of the equation we know that we really only care about 2 parameters

Using variable substitution to simplify all of those extraneous constants

$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1 \qquad \frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

substitute variables so that we hide 1 constant for each variable

$$\tau = (k_1 + k_3) t \qquad \rho_1 = \frac{k_1 + k_3}{k_3 C} c_1 \qquad \rho_2 = \frac{(k_1 + k_3)^2}{k_1 k_3 C} c_2$$

rename the remaining factors so that everything looks pretty

$$\alpha = \frac{k_1 k_3}{(k_1 + k_3)^2} \qquad \beta = \frac{k_2}{k_1 + k_3}$$

By making those substitutions we can reduce our system to depend on just 2 parameters

$$\frac{dc_1}{dt} = k_3 C - k_3 c_2 - (k_3 + k_1) c_1 \qquad \frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

$$\frac{d\rho_1}{dt} = 1 - \alpha\rho_2 - \rho_1 \qquad \frac{d\rho_2}{dt} = \rho_1 - \beta\rho_2$$

$$\rho_1^* = 1 - \alpha\rho_2^* \qquad \rho_1^* = \beta\rho_2^*$$

$$c_1^* = \frac{k_3 C}{k_1 + k_3} - \frac{k_3 c_2^*}{k_1 + k_3} \qquad c_1^* = \frac{k_2 c_2^*}{k_1}$$

Saving time and complexity

We went from needing to understand

6 parameters and 3 variables ---------->

To only needing 2 parameters and 2 variables

$$\frac{dc_1}{dt} = k_3 c_3 - k_1 c_1$$

$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

$$\frac{dc_3}{dt} = k_2 c_2 - k_3 c_3$$

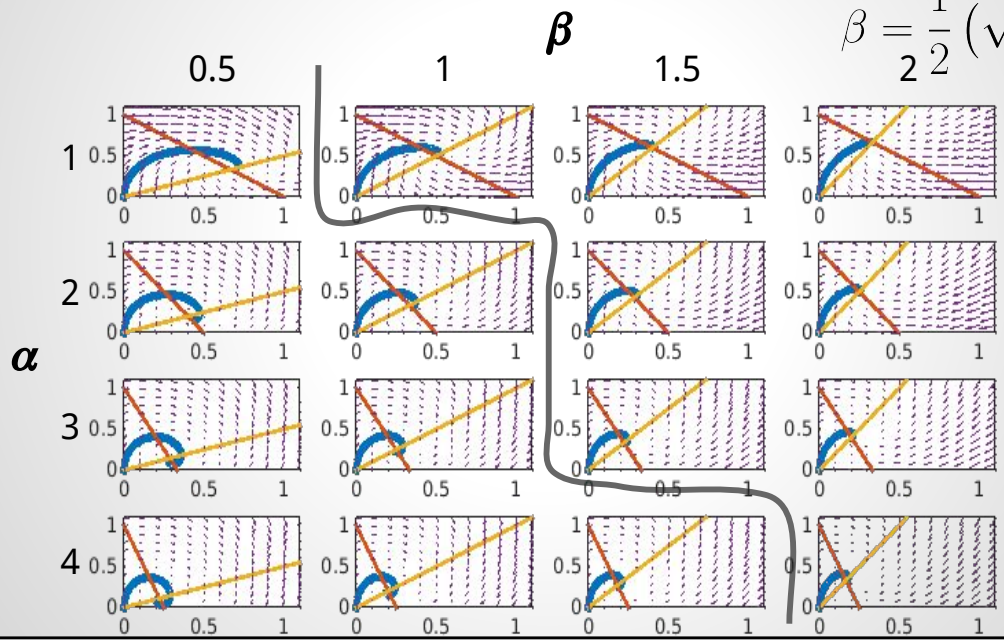
$$\frac{d\rho_1}{dt} = 1 - \alpha\rho_2 - \rho_1$$

$$\frac{d\rho_2}{dt} = \rho_1 - \beta\rho_2$$

```
function dc = myode(t,p,a,b)
        p1 = p(1);
        p2 = p(2);
        dp1 = 1 - a*p2 - p1;
        dp2 =p1 - b*p2;
        dp = [dp1;dp2];
end
```

With 2 variables we can see how the qualitative behavior changes depending on the value of α and β

$$\beta = \frac{1}{2}\left(\sqrt{4\alpha - 3} - 1\right)$$



You might be asking, "Was there some sleight of hand? How can you make the problem easier like that?"

Basically, what we did was pick the units of concentration and the units of time in a special way such that our parameters canceled out of the equations. Picking very special units doesn't matter because we can always get to new units by multiplying everything by the same amount.

For example, if I say some reaction takes 2 hours and then we change units so that it takes 72000 seconds, that doesn't mean the dynamical system changes in any way. All the numbers get bigger by the same amount.

Discussion: Let's simplify some equations (40 pts)

1. Break into groups
2. Each group will get a differential equation with a description of a system
3. Reduce the number of extra constants (5 pts)
4. Implement a simulation with the simplified equation (5pts)
5. Diagram how the behavior changes as the constants are varied (10 pts)
6. We'll regroup after 15-20 minutes and go through each one
7. 10 pts for explaining your reasoning
8. 10 pts for giving good feedback to others

Project: Diagram the behaviors of your system

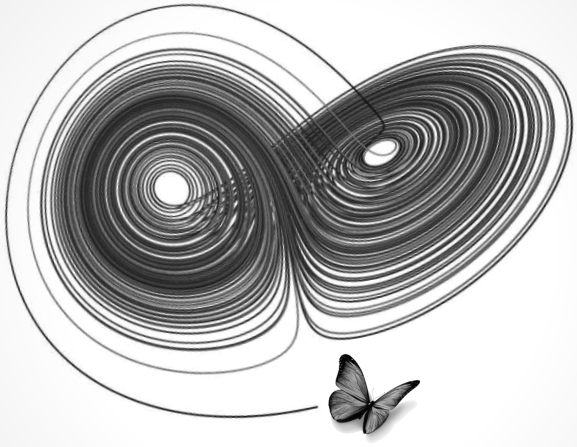
1. Continue with the system you wrote about in the last project assignment
2. Try to simplify your equations again (10 pts)
    a. Try to use tricks like the ones we came up with today
3. Modify your simulation to reflect your simplification (5 pts)
4. Systematically vary parameters to find qualitative changes (10pts)
5. Describe the meaning of each of your simplified parameters (15 pts)
    a. explain the role that each constant plays in setting the dynamics

Reading: Read the one-page handout and be ready for a simple quiz at the start of next week

1. There will be a 5 minute, 4 question (10 pts each) quiz at the start of next class.
2. It is just designed to determine if you read the handout.
3. You probably won't even have to think if you do the reading.

B.4.6 Explaining ever more complex systems

Biology and Dynamical Systems



Week 6: Explaining ever more complex systems

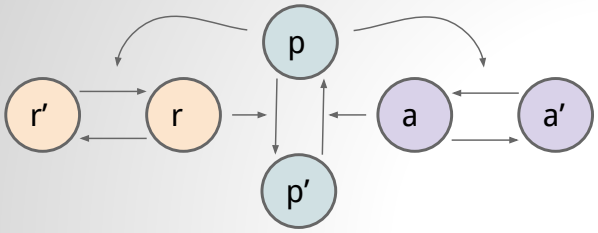
Today's aim is to get us working with systems that have 3 or more components

1. Quick review of the methods we learned for two component systems
2. Analyzing a 3 component system
3. We still need to boil down our models to focus on the key factors

What tools have we learned for building and analyzing mathematical models?

1. Converting reaction diagrams to equations
2. Graphing equilibrium points
3. Reducing the number of equations
4. Non-dimensionalization to remove extra constants
5. Running simulations and graphing output
6. Plotting nullclines and phase space plots

Looking at a three component system with feedback



activator and repressor

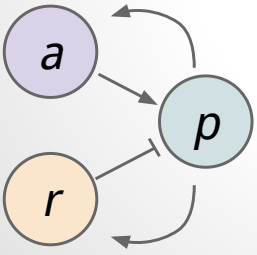
$$\frac{dr'}{dt} = k_1' r + k_1^p r p' - k_1' r' \qquad \frac{dr}{dt} = -k_1' r - k_1^p r p' + k_1' r'$$

$$\frac{da'}{dt} = k_2' a + k_2^p a p' - k_2 a' \qquad \frac{da}{dt} = -k_2' a - k_2^p a p' + k_2 a'$$

$$\frac{dp'}{dt} = k_3' p + k_3^a p a' - (k_3 + k_3^r r') p'$$

$$\frac{dp}{dt} = -k_3' p - k_3^a p a' + (k_3 + k_3^r r') p'$$

$$r' + r = r_T \qquad a' + a = a_T \qquad p' + p = p_T$$



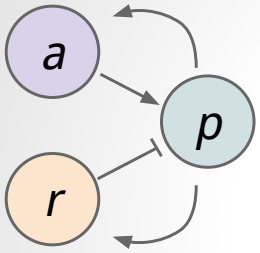
simplified diagram

$$\frac{dp}{dt} = k_{on}^p + k_{on}^{pa} a - (k_{off} + k_{off}^{pr} r) p$$

$$\frac{da}{dt} = k_{on}^a + k_{on}^{ap} p - k_{off}^a a$$

$$\frac{dr}{dt} = k_{on}^r + k_{on}^{rp} p - k_{off}^r r$$

We can reduce complexity by eliminating equations



$$\frac{dp}{dt} = k^p_{on} + k^{pa}_{on}a - (k_{off} + k^{pr}_{off}r)p$$

$$\frac{da}{dt} = k^a_{on} + k^{ap}_{on}p - k^a_{off}a$$

$$\frac{dr}{dt} = k^r_{on} + k^{rp}_{on}p - k^r_{off}r$$

start: 9 equations + 13 constants

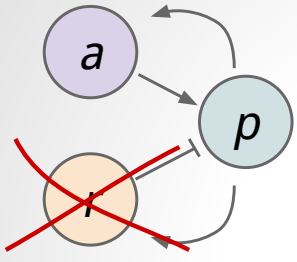
finish: 3 equations + 4 constants

$$\frac{dp}{dt} = 1 + \gamma_a a - (1 + \gamma_r r)p$$

$$\frac{da}{dt} = 1 + \beta_a p - \frac{a}{\kappa_a}$$

$$\frac{dr}{dt} = 1 + \beta_r p - \frac{r}{\kappa_r}$$

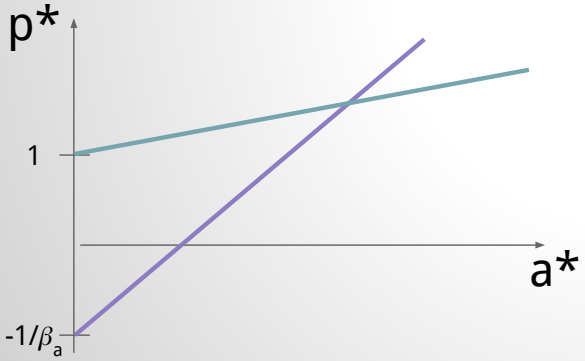
To start, let's see what happens if r = 0



$$\frac{dp}{dt} = 1 + \gamma_a a - (1 + \cancel{\gamma_r r})p$$

$$\frac{da}{dt} = 1 + \beta_a p - \frac{a}{\kappa_a}$$

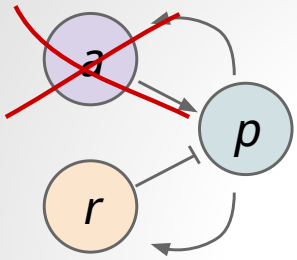
$$\cancel{\frac{dr}{dt} = 1 + \beta_r p - \frac{r}{\kappa_r}}$$



$$p^* = 1 + \gamma_a a^*$$

$$p^* = -\frac{1}{\beta_a} + \frac{a^*}{\kappa_a \beta_a}$$

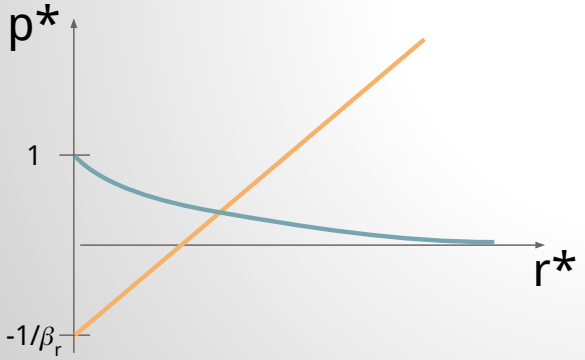
and what if a = 0



$$\frac{dp}{dt} = 1 + \gamma_a a - (1 + \gamma_r r)p$$

$$\frac{da}{dt} = 1 + \beta_a p - \frac{a}{\kappa_a}$$

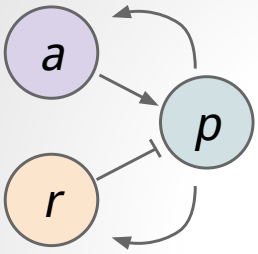
$$\frac{dr}{dt} = 1 + \beta_r p - \frac{r}{\kappa_r}$$



$$p^* = \frac{1}{1 + \gamma_r r^*}$$

$$p^* = -\frac{1}{\beta_r} + \frac{r^*}{\kappa_r \beta_r}$$

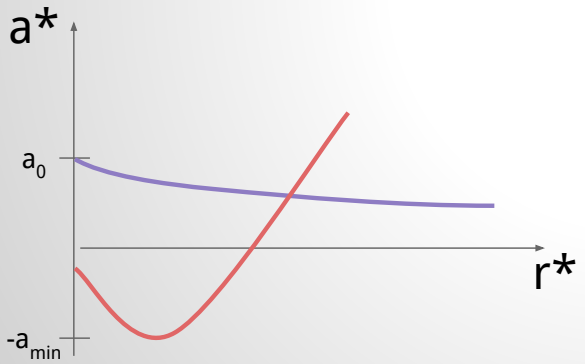
Let's just see what this system looks like if p very close to its equilibrium value.



$$\frac{dp}{dt} \approx 0 \Rightarrow \quad p^* = \frac{1 + \gamma_a a^*}{1 + \gamma_r r^*}$$

$$\frac{da}{dt} = 1 + \beta_a \frac{1 + \gamma_a a}{1 + \gamma_r r} - \frac{a}{\kappa_a}$$

$$\frac{dr}{dt} = 1 + \beta_r \frac{1 + \gamma_a a}{1 + \gamma_r r} - \frac{r}{\kappa_r}$$



$$a^* = \frac{a_0 + \omega_1 r^*}{1 + \omega_2 r^*}$$

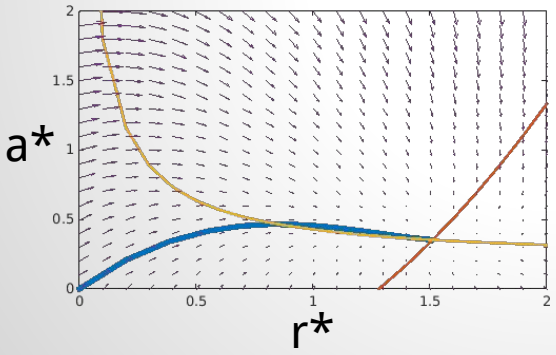
$$a^* = \omega_r (r - r_0)^2 - a_{min}$$

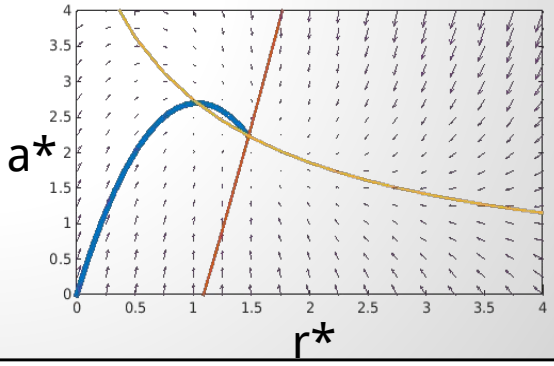
Now that we have simplified and gathered some intuition, we can start analyzing simulations

$$\frac{da}{dt} = 1 + \beta_a \frac{1+\gamma_a a}{1+\gamma_r r} - \frac{a}{\kappa_a}$$

$$\frac{dr}{dt} = 1 + \beta_r \frac{1+\gamma_a a}{1+\gamma_r r} - \frac{r}{\kappa_r}$$

```
function dc = myode2(t,c,ba,br,ya,yr,ka,kr)
        a = c(1);
        r = c(2);
        da = 1 +ba*(1+ya*a)/(1+yr*r) - a/ka;
        dr = 1 +br*(1+ya*a)/(1+yr*r) - r/kr;
        dc = [da;dr];
end
```





Discussion: Let's finish this problem (40 pts)

1. Break into groups
2. Each group works on a simulation of the full 3 component system (10pts)
3. We'll regroup after 15-20 minutes
4. Every group should present 5 minutes on one interesting finding (10 pts)
5. 10 pts will be awarded for the clearest explanation
6. 10 pts for giving good feedback to others

Project: Go forth and conquer

Keep working with your models! Solve your problem and present it clearly.

B.5 Quizzes and Project ideas

W2 Q1: Which is not a situation that is amenable to mathematical modeling? A metabolism, signal transduction pathways, gene regulatory networks, protein purification

Q2: Which is a category of biological mathematical model? A soft, prescient, discrete, topological

Q3: Which type of model is a differential equation? A continuous deterministic, discrete stochastic, continuous stochastic, discrete deterministic

Q4: What does a differential equation most closely resemble? A a protocol, a patent, a dissertation, a legume

W3 Q1: Which is the closest synonym for derivative in the context of diffeq? A Unoriginal, by-product, change, sum

Q2: Which is least similar to equilibrium? A balanced rates, constant concentrations, discrete probabilities, steady states

Q3: Which conditions set the transient behavior? A initial, parametric, hyperbolic, final

Q4: Nullclines are... A topographical, unimportant, lines, esoteric

W4 Q1: Number of simulations to run varies — with the number of free parameters. A quadratically, logarithmically, exponentially, combinatorially

Q2: Which is not a part of a differential equation model? A definitions, variables, parameters, the form of the equation

Q3:The Euler method gives exact solutions. A True, False, neither, depends who you ask

Q4: Who likes looking at a disorganized mess of an equation? A Your PI, your colleagues, your reviewer, nobody

W5 Q1: Once nondimensionalization is complete? A All variables have units of time, there are no more parameters left, equations are simpler to analyze, your solutions will fall on a line

Q2: In MATLAB — is the function for solving differential equations A ode23tb, ode113, ode45, ode15i

Q3: Which isn?t necessary to solve a differential equation A An ode function, constants, initial conditions, timepoints for solution

APPENDIX C
REDUCING POWER CONSUMPTION IN HIGH PERFORMANCE COMPUTING

C.1 Introduction

Data centers in the US consume an estimated 91 billion kilowatt-hours yearly, equivalent to the annual output of 34 large coal-fired power plants.[20] These same estimates show that only 6-12% of the electricity is used for powering servers while the rest is used to keep machines idling, wasting resources and money in the process. Data center electricity is not inexpensive, costing American businesses $13 billion annually in electricity bills.[20] Because cost is a strong motivating factor for businesses and universities, we consider data center energy efficiency in the context of cost savings for data center operations.

Demand response (DR) programs provide incentives to induce dynamic management of customers electricity load in response to power supply conditions, for example, reducing their power consumption in response to a request from the utility.[98] Anita can you add something around here that points out how demand response also increases the sustainability of the datacenter beyond just cost reduction.Many energy providers have Voluntary Load Response (VLR) programs, which encourage commercial consumers to reduce power demands during peak periods, such as particularly hot summer days. Participants are given between one and four hours notice of a request to shed some of their electric load, with two and eight hours of participation and the expectation to shed at least 10 kilowatts. We are interested in exploring more active ways in which to participate in electricity demand response programs while impacting the users minimally.

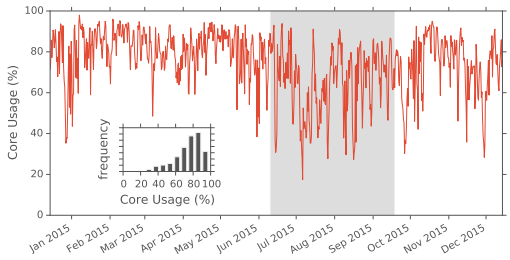


Figure C.1: Average core usage for a 244 node shared HPC partition in the Midway cluster. Insert shows usage statistics histogram.

In many university data centers, a significant portion of the data center is dedicated to high performance research computing which is typically Tier 1. While these jobs take longer periods of time to complete, they are less time sensitive and more flexible than systems which support core business functions such as the university's email. We wish to use the flexibility in scheduling of these jobs to reduce energy consumption of university data centers during periods of peak energy demand.

As shown in the example core usage data of Figure C.1, although the typical average usage during the school year is a fairly standard 80%, the averaged workload can fall to 65% of full capacity in the hottest summer months from June to September. These months also present the period of greatest electricity demand due largely to increased usage of air conditioning. This presents a valuable opportunity to potentially curtail electricity use in demand response scenarios by shifting load off of the peak periods of energy price. Toward this aim, this paper is our attempt to estimate the economic savings, feasibility, and any potential user impact from full or partial cluster shutdown during periods of increased energy demand.

C.1.1 Alternative Demand Response Options in Data Centers

Although we focus on load shifting for our study, we wish to point out prior work on alternative strategies for demand response.

Facility changes A study by Lawrence Berkeley National Laboratory (LBNL) found that 5% of the data center load can typically be shed in 5 minutes and 10% of the load can be shed in 15 minutes without changes to how the IT workload is handled, i.e., via temperature adjustment and other building management approaches[34]. Most data centers have local power due to a backup generator, which could also be used to absorb some load during peak time [58]. More recently, methods of energy storage have been proposed[73] in which UPS batteries are re-purposed for provisioning during periods of peak demand in addition to their primary purpose of backup power. However, these methods all entail manual intervention, with close monitoring and control.

Power capping is a strategy by which to run data center equipment within a set of constraints which assume the electricity draw for the data center as a whole cannot grow any larger. Some examples of this include and turning off or constraining CPU/GPU power consumption to values below the CPU Thermal Design Power (TDP) value, which requires less voltage. Many equipment manufacturers - including IBM, Intel and AMD - have implemented power capping technology that can be monitored at the processor level and applied at the rack level. One approach to power capping is Dynamic Voltage/Frequency Scaling (DVFS). However, as noted by Roundtree[79], no machine in the Top 500 list of supercomputers makes use of DVFS to save power or energy since the performance impact and the amount of power and energy saved was highly application dependent. Power capping doesnt necessarily equate to energy efficiency nor cost savings.

Schedulers Zhou et al[103] present a method for power-aware scheduling by using a combination of a scheduling window and 0-1 knapsack model, which shows promise. However, since SLURM is our scheduler, we decided to focus solely on SLURM. Bodas et al[7] demonstrate an integration of power capping into a power-aware scheduler, with the overall goal of maintaining average system power within a budget. Their work demonstrates that SLURMs auto mode can be used to maximize available power.

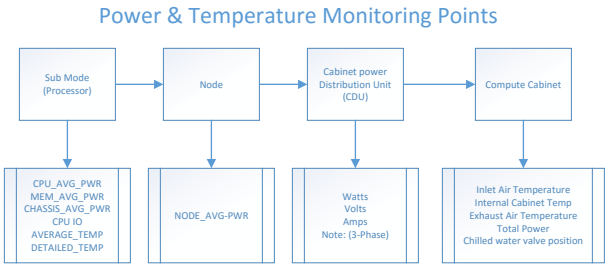


Figure C.2: Energy monitoring framework.

C.2 Problem Statement

Can load shifting of high performance computing tasks save universities money in energy demand response scenarios? To explore the relative costs of implementing load shifting in demand response scenarios, we have expressed the problem by modeling total dollar cost. We wish to use this framework to explore the optimization of price in the presence of various data center usage statistics and price fluctuation schemes.

C.2.1 Modeling Energy Costs

We generate a model total cost function composed of a fixed cost for purchasing and maintaining nodes plus a variable cost dependent on data center power usage and energy prices. We wish to minimize the cost function

$$C = p_n T n_{max} + \int_0^T dt \cdot p(t) \left(n(t)u(t) + u_w \frac{\Delta n(t)}{\Delta t} \right)$$

where $p(t)$ is the price of power at time t, $0 < n(t) < n_{max}$ is the number of running nodes, $u(t)$ is the average node power usage, u_w is the wasted power from turning on a node, p_n is the amortized lifetime cost of purchasing a node, and n_{max} is the total number of nodes in the cluster.

Based on our cluster usage statistics, we approximate that compute cycles are roughly interchangeable and that the main determiner of power usage is simply the CPU utilization of the node. In this case, node power usage takes the form

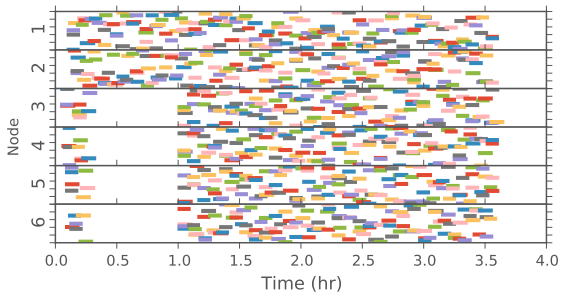


Figure C.3: Diagram of job scheduling during a four node temporary shutdown experiment. Each colored rectangle displays the execution time of a single LAMPPS test job running for approximately 5 minutes.

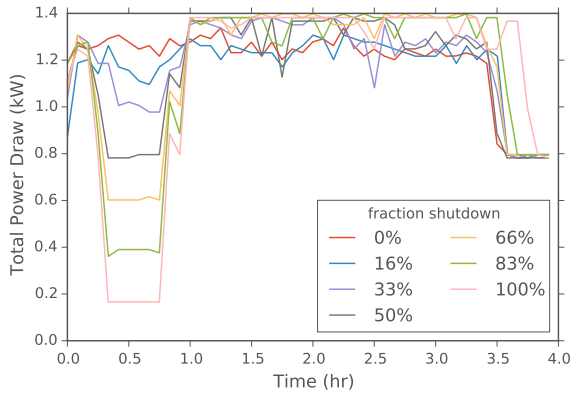


Figure C.4: Total power consumed during experiments where variable numbers of machines were shut down during simulated peak pricing.

$$u(t) = u_0 + u_v \cdot r(t)$$

where $0 < r(t) < 1$ is the fraction of CPU usage, u_0 is the cost of an idling node and u_v is the variable cost for doing r work on a machine.

We wish to minimize the cost function C subject to the constraint that the sum of the submitted CPU cycles, S, are all completed after a period T.

$$\int_0^T dt \cdot n(t) r(t) = S$$

C.2.2 Response to a Temporary Price Spike

In particular, we wish to use this framework to determine how to run our data center in the situation where every T days, we see a price spike from p_0 to p_s, lasting time period t_s. This condition is highly similar to the one facility managers face when utility provides impose usage tariffs during peak energy demand periods.

In this situation, the number of running machines will change stepwise between a high number of running machines, $n_H = n_{max}$, and a low number of running machines, n_L, and a high and low CPU utilization $r_H = 1$, r_L, with a corresponding u_H and u_L as defined above. The high usage will occur during the cheap energy supply, and the low usage will occur during the price spike. Therefore we can rewrite our cost function as

$$C = p_n n_H T + p_0(u_0 + u_v)n_H(T - t_s) + p_s(u_0 + u_v r_L)n_L t_s + p_0 u_w(n_H - n_L) \quad \text{(C.1)}$$

with the constraint

$$n_H(T - t_s) + n_L r_L t_s = S \quad \text{(C.2)}$$

Inserting the constraint into our cost function to replace r_L yields

$$C = p_s u_v S + n_L \cdot (p_s u_0 t_s - p_0 u_w t_w) + n_H \cdot (p_n T + p_0 u_w t_w - (\Delta p u_v - p_0 u_0)(T - t_s)) \quad \text{(C.3)}$$

where we've introduce the price difference, $\Delta p = p_s - p_0$.

We can analyze the change in costs as a function of n_L and n_H to determine the optimal cluster setup for known variables, t_s, p_s, p_n, p_0, u_0, u_v, and u_w.

From this analysis, whenever $p_s u_0 t_s < p_0 u_w t_w$, the cost of powering off nodes exceeds the cost of running those nodes idle so $n_L = n_H$ and $r_L = S/n_H t_s - (T - t_s)/t_s$. Otherwise powering off nodes saves money so the nodes that remain on run at full capacity $r_L = 1$, and n_L is minimized subject to constraints giving $n_L = S/t_s -$

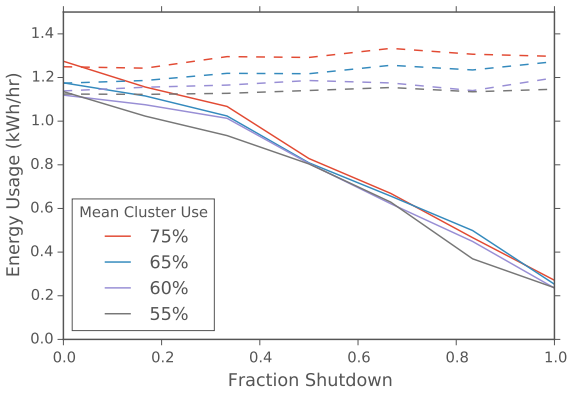


Figure C.5: Energy usage of test cluster during partial shutdown experiments. Solid lines indicate power usage during the shutdown, while dashed lines indicate power usage after returning to full operation.

$n_H(T - t_s)/t_s$.

If we can freely choose n_H to optimize cost, then whenever $(\Delta p u_v - p_0 u_0)(T - t_s) > p_n T + min(p_0 u_w t_w, p_s u_0 t_s)$, we would increase n_H (i.e. buy more machines) until all the work is done during the cheap energy period. Therefore $n_H = S/(T - t_s)$ and either n_L or r_L is 0. Otherwise, the cost of new machines is more than any cost savings achieved from exploiting the price difference, and we would simply ignore the price spike (i.e. set $n_H = n_L = S/T$ and $r_L = 1$).

C.3 EDEALS: Electricity Demand-response Easy Adjusted Load Shifting

For a data center manager to use the above model to determine their cost savings, they must collect and analyze usage and power data on their system. We have built a cluster data processing pipeline, EDEALS, to assess the magnitude of potential savings available from a full or partial cluster shutdown. We combine SLURM job scheduling, node level IMM power and usage metrics, and cabinet level CDU measurements to determine the optimum magnitude of demand response cluster shutdowns.

Here we describe our data center instrumentation, so that we ensure accurate

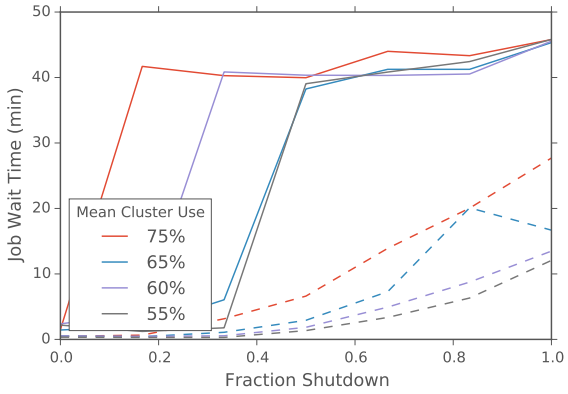


Figure C.6: Maximum (solid) and mean (dashed) job wait times during partial shutdown experiments.

measurements of performance of the workload management system and HPC cluster alone without the influence of extraneous components. Since our focus is the HPC cluster and SLURM manager, we need to ensure those components alone affect the reduced data center utility bill. As depicted in Figure C.2, we take measurements at the core, node, rack, and cabinet level. These data are combined to detect power losses at each step and to determine the correlation between the power measurements at the machine level and the true power draw at the facility level.

Combining this data with electricity pricing statistics from utility managers allows system administrators to determine when and by how much to reduce their power usage to save money. We have built a set of scripts particular to our system to implement machine level power down in response to predicted energy peaks. At the end of the peak energy period the machines automatically reboot and are added by to SLURMs available server pool. At this time, these power cycling scripts are manually executed by system administrators after evaluation of the likelihood of near-term energy demand peaks. However, as more data centers begin to implement smart metering, it will become possible to automate load shifting in response to real-time energy pricing indicators. We look forward to continuing this as future work.

C.4 Small-Scale Evaluation of EDEALS

To test our load shifting scheme, we launched a series of small-scale experiments on a 6 machine test cluster using SLURM batch management system to schedule jobs. We wished to compare the energy savings and job wait times during a full or partial cluster shutdown in response to an energy price spike.

C.4.1 Experimental Setup

We measured the total energy use over a 3.5 hour window of which the first 30 minutes comprised a partial cluster shutdown, followed by a 15 minute powerup routine. We explored the impact of shutting down between one and all six nodes during the 30 minute window. The shutdown was carried out by fully powering nodes off. We compared this to the energy usage without the partial shutdown.

Identical sample jobs were submitted to the cluster via SLURM scheduler at a constant rate to set the average cluster usage to approximately 55%, 60%, 65% and 75% capacity. We used custom state control commands to set the power states of individual machines in the test cluster. The SLURM scheduler automatically shifted queued jobs to run on the available machines, as shown in the example job schedule displayed Figure **??** from a four node shutdown experiment. We used our EDEALS data analysis pipeline to measure the changes in energy usage and job wait time in the queue.

C.4.2 Evaluation of Model Parameters

Importantly, EDEALS allowed us to determine appropriate power parameters, u_0, u_v, and u_w for both our test rack as well as a larger partition of the University of Chicago's Midway production cluster. Figure C.7 shows the measured relationship between CPU utilization and energy usage as determined from the machine level IPMI metric data.

To account for losses not measured at the IPMI level, we compare the sum IPMI power usage to the rack level power monitoring. This comparison revealed a correction

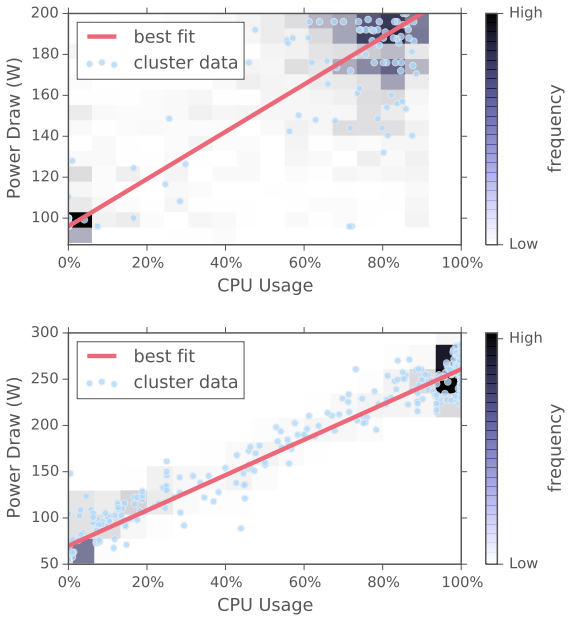


Figure C.7: Power data for test cluster (top) and production cluster (bottom) nodes in presence of variable usage. The slope and intercept of the line are used to determine u_v and u_0 respectively.

factor of 1.25 between the IPMI measurement and the total rack level energy draw. Using this corrected model, we were able to predict power consumption at the CDU level via CPU utilization under variable scheduler loads.

C.4.3 Relative Energy Savings and Max Wait Times

Our test cluster provided us with an important baseline in determining the effectiveness of a partial shutdown in reducing energy usage. As shown in Figure C.4, the total power draw from the test cluster was reduced dramatically during the shutdown period, and then returned to its baseline level.

These experiments were repeated with different job submission rates such that the average CPU usage varied from 55% to 75%. As shown in Figure C.5, the partial shutdowns reduced the total energy usage as measured at the CDU level. Not surprisingly, the power usage during cluster shutdown for all usage levels converged to roughly the same value at the point where all remaining operational machines reached

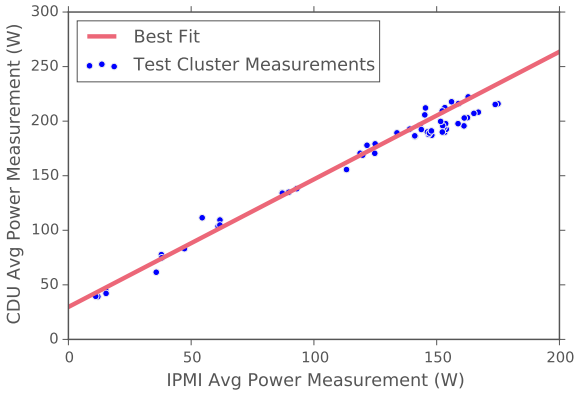


Figure C.8: Comparison between node level IPMI measurements and rack level CDU measurements. Best fit shows the model relationship used to convert IPMI data to estimated total power draw.

full capacity. Interestingly, the energy savings did not appear to be perfectly directly proportional to the fraction shut down. In particular, there was residual energy use associated with our machine's low power state even when the cluster was entirely shut down.

We also measured the difference between job submission and start time, as depicted in Figure C.6. As one would expect, both mean and max wait times increased as the shutdown fraction grew and the effect was more pronounced when the cluster usage was higher. However, we were pleasantly surprised to find that max wait times topped out at 45 minutes, which was the duration of the entire cluster down period. This indicates that SLURM doesn't add too much additional overhead, and therefore, the worst-case user wait times would not exceed the total period that the cluster was shut down.

C.5 Conclusion: Implication for An Operational HPC Datacenter

In an HPC datacenter, the variable cost to supply electricity to a facility can be decomposed into both a nominal cost per kilowatt-hour and a procurement cost from

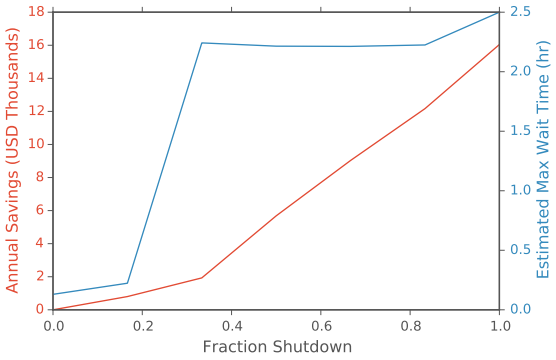


Figure C.9: Estimated savings from partial cluster shutdowns.

T	t_s	p_0	p_s
360 hr	2 hr	\$0.03/kWh	\$6.5/kWh

Table C.1: Model parameters estimated for medium scale HPC datacenter.

the supplier. Some suppliers impose a substantial procurement tariff based on electricity usage during the five, two hour long periods of highest demand in a year. In this scenario, the savings of load shedding can be orders of magnitude higher than the nominal price per kilowatt-hour. We estimate that by curtailing 1MW, 8 times per year, we can expect an annual savings of approximately \$100K, which in our case was roughly a 7

Combining these values with the power usage measurements from our production cluster, we can extrapolate the yearly savings based on fractional shutdowns of the data center. In addition, using the wait time statistics from our test cluster we can also estimate the worst-case impact on user wait-times that these cluster shutdowns will incur. We display this information in Figure C.9, as a function of the fraction of the cluster that we would be theoretically willing to shut down.

C.6 Acknowledgments

Special thanks goes out to Brandon for all his work getting our test cluster set up as well as for his useful input on machine pricing information. We also wish to thank

Matt Beach for his invaluable knowledge on energy pricing mechanisms and university power plans. Finally, we wish to thank Dr. Birali Runesha for his support in carrying out this project.

C.7 Availability

On our Github, we have provided all data collection scripts, analysis routines, and experimental setups, as well as detailed calculations and optimization methods for other price models.

`https://github.com/rcc-uchicago/datacenter`

APPENDIX D
SOURCE CODE AND DOCUMENTATION

D.1 Finding Source Code Online

Source code and up-to-date documentation are available online for the projects described in this dissertation.

- https://github.com/wmcfadden/smPress
- https://github.com/wmcfadden/activnet
- https://github.com/munrolab/pulse-reaction-dynamics

D.1.1 Instructions for running MATLAB pre-compiled code

D.1.2 Simulation Source

REFERENCES

[1] Jose Alvarado, Michael Sheinman, Abhinav Sharma, Fred C. MacKintosh, and Gijsje H. Koenderink. Molecular motors robustly drive active gels to a critically connected state. *Nat Phys*, 9(9):591–597, 09 2013.

[2] Shiladitya Banerjee, M. Cristina Marchetti, and Kristian Müller-Nedebock. Motor-driven dynamics of cytoskeletal filaments in motility assays. *Phys. Rev. E*, 84:011914, Jul 2011.

[3] Andreas R. Bausch, Florian Ziemann, Alexei A. Boulbitch, Ken Jacobson, and Erich Sackmann. Local measurements of viscoelastic parameters of adherent cell surfaces by magnetic bead microrheometry. *Biophysical Journal*, 75(4):2038 – 2049, 1998.

[4] Martin Behrndt, Guillaume Salbreux, Pedro Campinho, Robert Hauschild, Felix Oswald, Julia Roensch, Stephan W. Grill, and Carl-Philipp Heisenberg. Forces driving epithelial spreading in zebrafish gastrulation. *Science*, 338(6104):257–260, 2012.

[5] Poul M. Bendix, Gijsje H. Koenderink, Damien Cuvelier, Zvonimir Dogic, Bernard N. Koeleman, William M. Brieher, Christine M. Field, L. Mahadevan, and David A. Weitz. A quantitative analysis of contractility in active cytoskeletal protein networks. *Biophysical Journal*, 94(8):3126 – 3136, 2008.

[6] Hélène A Benink, Craig A Mandato, and William M Bement. Analysis of cortical flow models in vivo. *Molecular Biology of the Cell*, 11(8):2553–2563, 08 2000.

[7] Deva Bodas, Justin Song, Murali Rajappa, and Andy Hoffman. Simple power-aware scheduler to limit power consumption by hpc system within a budget. In *Proceedings of the 2Nd International Workshop on Energy Efficient Supercomputing*, E2SC '14, pages 21–30, Piscataway, NJ, USA, 2014. IEEE Press.

[8] Justin S. Bois, Frank Jülicher, and Stephan W. Grill. Pattern formation in active fluids. *Phys. Rev. Lett.*, 106:028103, Jan 2011.

[9] D Bray and JG White. Cortical flow in animal cells. *Science*, 239(4842):883–888, 1988.

[10] C. P. Broedersz, C. Storm, and F. C. MacKintosh. Effective-medium approach for stiff polymer networks with flexible cross-links. *Phys. Rev. E*, 79:061914, Jun 2009.

[11] Chase P. Broedersz, Martin Depken, Norman Y. Yao, Martin R. Pollak, David A. Weitz, and Frederick C. MacKintosh. Cross-link-governed dynamics of biopolymer networks. *Phys. Rev. Lett.*, 105:238101, Nov 2010.

[12] P. Broedersz, C. and C. MacKintosh, F.˙ Modeling semiflexible polymer networks. *Rev. Mod. Phys.*, 86:995–1036, Jul 2014.

[13] Anders E Carlsson. Actin dynamics: From nanoscale to microscale. *Annual review of biophysics*, 39:91–110, 06 2010.

[14] Preethi L. Chandran and Mohammad R. K. Mofrad. Averaged implicit hydrodynamic model of semiflexible filaments. *Phys. Rev. E*, 81:031920, Mar 2010.

[15] Priyamvada Chugh, Andrew G. Clark, Matthew B. Smith, Davide A. D. Cassani, Guillaume Charras, Guillaume Salbreux, and Ewa K. Paluch. Nanoscale organization of the actomyosin cortex during the cell cycle. *Biophysical Journal*, 110(3):198a, 2016/07/18 2016.

[16] Martine Coué, Stephen L. Brenner, Ilan Spector, and Edward D. Korn. Inhibition of actin polymerization by latrunculin a. *FEBS Letters*, 213(2):316–318, 1987.

[17] Christian J. Cyron, Kei W. Müller, Andreas R. Bausch, and Wolfgang A. Wall. Micromechanical simulations of biopolymer networks with finite elements. *Journal of Computational Physics*, 244(0):236 – 251, 2013. Multi-scale Modeling and Simulation of Biological Systems.

[18] Enrique M De La Cruz. How cofilin severs an actin filament. *Biophysical reviews*, 1(2):51–59, 05 2009.

[19] Enrique M. De La Cruz and Margaret L. Gardel. Actin mechanics and fragmentation. *Journal of Biological Chemistry*, 290(28):17137–17144, 07 2015.

[20] Pierre Delforge and Josh Whitney. Data center efficiency assessment: Scaing up energy efficiency across the data center industry: Evaluating key drivers and barriers. Technical report, National Resources Defense Council, August 2014.

[21] Kai Dierkes, Angughali Sumi, Jérôme Solon, and Guillaume Salbreux. Spontaneous oscillations of elastic contractile materials with turnover. *Phys. Rev. Lett.*, 113:148102, Oct 2014.

[22] M. Doi and S. F. Edwards. *The Theory of Polymer Dynamics*. Oxford University Press, USA, November 1986.

[23] Fangming Du, John E. Fischer, and Karen I. Winey. Effect of nanotube alignment on percolation conductivity in carbon nanotube/polymer composites. *Phys. Rev. B*, 72:121404, Sep 2005.

[24] Hajer Ennomani, Gaëlle Letort, Christophe Guérin, Jean-Louis Martiel, Wenxiang Cao, François Nédélec, Enrique M. De La Cruz, Manuel Théry, and Laurent Blanchoin. Architecture and connectivity govern actin network contractility. *Current Biology*, 26(5):616 – 626, 2016.

[25] E Evans and A Yeung. Apparent viscosity and cortical tension of blood granulocytes determined by micropipet aspiration. *Biophysical Journal*, 56(1):151–160, 07 1989.

[26] A. E. Filippov, J. Klafter, and M. Urbakh. Friction through dynamical formation and rupture of molecular bonds. *Phys. Rev. Lett.*, 92:135503, Mar 2004.

[27] Daniel A. Fletcher and R. Dyche Mullins. Cell mechanics and the cytoskeleton. *Nature*, 463(7280):485–492, 01 2010.

[28] Marco Fritzsche, Christoph Erlenkämper, Emad Moeendarbary, Guillaume Charras, and Karsten Kruse. Actin kinetics shapes cortical network structure and mechanics. *Science Advances*, 2(4), 2016.

[29] Marco Fritzsche, Alexandre Lewalle, Tom Duke, Karsten Kruse, and Guillaume Charras. Analysis of turnover dynamics of the submembranous actin cortex. *Molecular Biology of the Cell*, 24(6):757–767, 03 2013.

[30] Marco Fritzsche, Alexandre Lewalle, Tom Duke, Karsten Kruse, and Guillaume Charras. Analysis of turnover dynamics of the submembranous actin cortex. *Molecular Biology of the Cell*, 24(6):757–767, 2013.

[31] M. L. Gardel, F. Nakamura, J. Hartwig, J. C. Crocker, T. P. Stossel, and D. A. Weitz. Stress-dependent elasticity of composite actin networks as a model for cell behavior. *Phys. Rev. Lett.*, 96:088102, Mar 2006.

[32] M. L. Gardel, F. Nakamura, J. H. Hartwig, J. C. Crocker, T. P. Stossel, and D. A. Weitz. Prestressed f-actin networks cross-linked by hinged filamins replicate mechanical properties of cells. *Proceedings of the National Academy of Sciences of the United States of America*, 103(6):1762–1767, 2006.

[33] M. L. Gardel, J. H. Shin, F. C. MacKintosh, L. Mahadevan, P. Matsudaira, and D. A. Weitz. Elastic behavior of cross-linked and bundled actin networks. *Science*, 304(5675):1301–1305, 2004.

[34] Girish Ghatikar, Venkata Ganti, Nance Matson, and Mary Ann Piette. Demand response opportunities and enabling technologies for data centers: Findings from field studies. 2012.

[35] Nicole Gorfinkiel and Guy B Blanchard. Dynamics of actomyosin contractile activity during epithelial morphogenesis. *Current Opinion in Cell Biology*, 23(5):531 – 539, 2011. Cell-to-cell contact and extracellular matrix.

[36] Minakshi Guha, Mian Zhou, and Yu-li Wang. Cortical actin turnover during cytokinesis requires myosin ii. *Current Biology*, 15(8):732–736, 2016/12/15 2016.

[37] David A. Head, Alex J. Levine, and F. C. MacKintosh. Deformation of cross-linked semiflexible polymer networks. *Phys. Rev. Lett.*, 91:108102, Sep 2003.

[38] Claus Heussinger, Boris Schaefer, and Erwin Frey. Nonaffine rubber elasticity for stiff polymer networks. *Phys. Rev. E*, 76:031906, Sep 2007.

[39] T. Hiraiwa and G. Salbreux. Role of turn-over in active stress generation in a filament network. *ArXiv e-prints*, July 2015.

[40] S N Hird and J G White. Cortical and cytoplasmic flow polarity in early embryonic cells of caenorhabditis elegans. *The Journal of Cell Biology*, 121(6):1343–1355, 1993.

[41] Robert M Hochmuth. Micropipette aspiration of living cells. *Journal of Biomechanics*, 33(1):15 – 22, 2000.

[42] Jonathon Howard, Stephan W. Grill, and Justin S. Bois. Turing's next steps: the mechanochemical basis of morphogenesis. *Nat Rev Mol Cell Biol*, 12(6):392–398, Jun 2011.

[43] L W Janson, J Kolega, and D L Taylor. Modulation of contraction by gelation/solation in a reconstituted motile model. *The Journal of Cell Biology*, 114(5):1005–1015, 1991.

[44] K. E. Kasza, C. P. Broedersz, G. H. Koenderink, Y. C. Lin, W. Messner, E. A. Millman, F. Nakamura, T. P. Stossel, F. C. MacKintosh, and D. A. Weitz. Actin filament length tunes elasticity of flexibly cross-linked actin networks. *Biophysical Journal*, 99(4):1091–1100, 2015/03/12.

[45] Kinneret Keren, Patricia T. Yam, Anika Kinkhabwala, Alex Mogilner, and Julie A. Theriot. Intracellular fluid flow in rapidly moving cells. *Nat Cell Biol*, 11(10):1219–1224, 10 2009.

[46] Taeyoon Kim, Margaret L. Gardel, and Ed Munro. Determinants of fluid-like behavior and effective viscosity in cross-linked actin networks. *Biophysical Journal*, 106(3):526 – 534, 2014.

[47] Taeyoon Kim, Wonmuk Hwang, and Roger D Kamm. Dynamic role of cross-linking proteins in actin rheology. *Biophysical Journal*, 101(7):1597–1603, 10 2011.

[48] Gijsje H Koenderink, Zvonimir Dogic, Fumihiko Nakamura, Poul M Bendix, Frederick C MacKintosh, John H Hartwig, Thomas P Stossel, and David A Weitz. An active biopolymer network controlled by molecular motors. *Proceedings of the National Academy of Sciences of the United States of America*, 106(36):15192–15197, 09 2009.

[49] Frank PL Lai, Malgorzata Szczodrak, Jennifer Block, Jan Faix, Dennis Breitsprecher, Hans G Mannherz, Theresia EB Stradal, Graham A Dunn, J Victor Small, and Klemens Rottner. Arp2/3 complex interactions and actin network turnover in lamellipodia. *The EMBO Journal*, 27(7):982–992, 04 2008.

[50] Martin Lenz. Geometrical origins of contractility in disordered actomyosin networks. *Phys. Rev. X*, 4:041002, Oct 2014.

[51] Martin Lenz, Margaret L Gardel, and Aaron R Dinner. Requirements for contractility in disordered cytoskeletal bundles. *New Journal of Physics*, 14(3):033037, 2012.

[52] O. Lieleg and A. R. Bausch. Cross-linker unbinding and self-similarity in bundled cytoskeletal networks. *Phys. Rev. Lett.*, 99:158105, Oct 2007.

[53] O. Lieleg, M. M. A. E. Claessens, Y. Luan, and A. R. Bausch. Transient binding and dissipation in cross-linked actin networks. *Phys. Rev. Lett.*, 101:108101, Sep 2008.

[54] O. Lieleg, K. M. Schmoller, M. M. A. E. Claessens, and A. R. Bausch. Cytoskeletal polymer networks: Viscoelastic properties are determined by the microscopic interaction potential of cross-links. *Biophysical Journal*, 96(11):4725–4732, 6 2009.

[55] Oliver Lieleg, Mireille M. A. E. Claessens, and Andreas R. Bausch. Structure and dynamics of cross-linked actin networks. *Soft Matter*, 6:218–225, 2010.

[56] Yi-Chia Lin, Chase P. Broedersz, Amy C. Rowat, Tatjana Wedig, Harald Herrmann, Frederick C. MacKintosh, and David A. Weitz. Divalent cations crosslink vimentin intermediate filament tail domains to regulate network mechanics. *Journal of Molecular Biology*, 399(4):637 – 644, 2010.

[57] J. Liu, G. H. Koenderink, K. E. Kasza, F. C. MacKintosh, and D. A. Weitz. Visualizing the strain field in semiflexible polymer networks: Strain fluctuations and nonlinear rheology of f-actin gels. *Phys. Rev. Lett.*, 98:198304, May 2007.

[58] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *SIGMETRICS Perform. Eval. Rev.*, 41(1):341–342, June 2013.

[59] Samuel J. Lord, Hsiao-lu D. Lee, and W. E. Moerner. Single-molecule spectroscopy and imaging of biomolecules in living cells. *Analytical Chemistry*, 82(6):2192–2203, 2010. PMID: 20163145.

[60] Michael Mak, Taeyoon Kim, Muhammad H. Zaman, and Roger D. Kamm. Multiscale mechanobiology: computational models for integrating molecules to multicellular systems. *Integr Biol (Camb)*, 7(10):1093–1108, Oct 2015. 26019013[pmid].

[61] Michael Mak, Muhammad H. Zaman, Roger D. Kamm, and Taeyoon Kim. Interplay of active processes modulates tension and drives phase transition in self-renewing, motor-driven cytoskeletal networks. *Nat Commun*, 7, 01 2016.

[62] M. C. Marchetti, J. F. Joanny, S. Ramaswamy, T. B. Liverpool, J. Prost, Madan Rao, and R. Aditi Simha. Hydrodynamics of soft active matter. *Rev. Mod. Phys.*, 85:1143–1189, Jul 2013.

[63] John F. Marko and Eric D. Siggia. Stretching dna. *Macromolecules*, 28(26):8759–8770, 1995.

[64] Mirjam Mayer, Martin Depken, Justin S. Bois, Frank Julicher, and Stephan W. Grill. Anisotropies in cortical tension reveal the physical basis of polarizing cortical flows. *Nature*, 467(7315):617–621, 09 2010.

[65] N.G. McCrum, C.P. Buckley, and C.B. Bucknall. *Principles of Polymer Engineering*. Oxford science publications. Oxford University Press, 1997.

[66] David C. Morse. Viscoelasticity of concentrated isotropic solutions of semiflexible polymers. 2. linear response. *Macromolecules*, 31(20):7044–7067, 1998.

[67] Kei W. Müller, Robijn F. Bruinsma, Oliver Lieleg, Andreas R. Bausch, Wolfgang A. Wall, and Alex J. Levine. Rheology of semiflexible bundle networks with transient linkers. *Phys. Rev. Lett.*, 112:238102, Jun 2014.

[68] Edwin Munro, Jeremy Nance, and James R. Priess. Cortical flows powered by asymmetrical contraction transport {PAR} proteins to establish and maintain

anterior-posterior polarity in the early c. elegans embryo. *Developmental Cell*, 7(3):413 – 424, 2004.

[69] Michael Murrell and Margaret L. Gardel. Actomyosin sliding is attenuated in contractile biomimetic cortices. *Molecular Biology of the Cell*, 25(12):1845–1853, 2014.

[70] Michael Murrell, Patrick W. Oakes, Martin Lenz, and Margaret L. Gardel. Forcing cells into shape: the mechanics of actomyosin contractility. *Nat Rev Mol Cell Biol*, 16(8):486–498, 08 2015.

[71] Michael P. Murrell and Margaret L. Gardel. F-actin buckling coordinates contractility and severing in a biomimetic actomyosin cortex. *Proceedings of the National Academy of Sciences*, 109(51):20820–20825, 2012.

[72] Kausalya Murthy and Patricia Wadsworth. Myosin-ii-dependent localization and dynamics of f-actin during cytokinesis. *Current Biology*, 15(8):724–731, 2016/12/15 2016.

[73] Lyswarya Narayanan, Di Wang, Abdullah-Al Mamun, Anand Sivasubramaniam, and Hosam K. Fathy. Should we dual-purpose energy storage in datacenters for power backup and demand response? In *6th Workshop on Power-Aware Computing and Systems (HotPower 14)*, Broomfield, CO, 2014. USENIX Association.

[74] F. J. Nedelec, T. Surrey, A. C. Maggs, and S. Leibler. Self-organization of microtubules and motors. *Nature*, 389(6648):305–308, 09 1997.

[75] Matteo Rauzi, Pierre-Francois Lenne, and Thomas Lecuit. Planar polarized actomyosin contractile flows control epithelial junction remodelling. *Nature*, 468(7327):1110–1114, Dec 2010.

[76] Anne-Cécile Reymann, Rajaa Boujemaa-Paterski, Jean-Louis Martiel, Christophe Guérin, Wenxiang Cao, Harvey F. Chin, Enrique M. De La Cruz, Manuel Théry, and Laurent Blanchoin. Actin network architecture can determine myosin motor activity. *Science*, 336(6086):1310–1314, 2012.

[77] Francois B Robin, William M McFadden, Baixue Yao, and Edwin M Munro. Single-molecule analysis of cell surface dynamics in caenorhabditis elegans embryos. *Nat Meth*, 11(6):677–682, 06 2014.

[78] Francois B. Robin, William M. McFadden, Baixue Yao, and Edwin M. Munro. Single-molecule analysis of cell surface dynamics in caenorhabditis elegans embryos. *Nat Meth*, 11(6):677–682, Jun 2014. Article.

[79] Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. Beyond dvfs: A first look at performance under a hardware-enforced power bound. *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, 0:947–953, 2012.

[80] G. Salbreux, J. Prost, and J. F. Joanny. Hydrodynamics of cellular cortical flows and the formation of contractile rings. *Phys. Rev. Lett.*, 103:058102, Jul 2009.

[81] Guillaume Salbreux, Guillaume Charras, and Ewa Paluch. Actin cortex mechanics and cellular morphogenesis. *Trends in Cell Biology*, 22(10):536 – 545, 2012.

[82] Guillaume Salbreux, Guillaume Charras, and Ewa Paluch. Actin cortex mechanics and cellular morphogenesis. *Trends in Cell Biology*, 22(10):536 – 545, 2012.

[83] Tim Sanchez, Daniel T. N. Chen, Stephen J. DeCamp, Michael Heymann, and Zvonimir Dogic. Spontaneous motion in hierarchically assembled active matter. *Nature*, 491(7424):431–434, 11 2012.

[84] Evan Spruijt, Joris Sprakel, Marc Lemmers, Martien A. Cohen Stuart, and Jasper van der Gucht. Relaxation dynamics at different time scales in electrostatic complexes: Time-salt superposition. *Phys. Rev. Lett.*, 105:208301, Nov 2010.

[85] Dimitrije Stamenovic. Cell mechanics: Two regimes, maybe three? *Nat Mater*, 5(8):597–598, 08 2006.

[86] Cornelis Storm, Jennifer J. Pastore, F. C. MacKintosh, T. C. Lubensky, and Paul A. Janmey. Nonlinear elasticity in biological gels. *Nature*, 435(7039):191–194, 05 2005.

[87] Thomas Surrey, François Nédélec, Stanislas Leibler, and Eric Karsenti. Physical properties determining self-organization of motors and microtubules. *Science*, 292(5519):1167–1171, 2001.

[88] R. Tharmann, M. M. A. E. Claessens, and A. R. Bausch. Viscoelasticity of isotropically cross-linked actin networks. *Phys. Rev. Lett.*, 98:088103, Feb 2007.

[89] Julie A. Theriot and Timothy J. Mitchison. Actin microfilament dynamics in locomoting cells. *Nature*, 352(6331):126–131, Jul 1991.

[90] Hervé Turlier, Basile Audoly, Jacques Prost, and Jean-François Joanny. Furrow constriction in animal cell cytokinesis. *Biophysical Journal*, 106(1):114 – 123, 2014.

[91] Michael J. Unterberger and Gerhard A. Holzapfel. Advances in the mechanical modeling of filamentous actin and its cross-linked networks on multiple scales. *Biomechanics and Modeling in Mechanobiology*, 13(6):1155–1174, 2014.

[92] David Van Goor, Callen Hyland, Andrew W Schaefer, and Paul Forscher. The role of actin turnover in retrograde actin network flow in neuronal growth cones. *PLoS ONE*, 7(2):e30959, 2012.

[93] Andrea Vanossi, Nicola Manini, Michael Urbakh, Stefano Zapperi, and Erio Tosatti. *Colloquium* : Modeling friction: From nanoscale to mesoscale. *Rev. Mod. Phys.*, 85:529–552, Apr 2013.

[94] D.H. Wachsstock, W.H. Schwarz, and T.D. Pollard. Cross-linker dynamics determine the mechanical properties of actin gels. *Biophysical Journal*, 66(3, Part 1):801 – 809, 1994.

[95] Andrew Ward, Feodor Hilitski, Walter Schwenger, David Welch, A. W. C. Lau, Vincenzo Vitelli, L. Mahadevan, and Zvonimir Dogic. Solid friction between soft filaments. *Nat Mater*, advance online publication:–, 03 2015.

[96] Sabine M. Volkmer Ward, Astrid Weins, Martin R. Pollak, and David A. Weitz. Dynamic viscoelasticity of actin cross-linked with wild-type and disease-causing mutant alpha-actinin-4. *Biophysical Journal*, 95(10):4915 – 4923, 2008.

[97] Naoki Watanabe and Timothy J. Mitchison. Single-molecule speckle analysis of actin filament turnover in lamellipodia. *Science*, 295(5557):1083–1086, 2002.

[98] A. Wierman, Zhenhua Liu, I. Liu, and H. Mohsenian-Rad. Opportunities and challenges for data center demand response. In *Green Computing Conference (IGCC), 2014 International*, pages 1–10, Nov 2014.

[99] Jan Wilhelm and Erwin Frey. Elasticity of stiff polymer networks. *Phys. Rev. Lett.*, 91:108103, Sep 2003.

[100] Cyrus A. Wilson, Mark A. Tsuchida, Greg M. Allen, Erin L. Barnhart, Kathryn T. Applegate, Patricia T. Yam, Lin Ji, Kinneret Keren, Gaudenz Danuser, and Julie A. Theriot. Myosin ii contributes to cell-scale actin network treadmilling through network disassembly. *Nature*, 465(7296):373–377, 05 2010.

[101] M. Wyart, H. Liang, A. Kabla, and L. Mahadevan. Elasticity of floppy and stiff random networks. *Phys. Rev. Lett.*, 101:215501, Nov 2008.

[102] Norman Y. Yao, Daniel J. Becker, Chase P. Broedersz, Martin Depken, Frederick C. MacKintosh, Martin R. Pollak, and David A. Weitz. Nonlinear viscoelasticity of actin transiently cross-linked with mutant alpha-actinin-4. *Journal of Molecular Biology*, 411(5):1062 – 1071, 2011.

[103] Zhou Zhou, Zhiling Lan, Wei Tang, and Narayan Desai. Reducing energy costs for ibm blue gene/p via power-aware job scheduling. In Narayan Desai and Walfredo Cirne, editors, *Job Scheduling Strategies for Parallel Processing*, volume 8429 of *Lecture Notes in Computer Science*, pages 96–115. Springer Berlin Heidelberg, 2014.

[104] Alexander Zumdieck, Karsten Kruse, Henrik Bringmann, Anthony A. Hyman, and Frank Jülicher. Stress generation and filament turnover during actin ring constriction. *PLoS ONE*, 2(8):e696, 08 2007.