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
SEPT 2016

To Claire

ABSTRACT

Blar Blar

ACKNOWLEDGEMENTS

I'd like to thank Patrick McCall, Pete Dahlberg, Shiladitya Banerjee and a bunch of other people.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Biological context of cortical flows

Cortical flow is a fundamental and ubiquitous form of cellular deformation that underlies cell polarization, cell division, cell crawling and multicellular tissue morphogenesis[7, 29]. These flows arise within the actomyosin cortex, a thin layer of cross-linked actin filaments and myosin motors that lies just beneath the plasma membrane [62]. The active forces that drive cortical flows are thought to be generated by myosin motors pulling against individual actin filaments [54]. These forces must be integrated within cross-linked networks to build macroscopic contractile stress. At the same time, cross-linked networks resist deformation and this resistance must be dissipated by network remodeling to allow macroscopic network deformation and flow. How force production and dissipation depend on motor activity, network architecture and remodeling remains poorly understood.

1.2 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[10]. Cross-linked networks of cytoskeletal polymers have been a subject of great interest to biologists because of their importance as structural components of cells[20, 63].

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[66].

For *in vitro* reconstitutions, this viscous relaxation is thought to result from transient unbinding and rebinding of intermolecular cross-links[73, 9]. 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.

1.2.1 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 [69]. 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 [10], but we will shortly repeat some important milestones here.

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 [23]. 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 [49, 16], and comprehensive theories of semi-flexible filament solutions, [52] laid a groundwork for theoretical considerations of cross-linked networks. Beginning with the so-called mikado model descriptions[25, 77], 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[67], 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.

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[75, 42]. 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[21, 22, 31, 43]. Making matters even more complicated, ongoing research has begun to uncover added complexity from more highly complex issues such as filament bundling[53, 13]and the effects of active cross-linking by molecular motors[35].

While theorists have built a number of largely successful models that help characterize different aspects of the cross-link dominated response[27, 78, 8], the diversity of behaviors of these networks makes a precise yet general theory more difficult.

1.2.2 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[50, 29, 7, 30, 18, 4]. 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[40, 41, 79, 44]. 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[73], there is still a gap in our understanding of how microscopic cross-link unbinding relates to viscous flows.

Models of Stress Relaxation with Transient Cross-links

The dependence of network rheology on cross-link unbinding is an active subject of theoretical research[53].

Several theoretical methods have addressed cross-link binding and unbinding directly [9, 53] 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 [34] 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[41, 39].

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.

Novelty of Cross-link Slip Approach

Here, we introduce a coarse-grained representation of filament cross-linking in which cross-linked filaments which are able to slide past each other as molecular bonds form and rupture, akin to coarse-grained models of molecular friction[72, 65, 19]. This drag-like coupling has been shown to be an adequate approximation in the case of ionic cross-linking of actin[74, 11], and can be found in the theoretical basis of force-velocity curves for myosin bound filaments[3]. 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.

1.3 Biophysics of filament turnover and cortical flow

1.3.1 Active fluid models

Current models for cortical flow rely on coarse-grained descriptions of actomyosin networks as active fluids, whose motions are driven by gradients of active contractile stress and opposed by an effectively viscous resistance[50]. In these models, gradients of active stress are assumed to reflect spatial variation in motor activity and viscous resistance is assumed to reflect the internal dissipation of elastic resistance due to local remodeling of filaments and/or cross-links [6]. A key virtue of these models is that their behavior is governed by a few parameters (active stress and effective viscosity). By coupling an active fluid description to simple kinetic models for network assembly and disassembly and making active stress and effective viscosity depend on e.g network density and turnover rates, it is possible to capture phenomenological descriptions of cortical flow. Models based on this active fluids description can successfully reproduce spatiotemporal dynamics of cortical flow observed during polarization [50], cell division [70, 61], cell motility [32, 48] and tissue morphogenesis [26].

However, 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.

1.3.2 Models of Stress Relaxation in Active Networks

Studies in living cells have documented fluid-like stress relaxation on timescales of 10-100s of seconds [50, 29, 7, 30, 18, 4]. These modes of stress relaxation are thought to arise both from the transient binding/unbinding of individual cross-links and from the turnover (assembly/disassembly) of actin filaments (ref). Studies of cross-linked and/or bundled actin networks *in vitro* suggest that cross-link unbinding may be sufficient to support viscous relaxation (creep) on very long timescales[73, 40, 41, 79, 44], but is unlikely to explain the rapid large scale cortical deformation and flow observed in living cells. It has been proposed in the field that rapid actin turnover must play a significant role as well. Indeed, photokinetic and single molecule imaging studies studies reveal rapid turnover of cortical actin filaments in living cells on timescales of 10-100 seconds [59]. Previous theoretical models have explored the dependence of stress relaxation on cross-link binding and unbinding analytically [9, 53] and others have explicitly modeled reversible cross-linking in combination with complex mechanics of filament bundles [34, 41, 39], leading to complex viscoelastic stress relaxation. However, until very recently [47] very little attention has been paid to actin turnover as mechanism of stress relaxation.

Recent work has also begun to reveal insights into mechanisms that govern active stress generation in disordered actomyosin networks. In vitro studies have confirmed that local interactions among actin filaments and myosin motors are sufficient to drive macroscopic contraction of disordered networks [56]. Theoretical studies suggest that asymmetrical compliance of actin filaments (stiffer under extension than compression) and spatial differences (dispersion) in motor activity are sufficient conditions for contraction in one [38] and two [37] dimensional networks, although other routes to contractility may also exist [37]. Further work has explored how modulation of network architecture, cross-link dynamics and motor density, activity and assembly state can shape rates and patterns of network deformation [36, 1, 2] or network rheology [46, 35].

Significantly, *in vitro* models for disordered actomyosin networks have used stable actin filaments, and these networks support only transient contraction, either be-

cause of network collapse[1], or buildup of elastic resistance[55], or because network rearrangements (polarity sorting) dissipate the potential to generate contractile force [58, 68]. This suggests that continuous turnover of actin filaments may play a key role in allowing sustained deformation and flow. Recent theoretical and modeling studies have begun to explore how this could work [28, 47, 81], and to explore dynamic behaviors that can emerge in contractile material with turnover [15]. However, there is much to learn about how the buildup and maintenance of contractile force during continuous deformation and flow depends on the local interplay of network architecture, motor activity and filament turnover.

1.3.3 Goal of Present Work

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 [72, 65, 19], 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 [3]. 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.

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

2.1 Measuring *in vivo* turnover rates with smPReSS

In order to better understand the role of , we needed to have a way to . One alternative would have been use .

2.1.1 smPReSS measurement technique

Copy and paste from paper when I have access

2.1.2 Measurements of turnover rate in dividing C. elegans cortex

Copy and paste from paper when I have access

2.2 Disruption of Cortical flows with Jasplakinolide Treatment

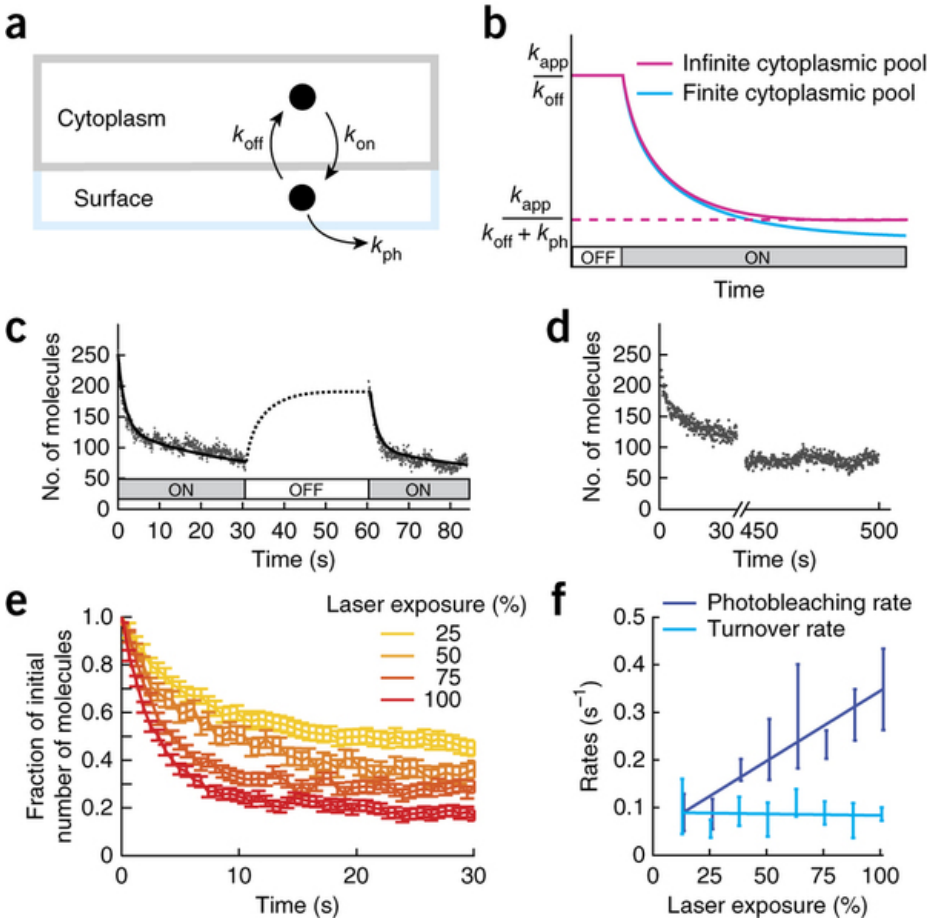


Figure 2.1: smPReSS measurement technique. **a)** The basic kinetic principle: during imaging, the level of surface-associated proteins is set by a dynamic balance of appearance (binding or assembly) at an observable rate k_{app}, disappearance (unbinding or disassembly) at a per-molecule rate k_{off}, 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 $\approx 1.6\mu W/\mu m^2$.

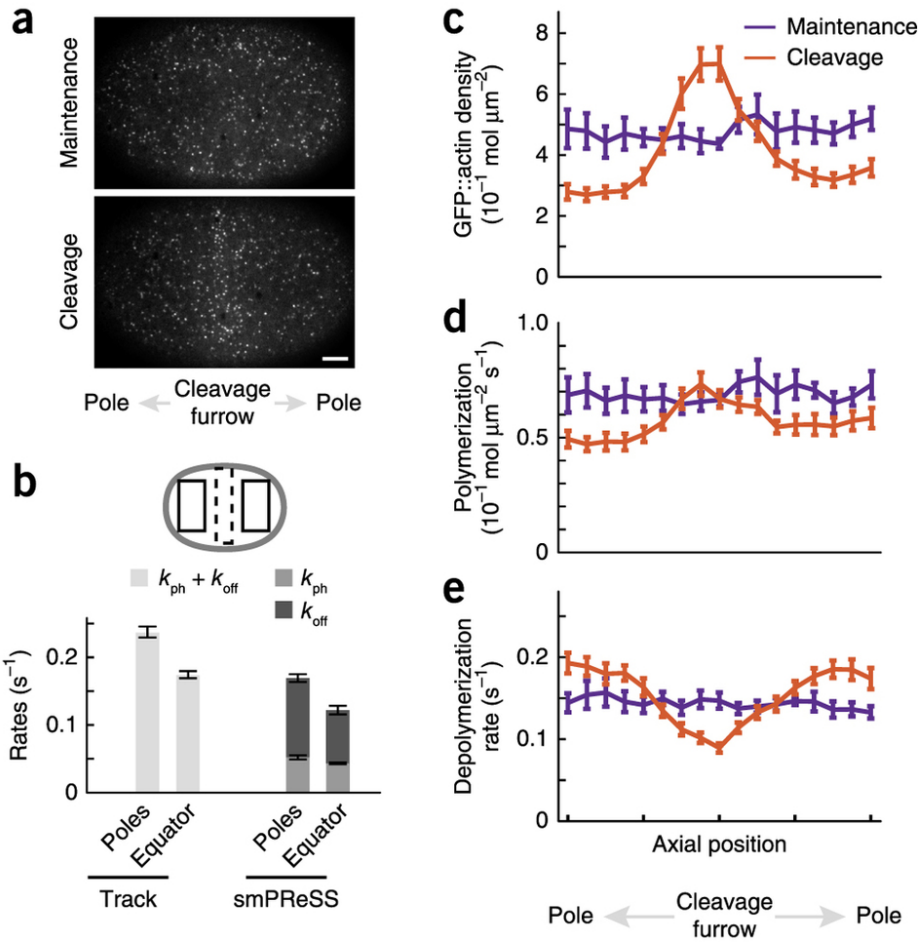


Figure 2.2: Measurement of cortical actin lifetimes. **a)** Near-TIRF micrographs of GFP::actin during the polarity maintenance phase (top) and cleavage (bottom). **b)** Measurements of the indicated turnover rates at the equator and poles during anaphase using tracking (left) or smPReSS (right). Schematic 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, $5\mu m$.

CHAPTER 3
MODELING 2D ACTIVE NETWORKS WITH RECYCLING

3.1 Overview of Model Components

Our motivation is to model essential microscope features of cross-linked actomyosin networks (semi flexible actin filaments with asymmetric compliance, dynamic cross-links, active motors and and continuous filament recycling), in a way that is simple enough to allow systematic exploration of how tuning these microscopic features controls macroscopic network deformation and flow. We focus on 2D networks for computational tractability and because they capture a reasonable approximation of the quasi-2D cortical actomyosin networks that govern flow and deformation in many eukaryotic cells[50, 12], or the quasi-2D networks studied recently in vitro[56, 64].

3.1.1 Asymmetric filament compliance

We model individual filaments as chains of springs with relaxed length l_s. Filaments can therefore be represented as a sequence of nodes with positions $\mathbf{x_i}$, where the index i enumerates over all nodes of all segments. The internal elastic resistance of filament segments gives rise to nearest neighbor force interactions, $\mathbf{F^{\mu}_{i,i+1}}$, of the form

$$\mathbf{F^{\mu}_{i,i+1}} = \mu \frac{|\mathbf{x_{i+1}} - \mathbf{x_i}| - l_s}{l_s} \left(\frac{\mathbf{x_{i+1}} - \mathbf{x_i}}{l_s} \right) \tag{3.1}$$

where the modulus, μ, is a composite quantity representing both filament and cross-linker compliance in a manner similar to a proposed effective medium theory [8]. To model asymmetric filament compliance, we assign a different value to the modulus μ, depending on whether the strain on a given filament segment, $(|\mathbf{x_{i-1}} - \mathbf{x_i}| - l_s)/l_s$, is greater or less than 0. In the limit of highly rigid cross-links and flexible filaments, our

model reduces to the pure semi-flexible filament models of [25, 77]. In the opposite regime of nearly rigid filaments and highly flexible cross links, our model is essentially the same as that of [8] in small strain regimes before any nonlinear cross link stiffening. In a departure from those previous models, we assume here that the magnitude of the force on interior cross-links is the same as those on the exterior. This approach ignores the variation in strain on these two sets of cross-links as addressed in [8], but we choose to ignore this variation in favor of an approximated, global mean approach.

Because we are dealing with semi-flexible filaments we also introduce a bending modulus between our filament segments such that the restoring force is proportional to the angle between the filament segments and points in the direction orthogonal to the filament direction, $\mathbf{u_i} = (\mathbf{x_{i-1}} - \mathbf{x_i})/|\mathbf{x_{i-1}} - \mathbf{x_i}|$.

The total internal force on a filament node i can therefore be written as:

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

Introducing a filament bending stiffness adds another mode of asymmetric compliance since filaments can bend/buckle internally under compression. For the majority of the work presented here, we have set $l_s = L$ to obviate dependence on bending driven asymmetries. However, as shown in supplemental figure xxx (TBA) the major points of the paper are still valid for $l_s < L$ under the condition that $\kappa/l_s \gg \mu_c$.

3.1.2 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 [34, 8]. Here, we introduce a simpler coarse-grained model for dynamic cross-links by replacing many transient elastic interactions with an effective drag-like coupling between every pair of overlapping segments.

$$\mathbf{F^{\xi}_{i-1,i}} = \xi \sum_j \frac{l_s - |s_{ij} - s_i|}{l_s} (\mathbf{v_{i-1,i}} - \mathbf{v_{j-1,j}}) \tag{3.3}$$

where $\mathbf{v_{n-1,n}}$ represent the average velocity of the filament segment spanning nodes

$n-1$ and n, and the sum is taken over all filament segments such that the segment from node $j-1$ to node j intersects the segment from $i-1$ to i at the location s_{ij}.

$$\mathbf{F_i^{coup}} = \mathbf{F^{\xi}_{i-1,i}} + \mathbf{F^{\xi}_{i,i-1}} \tag{3.4}$$

This model assumes a linear relation between the drag force and the velocity difference between attached segments. This drag-like coupling has been shown to be an adequate approximation in the case of ionic cross-linking of actin[74, 11], and can be found in the theoretical basis of force-velocity curves for myosin bound filaments[3]. Although non-linearities can arise through force dependent detachment kinetics and/or non-linear force extension of cross-links, we assume that inhomogeneities from non-linear effects are of second or higher order. With this assumption, the motion of filaments can be described by a deterministic dynamical equation of the form

$$0 = -l_s\zeta\mathbf{v_i} - \mathbf{F_i^{coup}} + \mathbf{F_i^{int}} \tag{3.5}$$

Here, the first term is the filament's intrinsic drag through its embedding fluid, ζ, while the second comes from the drag-like coupling between filaments, ξ.

3.1.3 Active coupling for motor driven filament interactions

To add motor activity we select a subset of cross-linked points and impose an additional force of magnitude υ on each of the overlapping filament segments, directed towards the (+) end of that segment, $\mathbf{u_i} = (\mathbf{x_{i-1}} - \mathbf{x_i})/|\mathbf{x_{i-1}} - \mathbf{x_i}|$. Thus, the total "active" force on a given filament segment is

$$\mathbf{F^{\upsilon}_{i-1,i}} = \upsilon\mathbf{u_i}\sum_j \frac{l_s - |s_{ij} - s_i|}{l_s} q_{ij} \tag{3.6}$$

where q_{ij} equals 0 or 1 depending on whether there is an "active" cross-linker at this location. To model dispersion of motor activity, we set $q_{ij} = 1$ on a randomly selected subset of cross-link points, such that $\bar{q} = \phi$, where $\bar{q}$ indicates the mean of q.

Finally, for each active force, $\mathbf{F_j^{act}}$, imparted by filament j, we must also impart

the opposite force onto the filament between i and $i+1$ as well. Therefore, the entire equation for activity will appear as

$$\mathbf{F_i^{act}} = \mathbf{F^v_{i-1,i}} + \mathbf{F^v_{i,i+1}} - \sum_j \mathbf{F^v_{j-1,j}} q_{ij} \tag{3.7}$$

This will leave us with a full equation of motion given by the sum of each of the parts defined above.

$$0 = -L\zeta \mathbf{v_i} - \mathbf{F_i^{coup}} + \mathbf{F_i^{int}} + \mathbf{F_i^{act}} \tag{3.8}$$

3.1.4 2D network formation

We used a mikado model approach [71] to initialize a minimal network of connected unstressed linear filaments in a rectangular 2D domain. We generate 2D networks of these semi-flexible filaments by laying down straight lines of length, L, with random position and orientation. We then assume that overlapping filaments become cross-linked at their points of overlap. Although real cytoskeletal networks may form with non-negligible anisotropy, for simplicity, we focus on isotropically initialized networks. 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[25]. 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, $2/l_c$. In the absence of cross-link slip, we expect the network to form a connected solid with a well defined elastic modulus[25, 77].

3.1.5 External applied stress

We can model our active networks as a coupled system of differential equations satisfying 3.8. However, to probe the passive response of the network, we also wish to incorporate externally applied stresses. Although the general passive mechanical response of this system may be very complex, we focus our attention on low frequency

deformations and the steady-state creep response of the system to an applied stress. To do this we introduce a fixed stress, σ along a fixed domain at one edge of the network. The stress is applied via individual forces to the filaments lying within a patch of size D_w such that the sum of individual forces is equal to the applied stress times the height of the domain. These forces point in the direction, $\hat{\mathbf{x}}$, producing an extension of the patch. The region of applied stress does not move as the network deforms, allowing us to more easily focus our attention on a fixed sized domain.

Finally, we add a 0 velocity constraint at the other edge of our domain of interest. We assume that our network is in the "dry," low Reynold's number limit, where inertial effects are so small that we can equate our total force to 0. Therefore, we have a dynamical system of wormlike chain filaments satisfying

$$0 = -L\zeta \mathbf{v_i} - \mathbf{F_i^{coup}} + \mathbf{F_i^{int}} + \mathbf{F_i^{act}} + \sigma \hat{\mathbf{u}}(\mathbf{x_i}) \tag{3.9}$$

subject to constraints such that $\mathbf{v_i}(\mathbf{x})$ is 0 with $x = 0$. This results in an implicit differential equation for filament segments which can be discretized and integrated in time to produce a solution for the motion of the system.

3.1.6 Modeling filament turnover

In living cells, actin filament assembly is governed by multiple factors that control nucleation, elongation and filament branching. Likewise filament disassembly is governed by multiple factors that promote filament severing and monomer dissociation at filament ends. Here, we focus on a lowest order model for filament recycling 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. 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 assumption, at fixed time interval $\tau_s < 0.01 \cdot \tau_r$ (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 this continuous turnover as filament recycling, to $k_{diss} = 1/\tau_r$ as the recycling rate, and to τ_r as the recycling time.

3.1.7 Simulation methods

Details of our simulation approach can be found in the Appendix. Briefly, equations 3.1,3.4,3.6 and 3.9 define a coupled system of ordinary differential equations for the velocities of the endpoints of filament segments, $\dot{\mathbf{x}}$. These equations are coupled by the effective cross-link friction on segment overlap points, yielding a system of the form:

$$\mathbf{A} \cdot \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{3.10}$$

where $\mathbf{A}$ represents a coupling matrix between endpoints of filaments that overlap, and $\mathbf{f}(\mathbf{x})$ is the spring force between pairs of filament segment endpoints. 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 segment 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 segment endpoints within a distance $0.05 \cdot D_x$ from the left edge of the patch. To simulate free contraction, we removed all constraints at boundaries; to assess buildup of contractile stress under isometricc conditions, we pinned both left and right edges of the network as described above.

We smoothed all filament interactions, force fields, and constraints over small regions such that the equations contained no sharp discontinuities. The nominal units for length, force, and time are μm, nN, and s, respectively. We explored parameter space around an estimate of biologically relevant parameter values given in Table 3.1.

Table 3.1: Simulation Parameter Values

parameter	**symbol**	**physiological estimate**
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	$0.5\mu m$
domain size	$D_x \times D_y$	$20 \times 50\mu m$

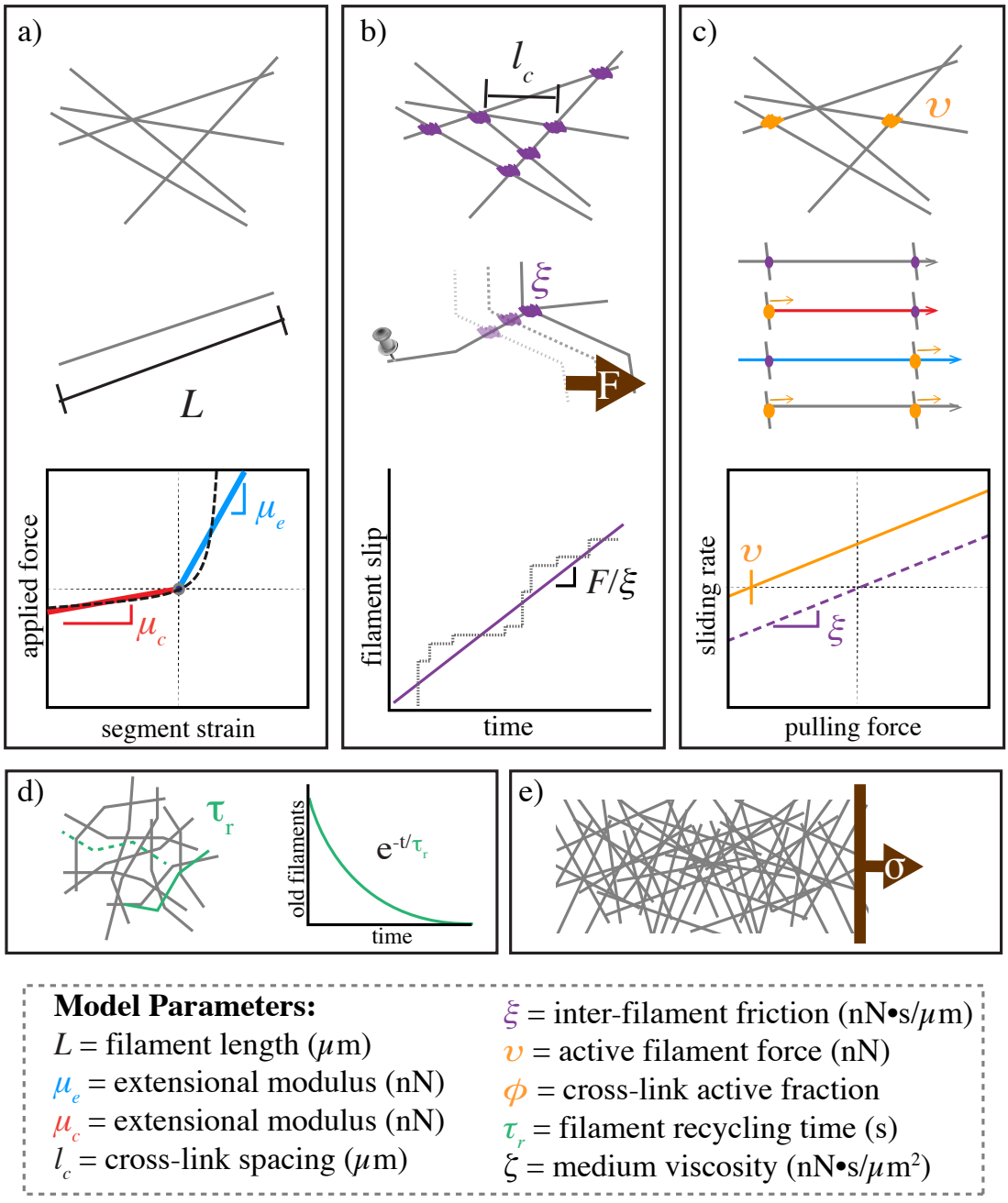


Figure 3.1: Schematic of modeling framework. a) Asymmetric filament compliance.Filaments have smaller spring constant for compression than for extension. b) Cross-link slip. Cross-links are coupled by an effective drag, such that their relative motion is proportional to any applied force. c) Motor activity. Filament activity manifests as a basal sliding rate even in the absence of an external force. Fractional activity. Only a subset of filament cross-links are active, resulting in differential force exertion along the filament. d) Filament recycling. Filaments are turned over at a constant rate, leading to a refreshing in the strain state of all filaments after a characteristic timescale. e) Applied stress. In simulations with passive cross-links, and external stress is applied as force filed acting on a fixed spatial domain.

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 4.5 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 $L/l_c \sim 6$.

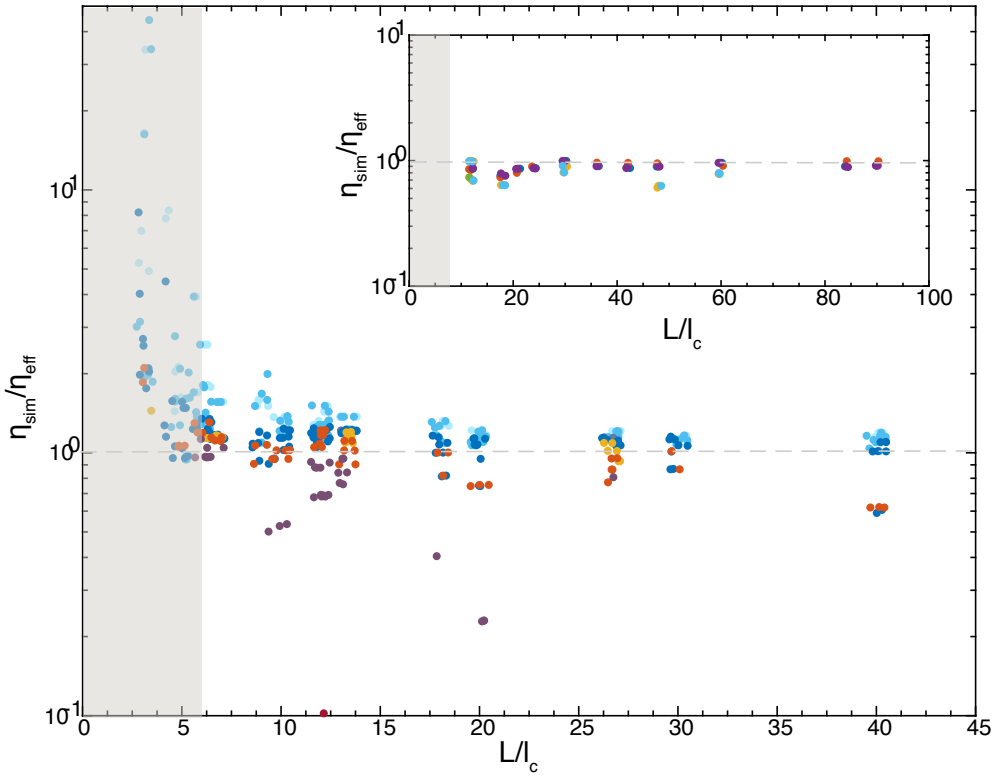


Figure 4.1: Ratio of effective viscosity measured by shear simulation to predicted effective viscosity as a function of connectivity, L/l_c. Inset: Same measurement for extensional simulations

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[25, 77].

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 4.1.1. 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

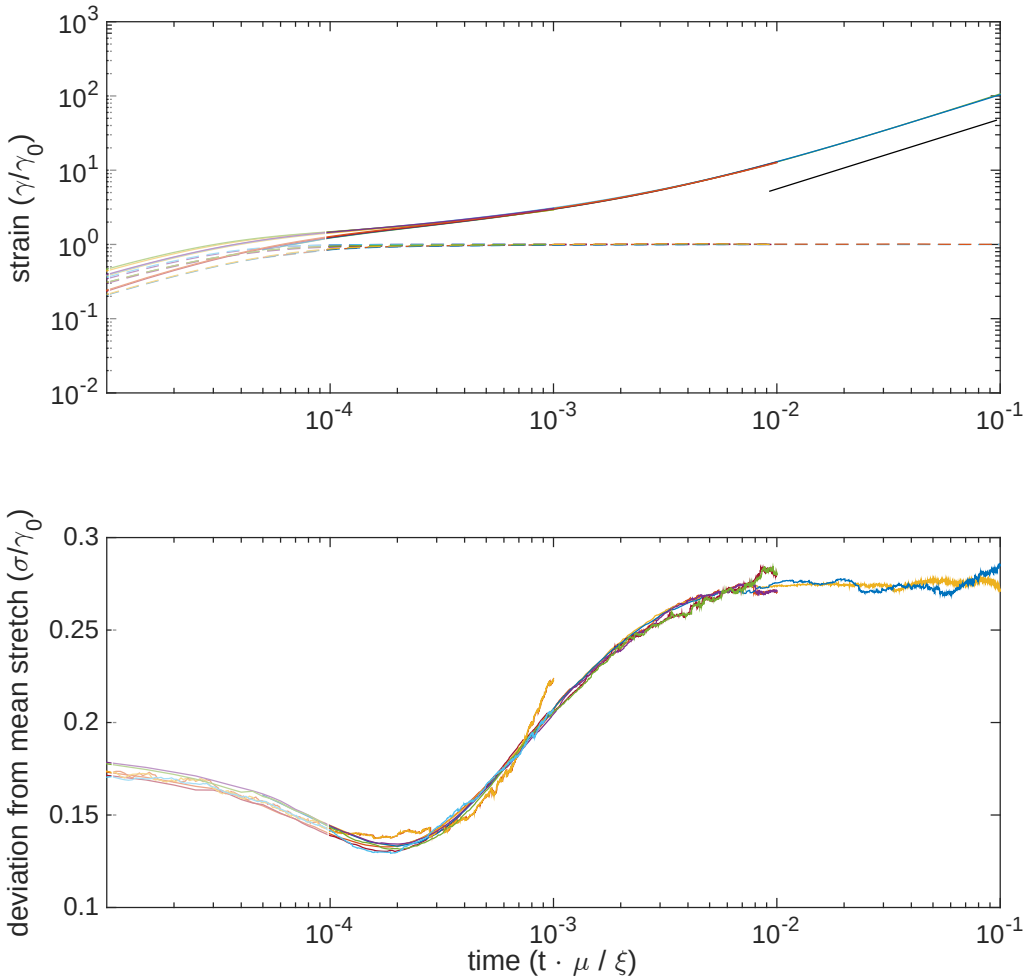


Figure 4.2: Network and filament strain for different filament drag coefficient parameters. (top) Plot of total strain normalized by the final mean filament strain, $\delta L/L$. Dashed lines show the amount of strain from affine mechanical stretching. (bottom) Standard deviation of filament extension 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.

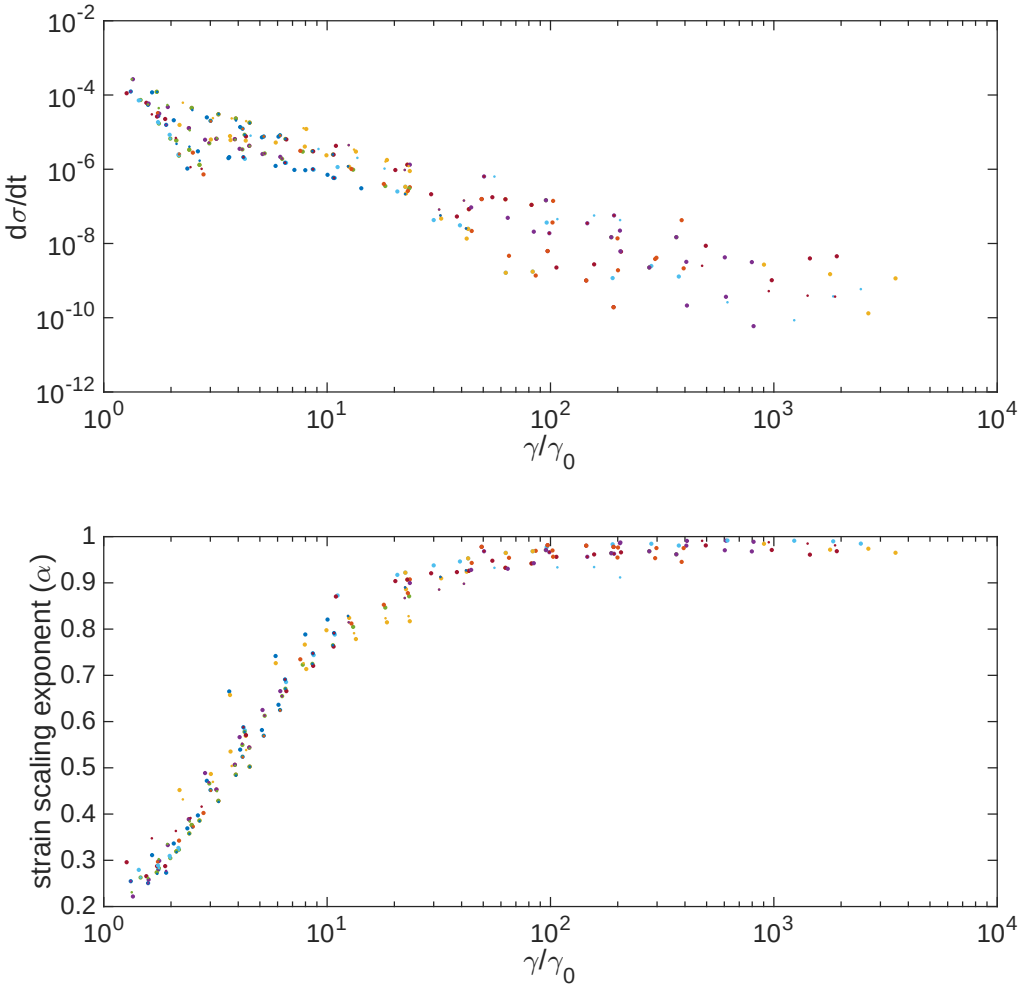


Figure 4.3: Sublinear network strain ends as change in filament strain decays. (top) Change in standard deviation of filament strain, σ, as a function of strain relative to pure mechanical strain. (bottom) Dependence of strain rate exponent as a function of strain relative to pure mechanical strain, 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[17].

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

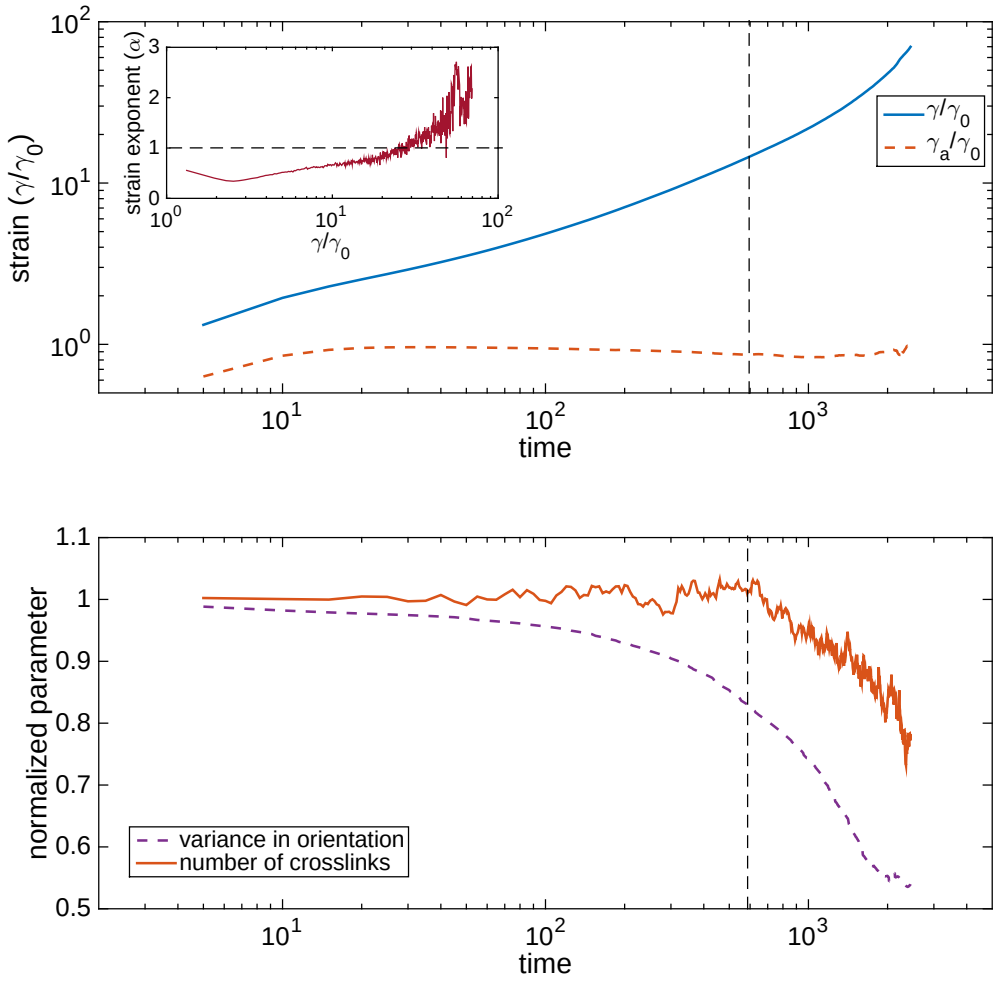


Figure 4.4: Creep response of a network transitioning to phase D. (top) Strain curves for a network undergoing large scale deformation. Inset shows strain exponent as a function of strain (exponent passes 1). (bottom) Traces for the variance in filament orientation and number of cross links. Vertical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

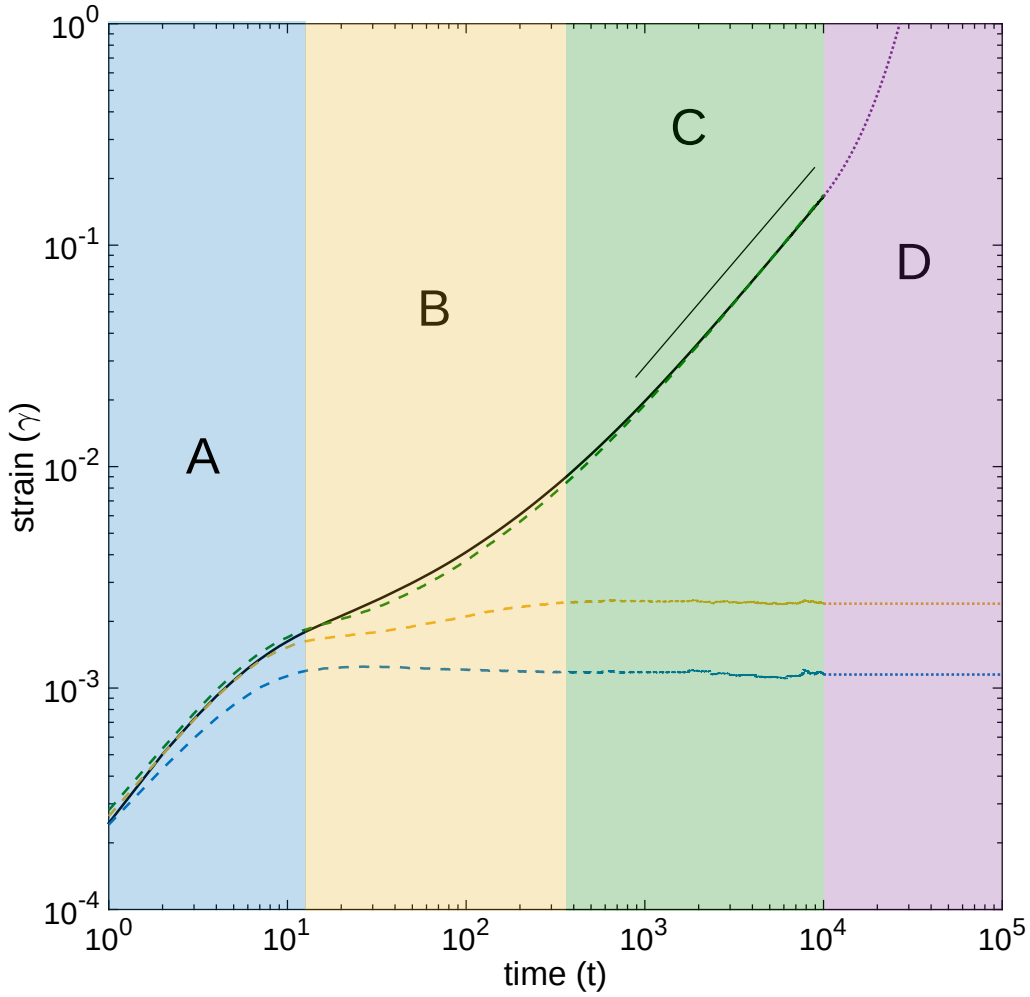


Figure 4.5: Schematic of the general creep response of compliant filament networks illustrating the 4 phases of deformation: A) rapid mechanical response, B) combination of slow filament stretching and cross-link slip, C) cross-link slip dominated (line indicates slope of one), D) network tearing from 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.

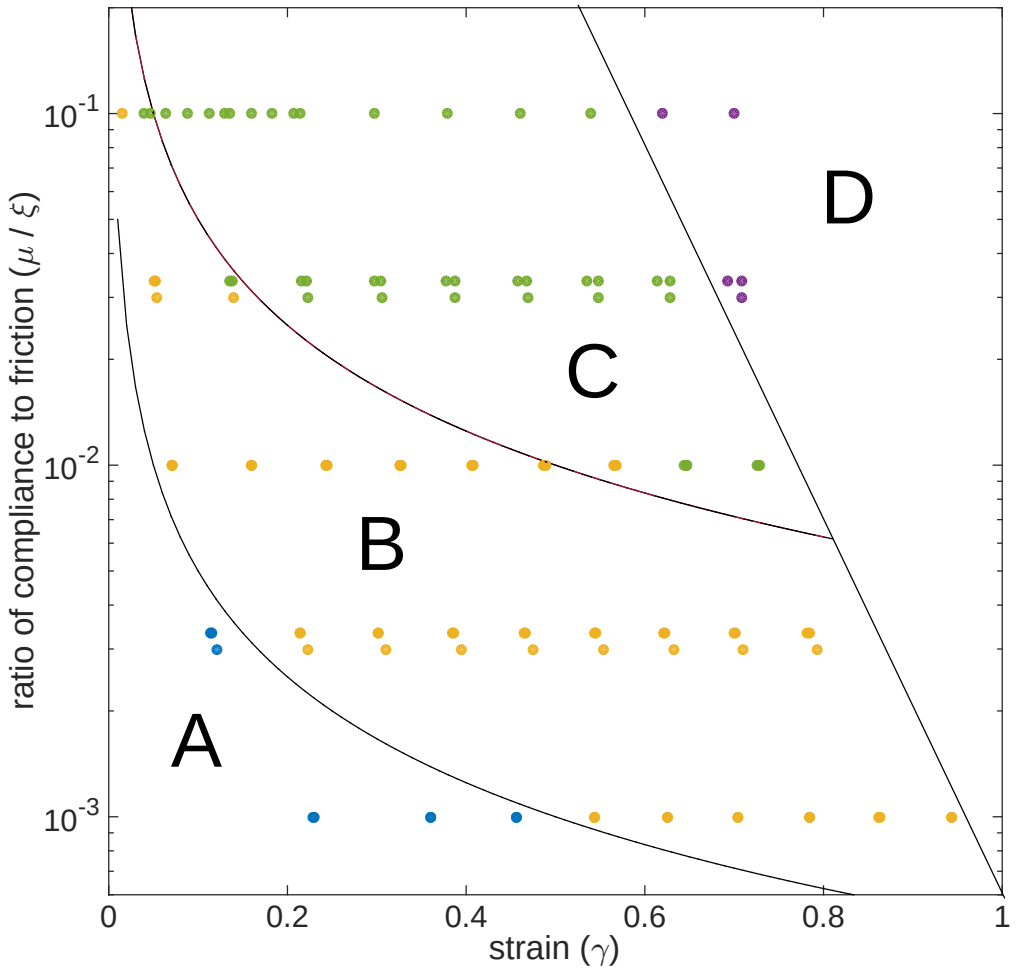


Figure 4.6: phase diagram of creep response for different filament extension, μ and cross-link friction, ξ. Yellow, green, and purple dots correspond to creep measurements $\gamma \sim t^{\alpha}$ with $\alpha < 0.92$, $0.92 < \alpha < 0.98$, or $\alpha > 0.98$ respectively. Blue dots represent creep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

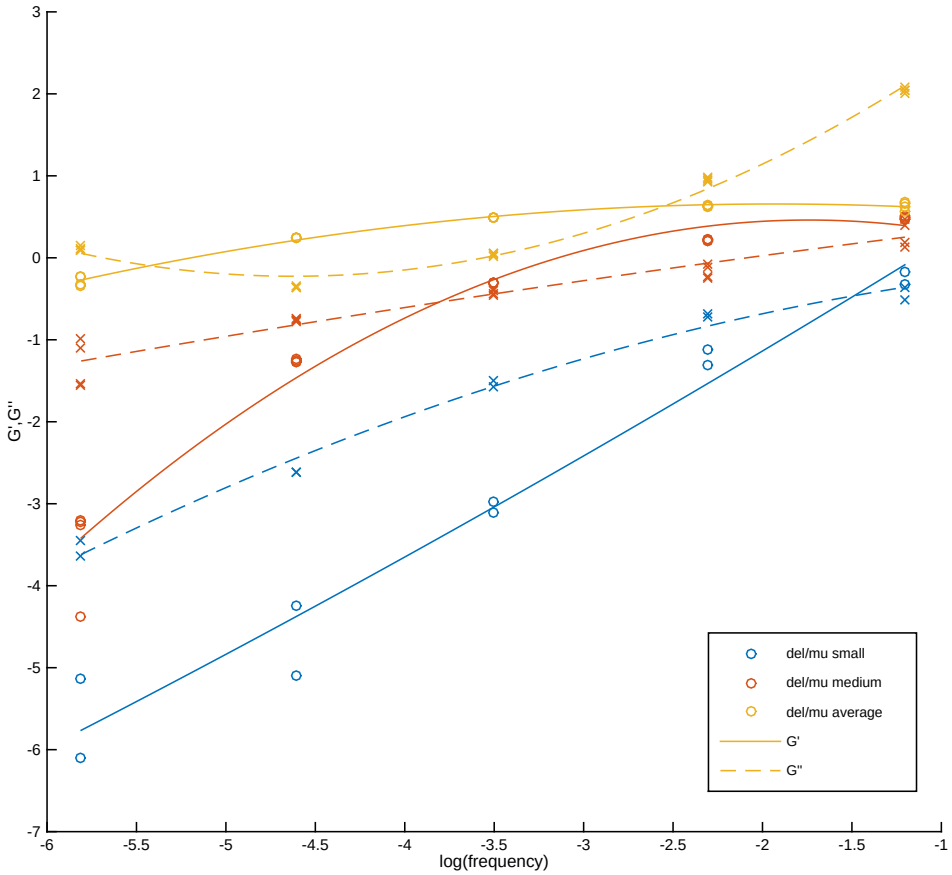


Figure 4.7: Frequency dependent moduli for networks.

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.

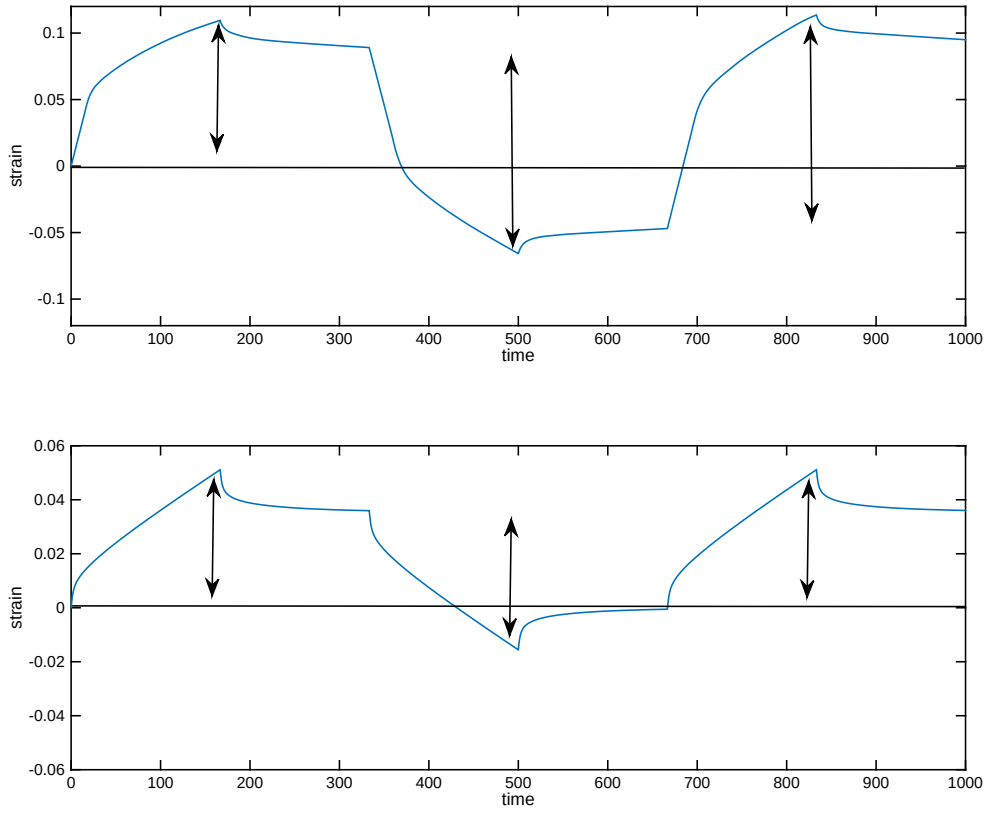


Figure 4.8: Creep curves in the presence of reversing applied stress for (a) nonlinear extension or (b) linear extension. Note that for linear filaments the induced strain returns to approximately 0 after a complete cycle, while in the nonlinear case the cycle is not completely reversible.

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[56] as well as in eukaryotic actomyosin cortices[50].

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 [9], 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 and Discussion

We aim to characterize how rates and patterns of cortical flow are shaped by complex dependencies of active stress generation and passive stress dissipation on network architecture, local coupling (active and passive) between filaments and filament recycling. We approached this in three steps: First, we analyzed the passive deformations of cross-linked networks (absent active motors) in response to a constant external force. Then, we analyzed the dynamics of internal stress buildup and dissipation in the same networks, but with active motors, as they contract freely or build force against fixed external boundaries. Finally, we consider the dynamic interplay of internal stress buildup, contraction, and stress relaxation in networks that undergo steady state flow in response to spatial gradients of motor activity.

5.1.1 Filament recycling prevents cortical tearing and modulates the viscous stress relaxation of passive filament networks

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 in the absence of filament recycling and motor activity, we imposed an external force on the simulated network, and then quantified the mechanical response in terms of internal network stress and network strain as a function of time. Figure 5.1a shows the typical response of a simulated network. We measured the local velocity of the network at different

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

During early (not shown) and intermediate (Figure 5.1b) stages of the deformation, the internal stress (blue) was nearly constant throughout the material while the velocity (orange) increased linearly with distance from the site of network attachment, indicating an approximately uniform deformation (strain) rate throughout the material. Accordingly, we report the network response in terms of time-dependent bulk material stress and strain.

Plotting the bulk stress and strain as a function of time revealed that the deformation occurred in three qualitatively distinct phases (Figure 5.1a,c). On short timescales the response was viscoelastic, with a rapid buildup of internal stress and a rapid ∼exponential approach to a fixed strain, which represents the elastic limit in the absence of cross-link slip predicted by [25]. On intermediate timescales, the 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 slipped past one another against the effective cross-link drag. This linear relationship between strain and time characterizes a material with an effective viscosity, η, given by the ratio of the applied stress to the strain rate. We define the transition time between the fast, viscoelastic phase and the slower, effectively viscous deformation phase as τ_c. Finally, as the network strain approached a critical value ($\sim$ 30% for the simulation in Figure 5.1), strain thinning led to decreased network connectivity, local tearing, and acceleration of the network deformation (see inset in Figure 5.1c), eventually resulting in the highly heterogeneous network structure shown in the t=440s example of Figure 5.1a.

Network architecture sets the rate and timescales of deformation. To better understand how network architecture and cross-link dynamics control effective viscosity and the timescale for transition to viscous behavior, we systematically varied

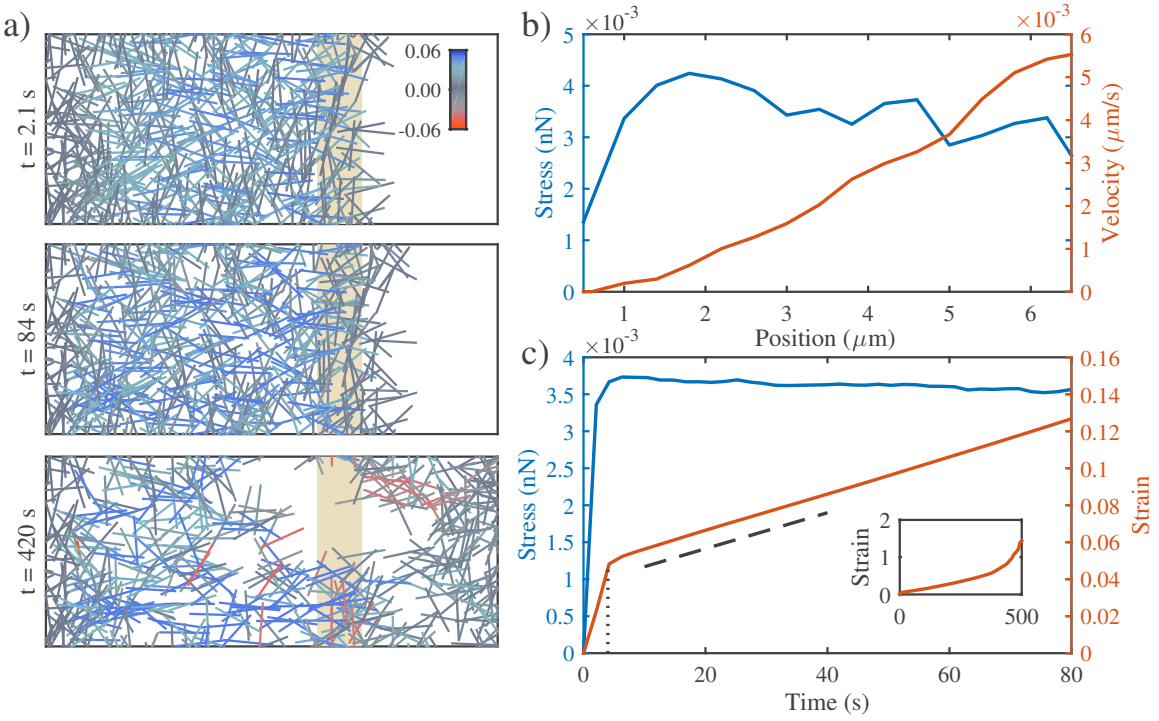


Figure 5.1: Networks with passive cross-links and no filament turnover undergo three stages of deformation in response to an extensional force. **a)** Three successive time points from a simulation of a $4 \times 6.6\,\mu m$ network deforming under an applied extensional stress of $0.005\ nN/\mu m$ (stress is applied to filaments in the region indicated by the tan bar). Network deforms to a final dimension of $\sim 4 \times 10 \mu m$. In this and all subsequent figures, filaments are color-coded with respect 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 phase accompanies rapid buildup of internal network stress; (ii) after a characteristic time τ_c (indicated by vertical dotted line) the network deforms like a material with a constant effective viscosity, η_c, as indicated by the slope of the dashed line; (iii) at long times, the strain accelerates (see inset) as the network undergoes strain thinning and eventually tears.

network parameters (see S1 Table), and measured the elastic modulus, G_0, effective viscosity, η, and transition time, τ_c, in response to a fixed external stress. We observed the transition from a viscoelastic to an effectively viscous phase for the entire

range of parameters that we sampled. The elastic limit that we observe during the viscoelastic phase agreed closely with the closed form solution for the elastic modulus $G_0 \sim \mu/l_c$ predicted by a previous model [25] for networks of semi-flexible filaments with irreversible cross-links (Mechanical properties of passive networks. **a)** Elastic modulus of networks. Our measurements closely match prediction of $G_0 \sim \mu/l_c$. **b)** Placeholder for inevitably another figure relevant to passive properties). A simple theoretical analysis (shown in S1 Text) predicts that in the viscous phase, the effective viscosity should be proportional to the cross-link drag coefficient and to the square of the number of cross-links per filament, with a constant of proportionality $\pi/4$. We define this predicted scaling of effective viscoisty as η_c.

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

As shown in Figure 5.2a, our simulations agree well with this prediction for a large range of network parameters. For many linear viscoelastic materials, the ratio of the viscosity, η_c, to the elastic modulus, G_0, is a general indicator of the transition timescale from elastic to viscous behavior[51]. Using our approximations of the elastic modulus and viscosity, we predict a crossover time, $\tau_c \approx L^2\xi/l_c\mu$. By measuring the time at which the strain rate became nearly constant (i.e $\gamma \sim t^n$ with $n > 0.8$) we obtained an estimate of this time for a wide variety of simulation parameters. As shown in Figure 5.2b, our approximation is in good agreement with the observed transition time, indicating that the passive responses of our simulated networks are well represented by effectively linear bulk properties, at least over small strains.

Filament recycling rescues network tearing and modulates effective viscosity. To explore how filament recycling shapes the passive network response to an applied force, we ran a series of simulations with identical filament lengths and network densities and cross-link drag coefficients, while varying the filament recycling time $\tau_r = 1/k_{diss}$. Figure 5.3a illustrates the results for a particular set of network parameters. In the absence of filament recycling, strain thinning and network tearing lead to a rapid increase in strain rate above a critical strain of $\sim 40\%$.

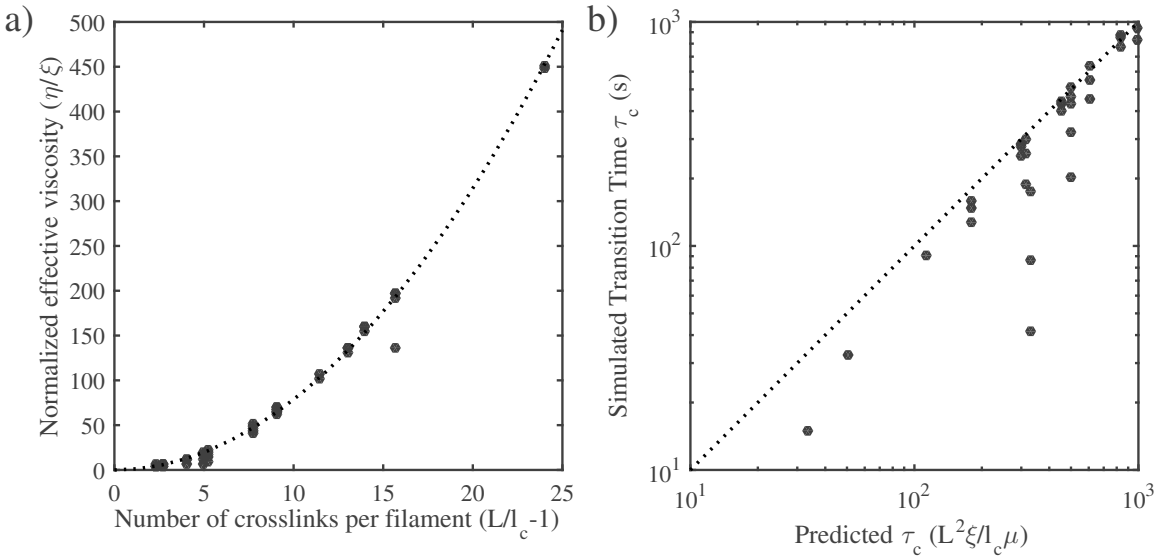


Figure 5.2: Network architecture sets the rate and timescales of deformation. **a)** The effective viscosity depends on the drag coefficient and the density of the network. Data points are the normalized effective viscosity from simulations (effective viscosity measured in fluid phase divided by the cross link friction coefficient) vs the number of cross links per filament $(L/l_c - 1)$. Dotted line indicates the relationship predicted by a simple theory, $\eta_c = \xi(L/l_c - 1)^2$ **b)** The transition to viscous behavior occurs at a characteristic time, τ_c.

Progressively decreasing the filament recycling time led to a progressive increase in the rate of network deformation during the effectively viscous phase and an increase in the critical strain at which the network began to tear. As we decreased the recycling time below a critical recycling time (τ_{crit}), the networks began to sustain effectively viscous deformation indefinitely, as shown by the lack of strain thinning in the strain profiles of (ANOTHER SUPPLEMENTAL FIGURE). Intuitively, steady state deformation is achieved when the rate of filament depletion by strain thinning is balanced by a sufficiently high rate of filament recycling (i.e. a sufficiently low recycling time). To determine the critical recycling time, we write an equation for the rate of change in filament density ρ, as a function of filament recycling $(k_{app} - k_{diss}\rho)$ and strain thinning $(-\dot{\gamma}\rho)$. These terms can be rewritten to give the following

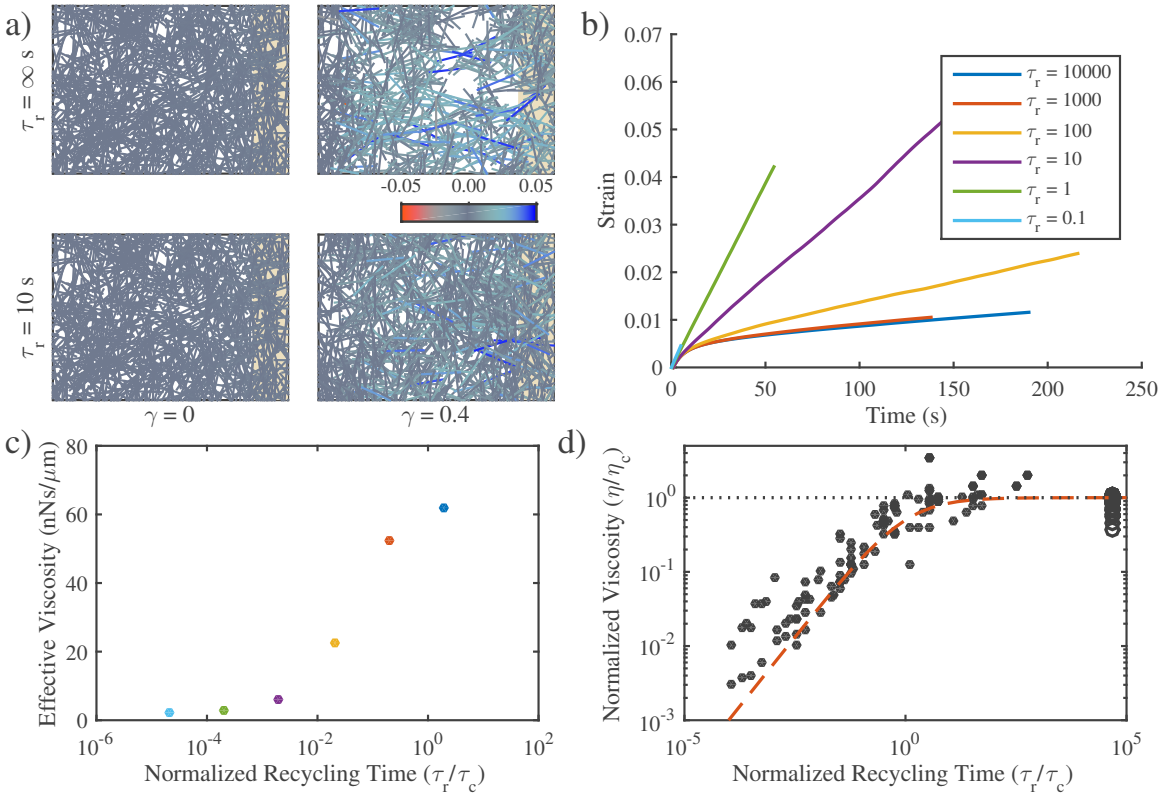


Figure 5.3: Filament recycling modulates effective viscosity in two regimes. **a)** Examples of $20 \times 12 \mu m$ network under 0.001 $nN/\mu m$ extensional stress with recycling ($\tau_r = 10s$) and without, ($\tau_r = \infty$). Both images are taken when the patches had reached a net strain of 0.4. The network with recycling doesn't appear to change shape because its components have been recycled to remain in the original domain. Network parameters: $L = 3\ \mu m$, $l_c = 0.5\ \mu m$, $\xi = 10\ nN \cdot s/\mu m$. **b)** Strain curves for identical networks with varying levels of filament recycling. Network parameters: $L = 3\ \mu m$, $l_c = 0.5\ \mu m$, $\xi = 10\ nN \cdot s/\mu m$. **c)** Plotting the effective viscosity derived from the slopes of the lines in panel a. **d)** Effective viscosities (normalized by the effective viscosity in the absence of recycling, η_c) as a function of the normalized recycling time. When the recycling timescale is significantly less than the passive relaxation timescale, the viscosity of the network becomes dependent on recycling time. Red dashed line indicates the approximation given in equation 5.3 for $m = 3/4$.

$$\frac{d\rho}{dt} = \frac{\rho_0 - \rho}{\tau_r} - \frac{\sigma}{\eta_c(\rho)}\rho \tag{5.2}$$

where k_{diss} has been replaced by $1/\tau_r$, and $\rho_0 = k_{app}\tau_r$, and $\dot{\gamma}$ has been replaced by

σ/η_c. For our networks, the effective viscosity, η_c, is dependent on the filament density (through l_c) so this dependence must be included. Solving this equation for its steady states, and replacing the initial density, ρ_0, with the length density approximation, $2/l_c$, we find that a constant steady state density only exists under the condition $\tau_r < \tau_{crit} = \eta_c/2\sigma$.

Reducing recycling time, τ_r, below τ_{crit} produced different effects on steady state deformation rates depending on the relative values of τ_r and τ_c, the characteristic time for the transition to effectively viscous deformation in the absence of recycling. For $\tau_r > \tau_c$, the deformation rate is dominated by cross-link resistance to sliding of strained filaments, and the effective viscosity remained ∼constant with decreasing τ_r; for $\tau_r < \tau_c$, effective viscosity decreased sublinearly with decreasing τ_r. Intuitively, this is because, for $\tau_r < \tau_c$, the deformation rate is limited by the level of elastic stress on partially strained filaments; By replacing partially strained with unstrained filaments, the network is able to tune the mean level of stress and thus the deformation rate.

To confirm this relationship more generally, we allowed filament lengths, network density and cross link friction to vary more widely, and we measured the network deformation rates while varying filament recycling times (Figure 5.3a,b). We then plotted the normalized effective viscosity (ratio of effective viscosity with recycling to effective viscosity without recycling, η_c) vs a normalized recycling rate (recycling time scaled by τ_c). Indeed, we found that the normalized effective viscosity measured during steady state flow begins to decrease when the recycling time falls below τ_c and below this value the effective viscosity falls off nearly linearly with recycling time to minimal values (Figure 5.3c).

To describe this we introduce (based on linear viscoelastic models of [51]) an effective recycling viscosity, η_r, which can be tuned between the τ_r dependent and independent regimes, depending on the value of the recycling timescale.

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

For $\tau_r \gg \tau_c$, this simplifies to $\eta_r \approx \eta_c$, while for $\tau_r \ll \tau_c$, this simplifies to

$\eta_r \sim (\tau_r/\tau_c)^m$, which matches with our measurements as found in Figure 5.3d for a large range of parameters (with $m = 3/4$). While the origins of the 3/4 scaling remain unclear, this model captures a simple quantitative description of our simulation data.

In summary, our simulations predict that tuning recycling times below a critical value τ_{crit}, allows networks to undergo continuous viscous deformation, for long times, without tearing, for a wide range of different effective viscosities and deformation rates. Given a suitably low strain rate, τ_{crit} will be substantially larger than the other timescales of interest. For $\tau_r < \tau_{crit}$, modulating filament recycling times can tune the network between two regimes. For $\tau_r > \tau_c$, the deformation is limited by effective cross-link friction. The effective viscosity depends on the strength of inter-filament cross-linking and the network's architecture, and is relatively insensitive to changes in recycling rate. For $\tau_r < \tau_c$, the deformation is governed by the buildup of elastic stress on network filaments, and effective viscosity becomes strongly dependent on recycling time.

These findings are in agreement with previous simulations of passive creep in cross-linked networks subjected to extensional stress [33] . Kim et al considered a different form of filament turnover (filament treadmilling) in networks with irreversible cross links.They identified two regimes of deformation depending on the level of applied stress and the filament turnover rate: a ?stress-dependent regime? in which filaments turnover before they are strained to an elastic limit and network deformation is linearly viscous and tuned by the turnover rate; and a ?stress-independent regime? in which filaments reach an elastic limit before turning over andthe creep rate depends only on the turnover rate, and is insensitive to variation in applied stress. The short recycling time regime ($\tau_r < \tau_c$) that we observe, in which the mechanics are governed by filament extension, is directly equivalent to the stress-dependent regime described by Kim et al. For this regime, our model yields a theoretical description of the effective viscosity found in [33]. The $\tau_r > \tau_c$ regime that we observe here corresponds to the stress-independent regime of [33], but with a key difference. in [33], there was no cross-link unbinding so without of filament turnover, the network would not deform beyond its elastic limit. In contrast, our simulations always require non-zero cross-link slip so there is always some viscous network deformation. Therefore, in the

regime of long recycling times our model approaches the limit of cross-link dominated viscosity whereas the model of [33] approached an infinite viscosity limit.

5.1.2 Filament recycling allows persistent stress buildup in active networks

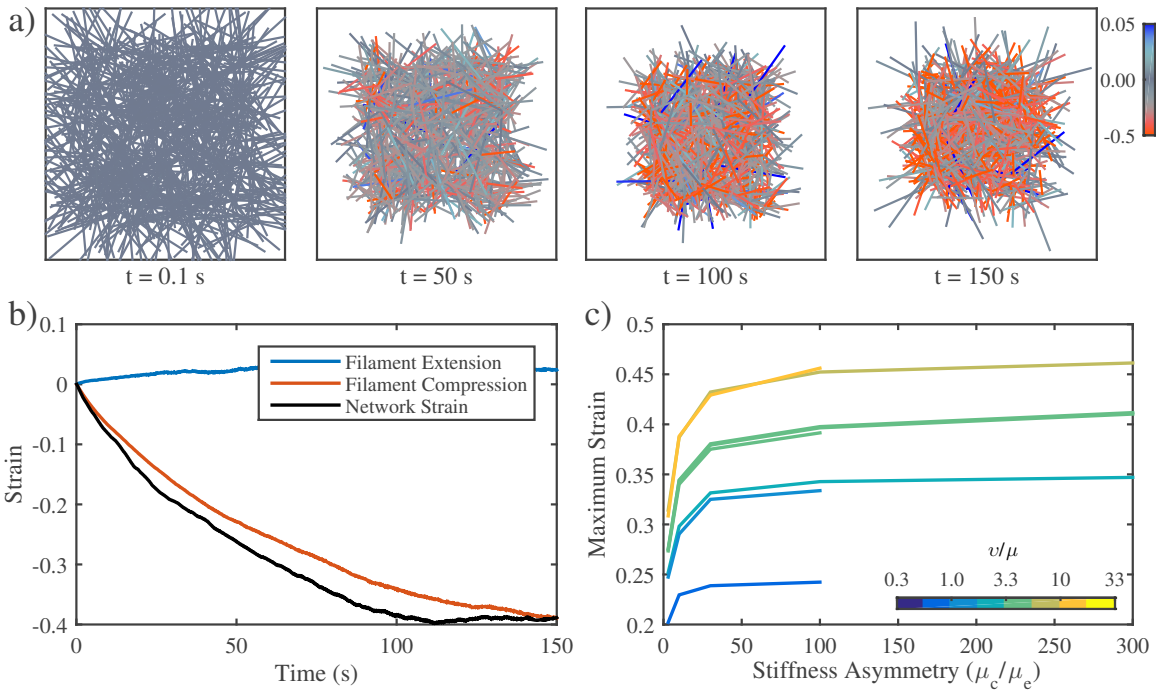


Figure 5.4: In the absence of filament recycling, active networks with free boundaries contract and then stall against passive resistance to network compression. **a)** Example of an active network contracting. Note the buildup of compressive stress as contraction approaches stall between 100 s and 150 s. Network parameters: $L = 5\mu m$, $l_c = 0.3\ \mu m$, $\xi = 100\ nN \cdot s/\mu m$, $v = 0.1\ nN$. **b)** Plots showing time evolution of total network strain and the average extensional (blue) or compressive (red) strain on individual filaments. **c)** The network's ability to deform relies on the magnitude of asymmetric filament compliance. Total network strain also increases with the applied myosin force v. Note that the extent of contraction approaches an asymptotic limit as the stiffness asymmetry approaches a ratio of ~ 100.

In the absence of filament recycling, active networks with free boundaries contract and then stall against passive resistance to network compression.

Previous theoretical and experimental studies[38, 56, 35] identified asymmetric filament compliance and dispersion in motor force as minimal requirements for contraction of disordered networks. To test if our simple implementation of these two requirements (see Models section) was sufficient to produce macroscopic contraction, we simulated active networks that were unconstrained by external attachments. Turning on motor activity in an initially unstrained network at $t = 0$ produced a rapid initial contraction, followed by a progressive buildup of elastic stress due to compression of individual filaments and an ∼exponential approach to stall (Figure 5.4). The time to stall, τ_s, scaled as $L\xi/\upsilon$ (see Tearing of active networks is prevented via recycling. **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 recycling time. **c)** Time trace of internal stresses for network in panel a. **d)** Time trace of internal stresses for network in panel bb), although the origins of this scaling relationship remain unclear. On a longer timescale, polarity sorting of individual filaments, as previously described [58, 68] rearranged the entire network, undoing the initial contraction (see S2 Video).

During the rapid initial contraction, bulk network strain matched closely the mean compressive strain on individual filaments Figure 5.4b, confirming that the origin of bulk contraction in our simulations is filament buckling due to asymmetric filament compliance, as predicted by [38, 37] and observed experimentally[56]. Contraction only occurred when the fractional motor activity $0 < \phi < 1$ (i.e. the fraction of filament intersections with active motors) was less than one, confirming the requirement for dispersion of motor activity (see Tearing of active networks is prevented via recycling. **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 recycling time. **c)** Time trace of internal stresses for network in panel a. **d)** Time trace of internal stresses for network in panel b). Thus, our model effectively captures a minimal mechanism for bulk contractility in disordered networks through asymmetric filament compliance and dispersion of motor activity.

We also determined how microscale parameters shape the rate and final extent of network contraction. Consistent with the idea that contraction stalls when the elastic

resistance to filament compression balances the contractile stress, the final extent of contraction increased sharply with motor activity (υ) and with the asymmetry in filament stiffness (i.e. the ratio of the extensional and compressive stiffnesses μ_e/μ_c, Figure 5.4c,

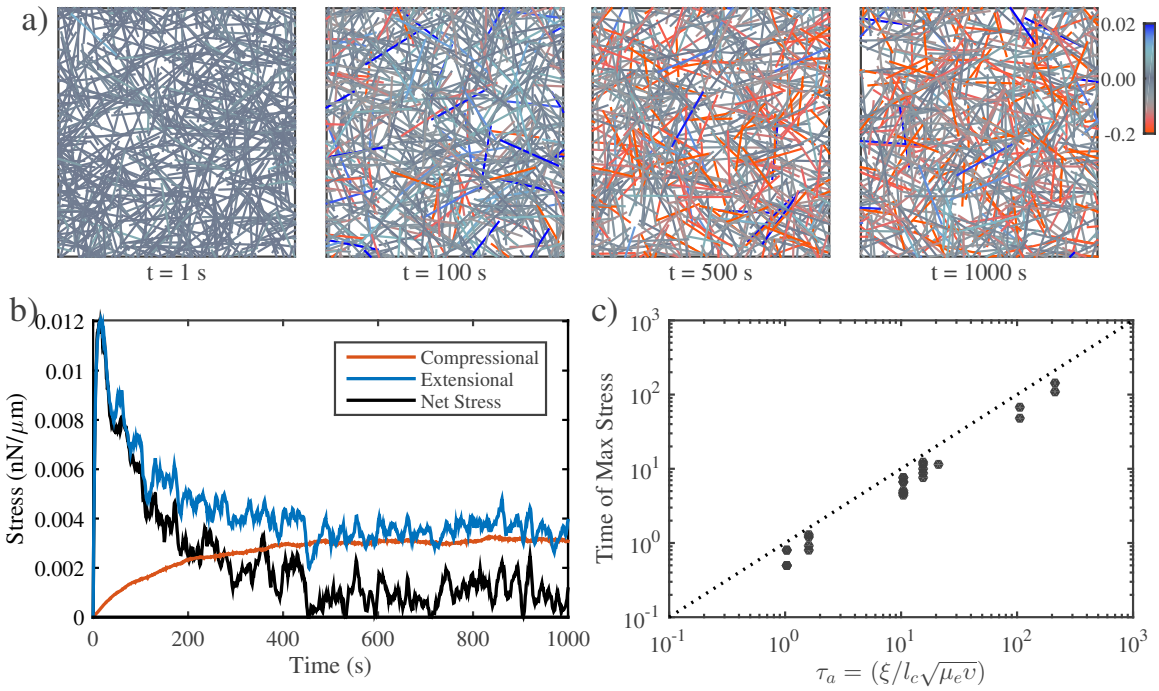


Figure 5.5: In the absence of filament recycling, active networks can only exert a transient force against a fixed boundary. **a)** Simulation of an active network with fixed boundaries illustrating progressive buildup of internal stress through local filament rearrangement and deformation. Note the progressive buildup of compressive stress on individual filaments. Network parameters: $L = 5\ \mu m$, $l_c = 0.3\ \mu m$, $\xi = 100\ nN \cdot s/\mu m$, $\upsilon = 0.1\ nN$. **b)** 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 dissipated at longer times as internal extensional and compressive stresses become balanced. **c**. Measurement and prediction of the characteristic time (τ_a) at which the maximum stress is achieved.

Active networks can only exert a transient stress against a fixed boundary in the absence of filament recycling. The previous results reveal internal

limits on the contraction of networks with free boundaries. However networks typically build force and contract against an external resistance. Therefore, we also 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 (Figure 5.5a), and monitored the time evolution of mean extensional (blue), compressional (red) and total (black) stress on network filaments (Figure 5.5b). We observed the same qualitative behavior for all network parameters examined: Total stress built rapidly to a peak value σ_a, and then decayed back to zero again. Sampling these dynamics over a large range of network parameters, we found that the peak stress occurred at a characteristic time, $\tau_a \sim \xi/l_c\sqrt{\mu_e \upsilon}$, as shown in Figure 5.5c. Although, the origins of this peculiar scaling are unclear, the measurement agrees with our intuitive predictions: The time to reach peak stress should vary directly with the cross-link coupling (ξ) and filament density ($2/l_c$), and it should inversely with the square root of both the filament stiffness (μ_e) and motor force (υ).

The slower falloff of active stress involves two contributions: the first is slow dissipation of extensional stress on network filaments (blue curve in Figure 6B), reflecting a decrease in active stress generation. A similar effect was recently documented in detailed simulations of cross-linked actomyosin networks, where it was associated with large scale reorganization and collapse of network structure [47]. Here, we find that significant dissipation of extensional stress can also occur through more local rearrangements without loss of network structure (Figure 6A,B). The second contribution involves a gradual buildup of compressional stress (red curve in Figure 6B), that balances the active stress at steady state such the the total stress exerted on the network boundary falls to zero. This buildup of compressional stress occurs through purely local rearrangements in the absence of large scale deformation.

We were unable to find a simple dependencies on network parameters either dissipation of extensional stress or buildup of compressional stress. However, for all parameters we sampled, this timescale for decay of stress was significantly longer than the timescale for stress buildup, presumably because the initial buildup involves rapid loading of extensional stress on individual filaments, while the slower dissipation

requires local filament rearrangement.

Filament recycling allows networks to exert sustained stress on a fixed boundary. To explore how the fall-off in stress could be relieved by filament recycling, we again considered an active network contracting against a fixed boundary, using the same parameters as in Figure 5.5, but now we systematically varied filament recycling rates. Adding filament recycling produced two general effects: First, as in the passive case (Figure 4), filament recycling could prevent catastrophic tearing by continuously repairing local structural heterogeneities, and by steadily opposing the effects of local strain thinning (see S3 Fig). Second, we found that filament recycling resulted in biphasic modulation of the level of steady state stress.

For the same network parameters as in Figure 5.6a and slow filament recycling ($\tau_r = 1000s$), the network stress built rapidly, peaked, and then fell to a lower value that persisted for times much longer than τ_a. Up to a certain point, decreasing recycling time produced a monotonic increase in the steady state stress, although the steady state stress remained lower than its peak value. However, beyond this point, further decreases in recycling time lead to decreases in the steady state stress as well as sharp decreases in the peak stress. IWe reasoned that this bimodal dependence of steady state stress on recycling rates emerges from continuous replacement of strained with unstrained filaments, combined with the different timescales for buildup of extensional vs the slower dissipation of extensional stress and buildup of compressive stress (Figure 5.6b). If so, then lowering the recycling time τ_c should increase net stress until τ_c is approximately equal to τ_a, the time required to build peak stress from an initially unstressed state. For shorter recycling times, the average filament will not have time to build maximum extensional stress before turning over, and thus the steady state stress should decrease with further decreases in τ_c. Indeed, plotting normalized steady state stress (steady state stress/peak stress) vs normalized recycling time (τ_c /τ_a) confirmed that this biphasic dependence of steady state stress on recycling times holds for a large range of sampled values for network parameters 5.6d.

Similar to the passive response (i.e. Equation 5.3), we can approximate the dependence of the steady state stress on the filament recycling rate using a simple

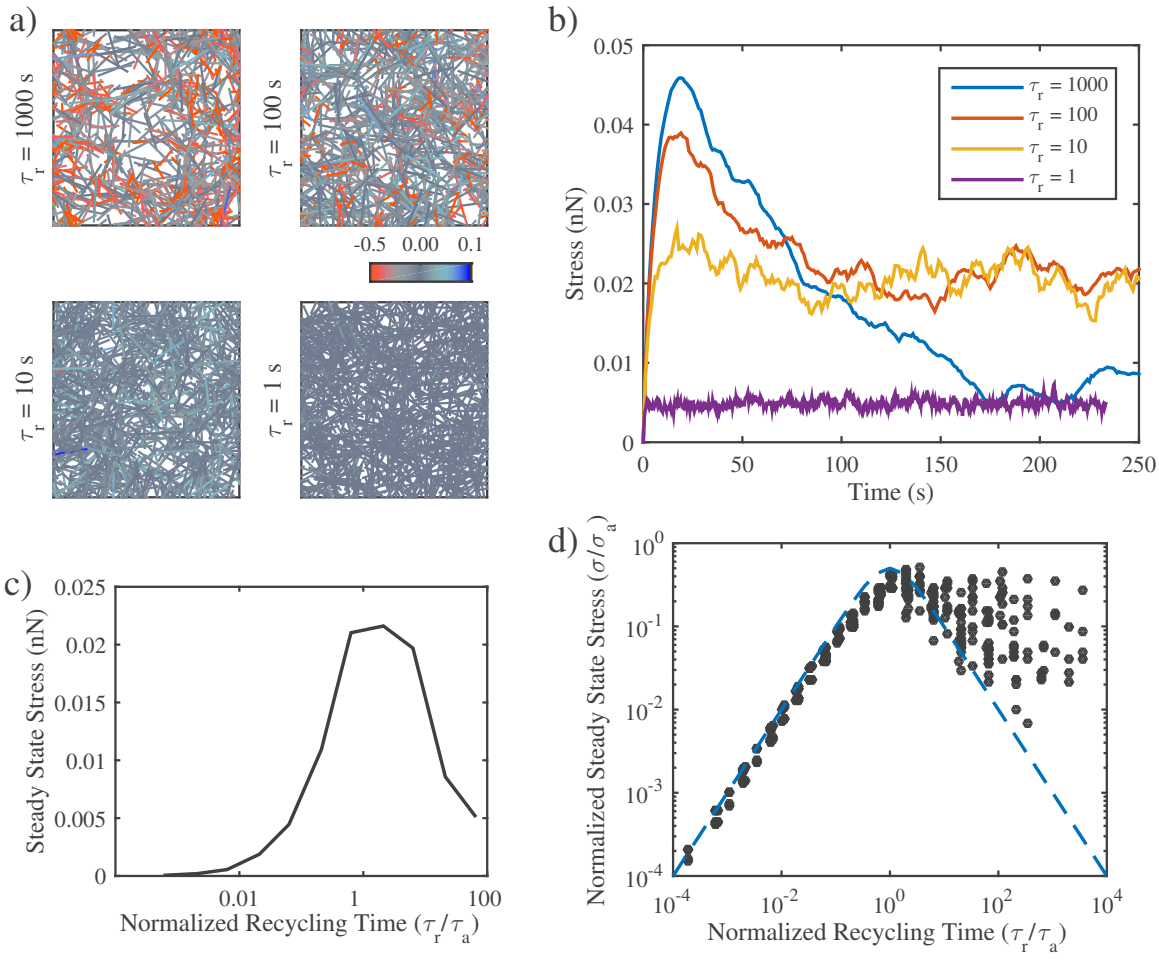


Figure 5.6: Filament recycling allows network to exert sustained stress on a fixed boundary. **a)** Snapshots from simulations of active networks with fixed boundaries for different timescales of filament recycling. Network parameters are the same as in Figure 6. Note that significant remodeling occurs for longer recycling 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. Increasing filament turnover rates reduces stress dissipation by recycling compressed filaments; however, very short recycling times prevent any stress from being built up in the first place. **c)** Plotting the steady state stress derived from the long term stress values of the stress in panel b. **d)** Normalized steady state stress as a function of normalized recycling time. The steady state stress is set by the timescale at which the network strain is refreshed relative to the timescale at which the max stress is reached. The values have been normalized to the predicted peak stress, σ_a in the absence of recycling. Blue dashed line indicates the approximation given in equation 5.4 for $n = 1$.

equation.

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

For $\tau_r \gg \tau_a$, this simplifies to $\sigma_{ss} \sim (\tau_a/\tau_r)^n$, while for $\tau_r \ll \tau_a$, this simplifies to $\sigma_{ss} \sim \tau_r/\tau_a$. What sets the scaling n remains unclear, and this scaling does not appear to be consistent across all simulation setups (Figure 5.6d). However, equation 5.4 still captures a qualitatively correct description of steady state stress in our simulation data.

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

Thus far, we have considered independently how filament recycling tunes effective viscosity during passive deformation in response to an externally applied stress, and how filament recycling tunes the steady state stress produced by an active network against an external resistance. We next sought to characterize how filament recycling tunes the steady flows produced by gradients of motor activity as regions of high motor activity contract against the passive resistance of a neighboring region with low motor activity.

Filament recycling allows sustained flows in networks with non-isotropic activity. We imposed a continuously asymmetric distribution of motor activity on an initially uniformly dense network of passively cross-linked filaments by allowing a fraction of cross links to be active only in the right half of the simulation domain. Then we examined the time-dependent deformation of the network for a range of different filament recycling times Figure 5.7a. We observed a sharp dependence of steady flow on filament recycling rate Figure 5.7b,c. For very longer 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 Stress and strain profiles of networks

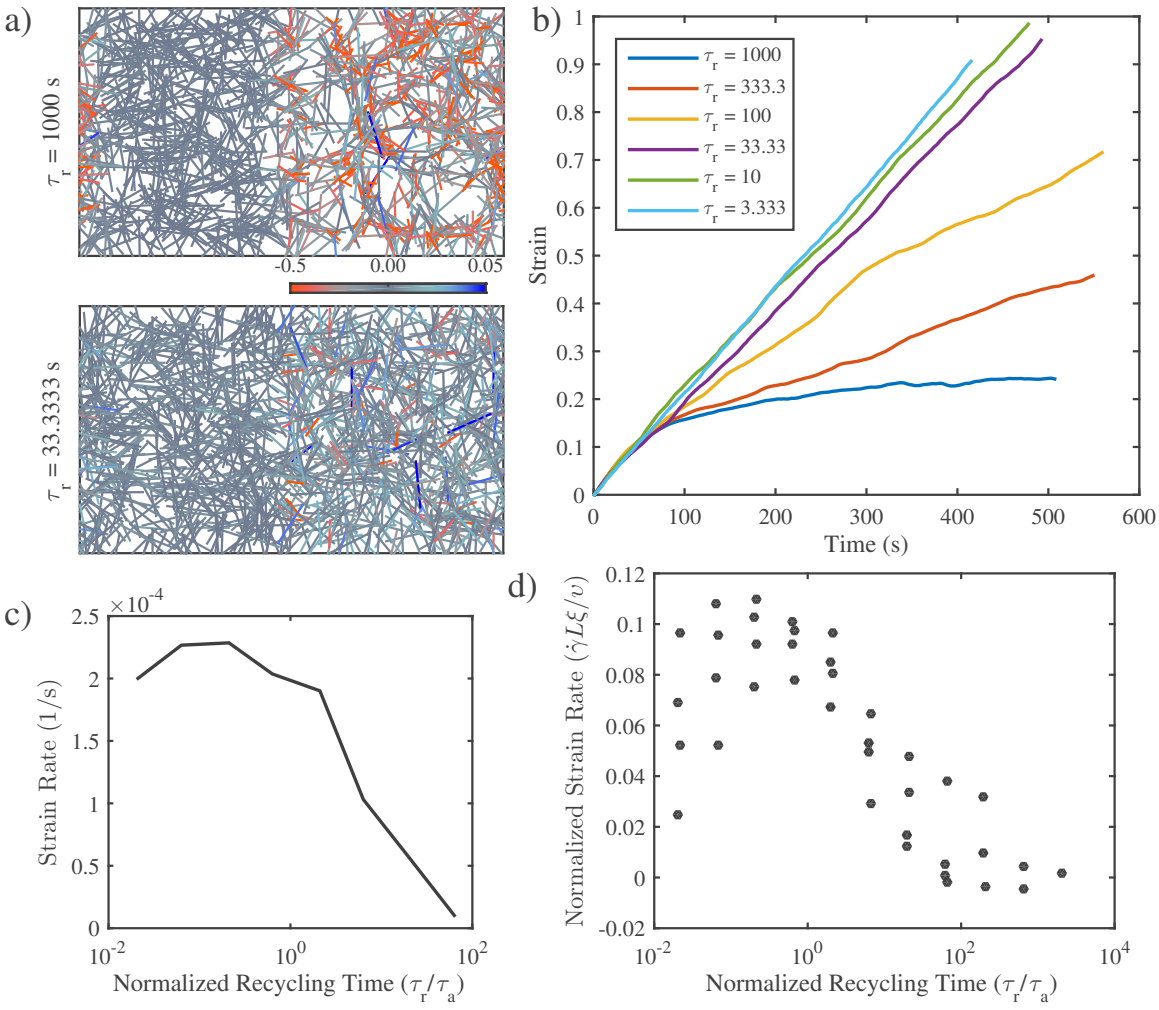


Figure 5.7: Filament recycling allows sustained flows in networks with non-isotropic activity. **a)** Example simulations of non-isotropic networks with long ($\tau_r = 1000$) and short ($\tau_r = 33$) recycling timescales. In these networks the left half of the network is passive while the right half is active. Network parameters are same as in Figures 5.5 and 5.6. Importantly, in all simulations $\tau_a < \tau_c$. **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. **c)** Graph of network long-term strain rate as a function of recycling timescale for simulations in a) and b). **d)** Graph of network long-term strain rate as a function of recycling timescale across a wide range of parameter space. Note that networks only begin to maintain long-term flows when the recycling time is less than $100\tau_a$.

with contractile and passive domains. **a)** Blue line indicates strain velocity profile while orange represents net stress as measured in the main text. **b)** Same as panel a except for the condition where recycling time is 10 s. Note the increase in net stress and the corresponding increase in flow rate). However, with decreasing filament recycling times, we found the network was able to largely sustain its deformation and that the long term strain rate remained relatively high (Figure 5.7c). We repeated these measurements for more network parameters and found that at the shortest recycling timescales measured, we still saw the effective viscosity remaining relatively high, indicating that for short recycling times the effective viscosity may be somewhat buffered against variation in recycling times (Figure 5.7d).

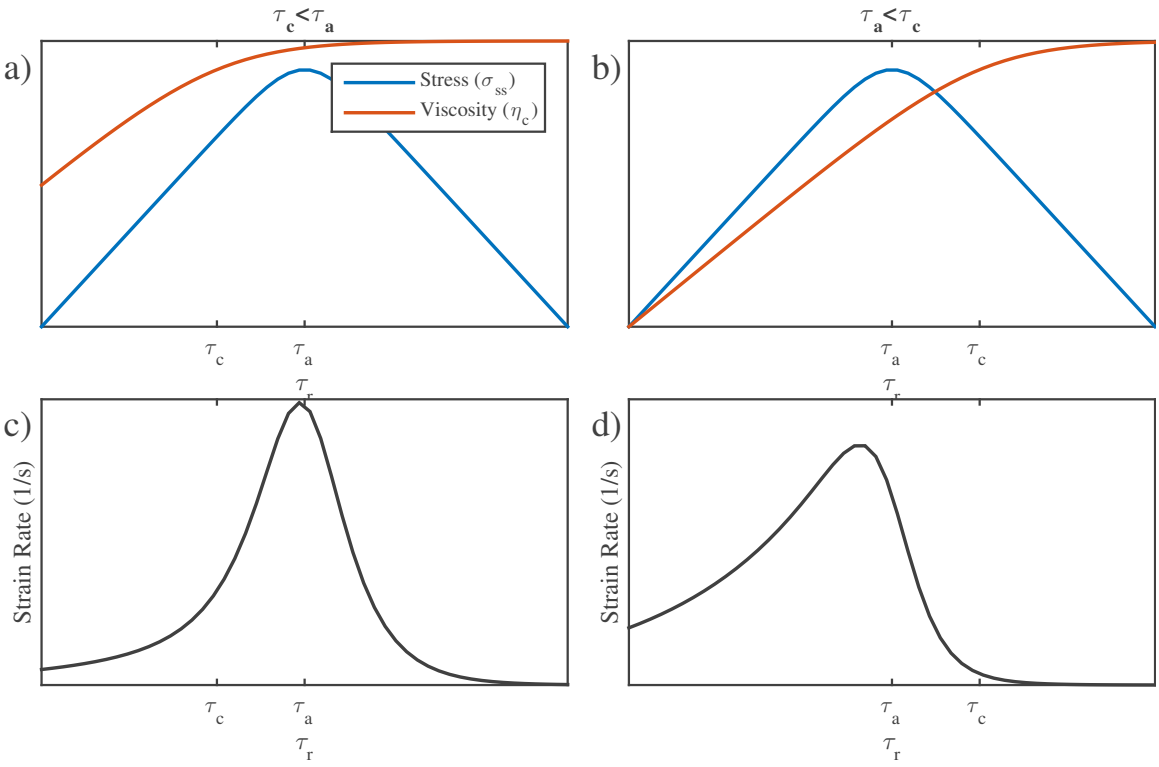


Figure 5.8: Filament recycling tunes the magnitudes of both effective viscosity and steady state stress. **a)** Dependence of steady state stress and effective viscosity on recycling time τ_r under the condition $\tau_c < \tau_a$. **b)** Same as a), but for the case where $\tau_a < \tau_c$. **c,d)** Resulting strain rates for network as a function of recycling time τ_r for the regimes in panels a and b..

Filament recycling tunes the magnitudes of both effective viscosity and steady state stress. This dependence of steady state deformation (flow) rate on filament recycling times can be understood in terms of our previous findings. During steady state flow, active contraction of the right half-domain is limited both by internal resistance to compression of filaments within the right half-domain (Figure 5), and by passive resistance of the left-half domain (Figure 4).

Monitoring these two forms of resistance as a function of filament recycling time for the simulations in Figure 8, we see that resistance to compression of filaments in the right half domain makes a significant contribution only for very low recycling rates. This is because using physiologically relevant values for network parameters (described above) sets up a condition where compressional resistance on filaments in the contracting right half domain takes longer to build than extensional resistance on filaments in the dilating right half-domain (i.e. $\tau_c < \tau_s$). As a consequence, except for very low recycling rates ($\tau_r > \tau_{crit}$), steady state deformation is governed by an equation of the form:

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

where σ_{ss} is the active stress generated by the right half-domain (less the internal resistance to filament compression), η_r is the effective viscosity of the left half domain and strain rate is measured in the left half-domain. Therefore, we can understand the dependence of network flow (i.e. strain rate) on filament recycling time τ_r in terms of the previously characterized dependencies of effective viscosity and steady state stress on τ_r (Figures 5.3d and 5.6d). In particular, recall that there is a transition in the dependence of η_r on τ_r at the characteristic time τ_c, and a transition in the dependence of σ on τ_r at the characteristic time τ_a. Thus, as shown in Figure 5.8, there are two qualitatively distinct cases for the dependence of strain rate on τ_r, depending on the relative magnitudes of τ_a and τ_c. For both cases, we expect a decrease in strain rate with filament recycling at long recycling times (where effective viscosity is insensitive to strain rate) and approach to a slowly varying strain rate at low recycling times, where both η_r and σ_{ss} fall off with different scalings. For $\tau_a < \tau_c$, we predict a

peak strain rate at intermediate recycling times followed by a rapid falloff at lower recycling times, whereas for $\tau_a > \tau_c$, we expect a more rapid approach to maximum strain rate and a slower fall off at lower recycling times. As shown in Figure 5.8d, for the range of network parameters we sampled, the strain rate rapidly increases as τ_r is lowered towards τ_a and then more slowly decays to 0 as τ_r is further decreased. This is to be expected because all the parameter values sampled (selected for physiological relevance) satisfied the condition $\tau_a > \tau_c$.

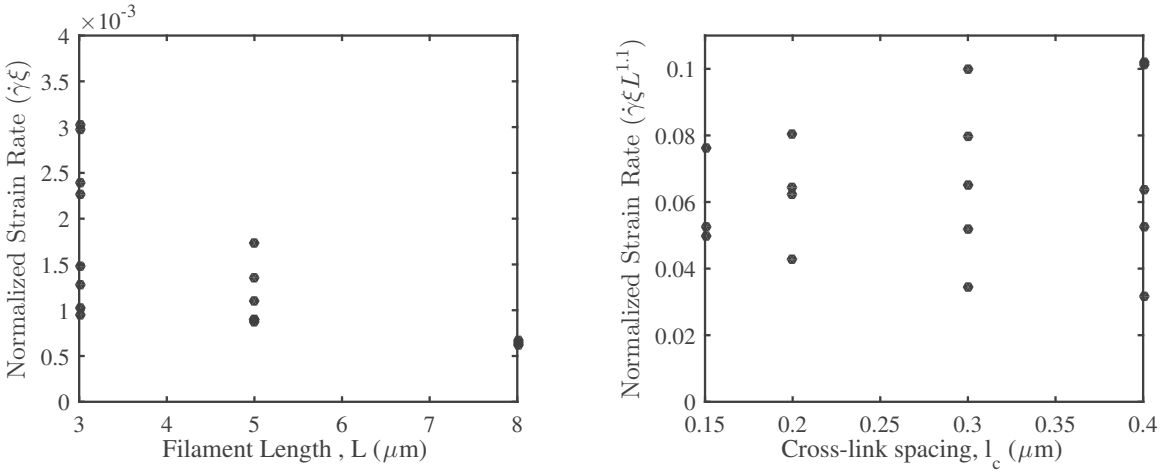


Figure 5.9: Filament recycling influences architectural control of flow rate. **a)** For a fixed filament recycling time, filament length tuned network deformation rate. **b)** Recycling rate is independent of cross-link spacing in this parameter space.

Filament recycling influences architectural control of flow rate. Finally, we examined how steady state flow depends on other network parameters when filament recycling rates are held constant. Interestingly, we found that flow rates are largely insensitive to cross link density (l_c) but vary inversely with filament length. Thus, in this region of parameter space, steady state flows may be buffered intrinsically against some forms of variation in network architecture.

5.2 Conclusion

Our work aimed to create a simulation framework that would allow us to analyze the origins of macroscopic flow in terms of a handful of physiologically relevant microscopic parameters. Toward this aim we developed a minimalist model of a 2D filament network and analyzed the network's reaction to a variety of situations. We found mathematical relationships that determined both the passive effective viscosity and the active stress generation of networks with and without recycling. From these relationships we were able to make predictions about the rates of network flow in non-isotropic networks mimicking those found in polarized eukaryotic actomyosin cortices.

Importantly, our work brings a theoretical understanding to the importance of actomyosin turnover in producing and maintaining long-term large scale flows. We propose the concept of filament recycling to refer to the multitude of biochemical interactions which can give rise to the piece by piece architectural resetting of filament networks. We believe that our analysis of networks in the presence of this filament recycling will be useful in further developing the qualitative and quantitative understanding the deformation of these complex networks.

5.3 Supporting Information

S1 Text. Bold the title sentence. Add descriptive text after the title of the item (optional).

S1 Fig. Bold the title sentence. Add descriptive text after the title of the item (optional).

S2 Fig. Mechanical properties of passive networks. a) Elastic modulus of networks. Our measurements closely match prediction of $G_0 \sim \mu/l_c$. **b)** Placeholder for inevitably another figure relevant to passive properties..

S3 Fig. Mechanical properties of active networks Add descriptive text after the title of the item (optional).

S4 Fig. Mechanical properties of active networks Add descriptive text after the title of the item (optional).

S6 Fig. Spatial velocity profile of networks containing passive and active domains.

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

S1 Video. Extensional strain in passive networks. Movie of simulation setup shown in Figure 5.1

S2 Video. Active networks contracting with free boundaries. Movie of simulation setup shown in Figure 5.4

Table 5.1: Simulation Parameter Values

Parameter	**Figure 3**	**Figure 4**	**Figure 5**	**Figure 6**	**Figure 7**	**Figure 8**
L	$1, 3, 5, 7, 10$	3	5	5	5	5
l_c	$0.2, 0.3, 0.5, 0.8$	$0.3, 0.5$	0.3	$0.2, 0.3$	$0.2, 0.3$	0.4
μ_e/μ_c	100	100	$3 - 300$	100	100	100
μ_c	0.01	0.01	$0.01 - 0.3$	0.01	0.01	0.01
ξ	$10, 100$	$5, 10, 100$	$1, 10, 100$	$10, 100$	$10, 100, 330$	$10, 100$
υ			$0.1, 0.3, 1$	$0.1, 1$	$0.1, 1, 3$	$0.1, 1$
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	$0.0002 - 0.01$	$0.00003 - 0.005$				

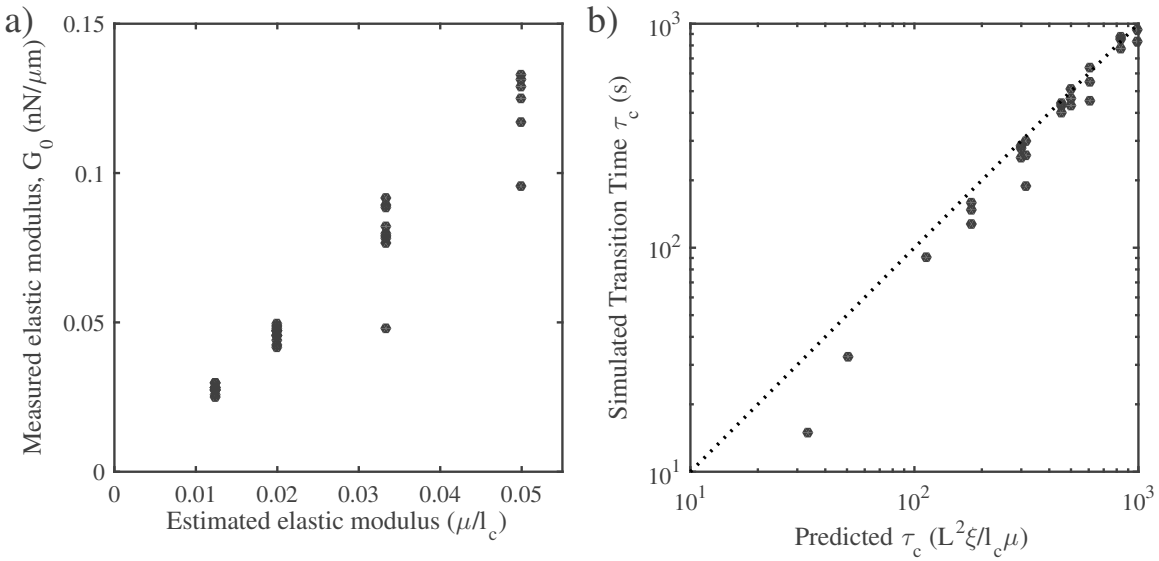


Figure 5.10: Mechanical properties of passive networks. **a)** Elastic modulus of networks. Our measurements closely match prediction of $G_0 \sim \mu/l_c$. **b)** Placeholder for inevitably another figure relevant to passive properties.

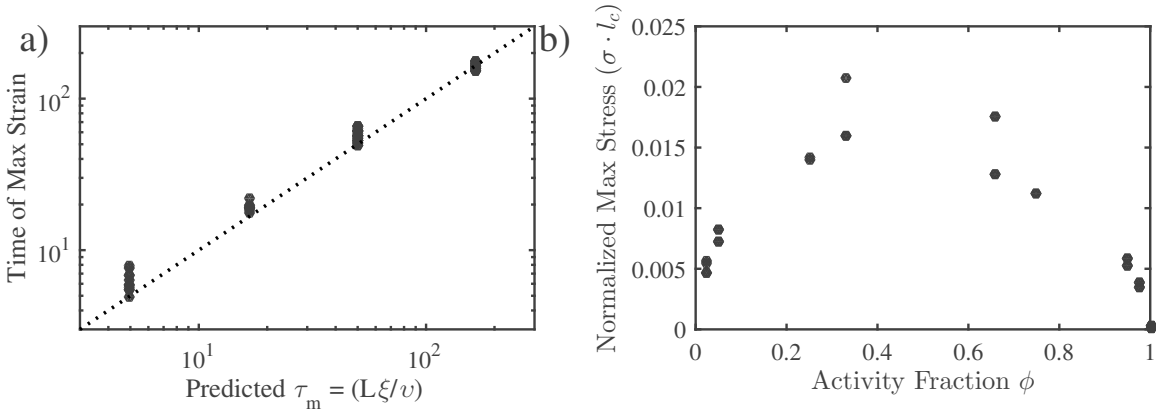


Figure 5.11: Mechanical properties of active networks. **a)** Timescale of maximum strain in networks free to contract. This relationship was found phenomenologically. **b)** Dependence of network stress on the fraction of cross-links which are active. Note that the network stress approaches 0 as ϕ approaches 0 or 1.

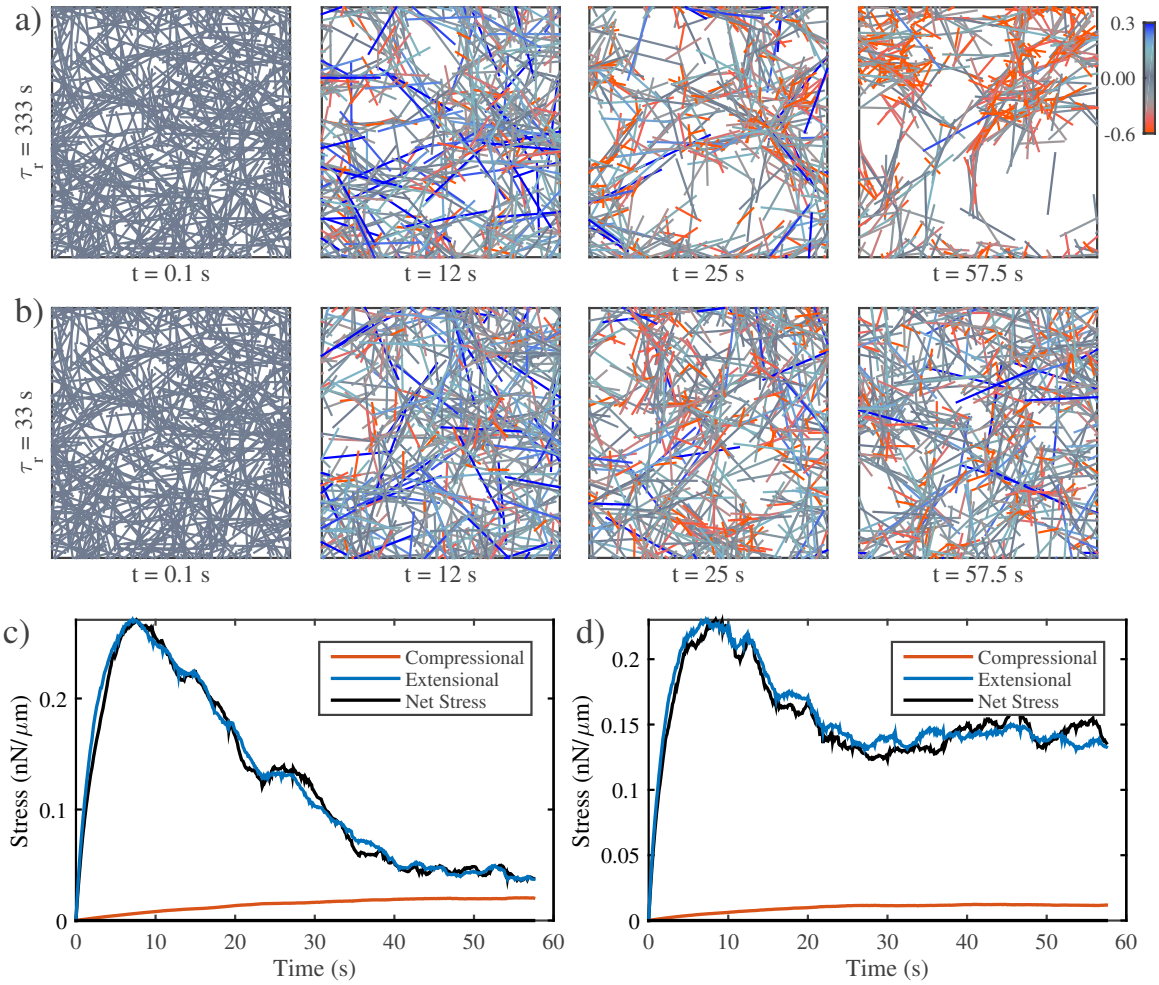


Figure 5.12: Tearing of active networks is prevented via recycling. **a)** An active network undergoing large scale deformations due to active filament rearrangements. **b)** The same network as in a) but with a shorter filament recycling time. **c)** Time trace of internal stresses for network in panel a. **d)** Time trace of internal stresses for network in panel b.

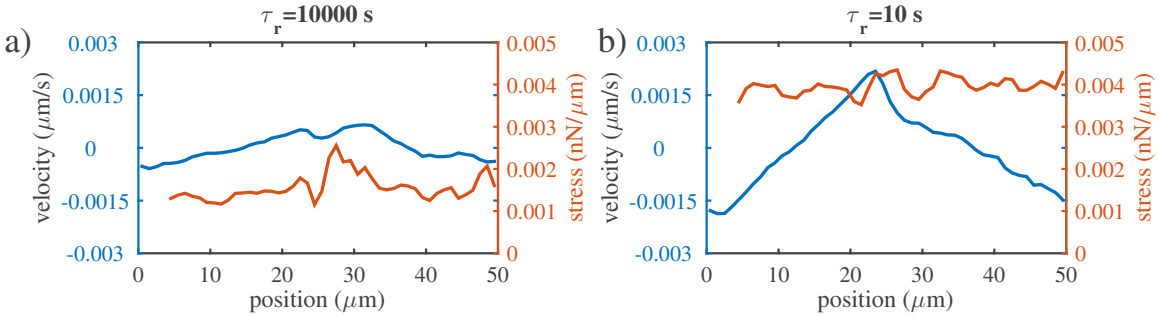


Figure 5.13: Stress and strain profiles of networks with contractile and passive domains. **a)** Blue line indicates strain velocity profile while orange represents net stress as measured in the main text. **b)** Same as panel a except for the condition where recycling time is 10 s. Note the increase in net stress and the corresponding increase in flow rate.

CHAPTER 6
A MODEL OF UPSTREAM ACTOMYOSIN REGULATORS IN PULSED CONTRACTIONS

6.1 Motivation and experimental context

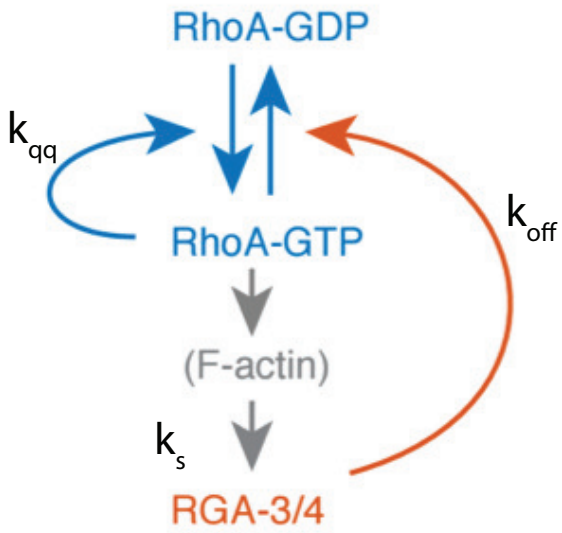


Figure 6.1: Reaction pathway with labels for coefficients associated with feedback stregths.

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).

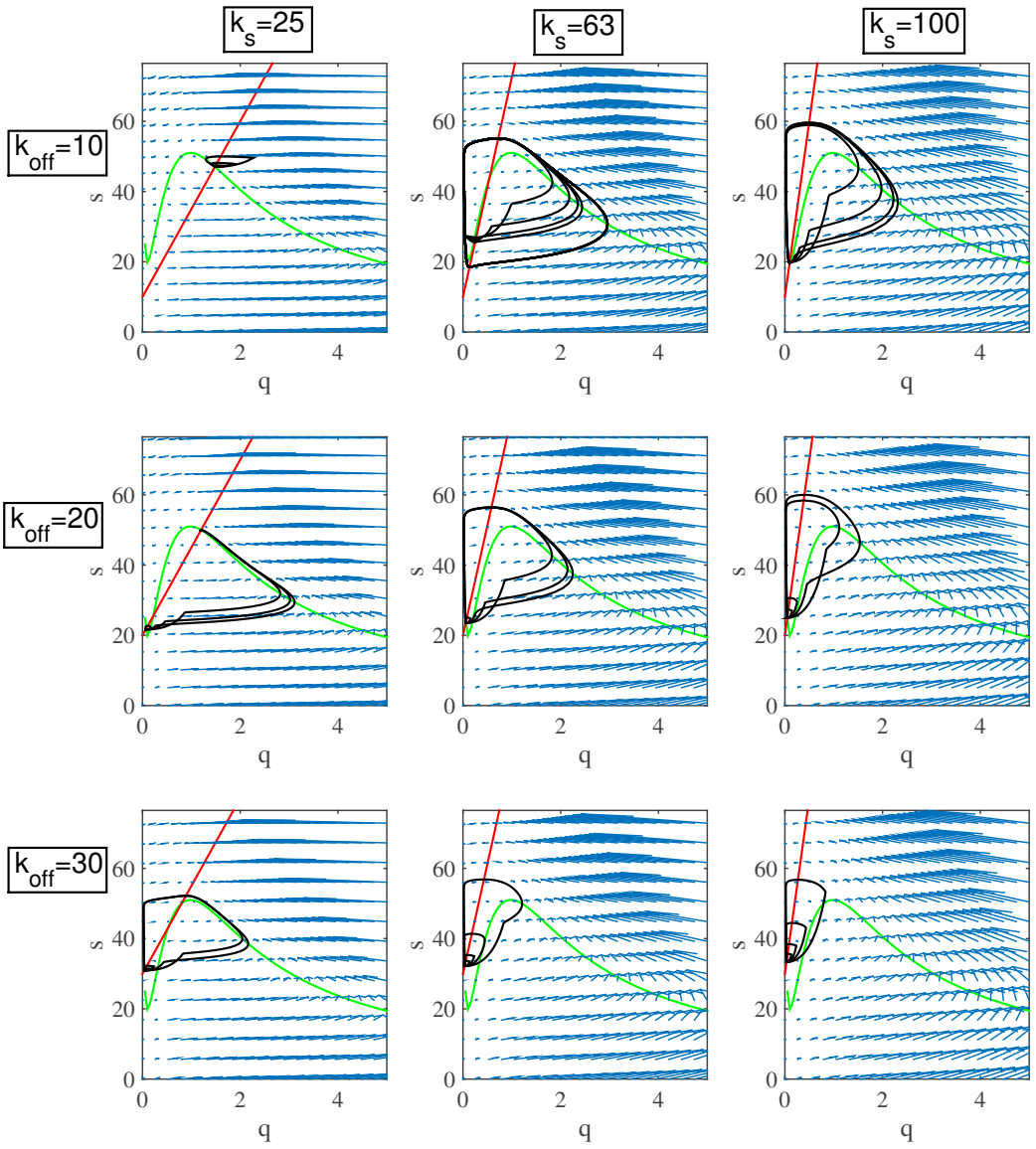


Figure 6.2: Perturbation simulation results plotted on phase planes for a variety of parameters.

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

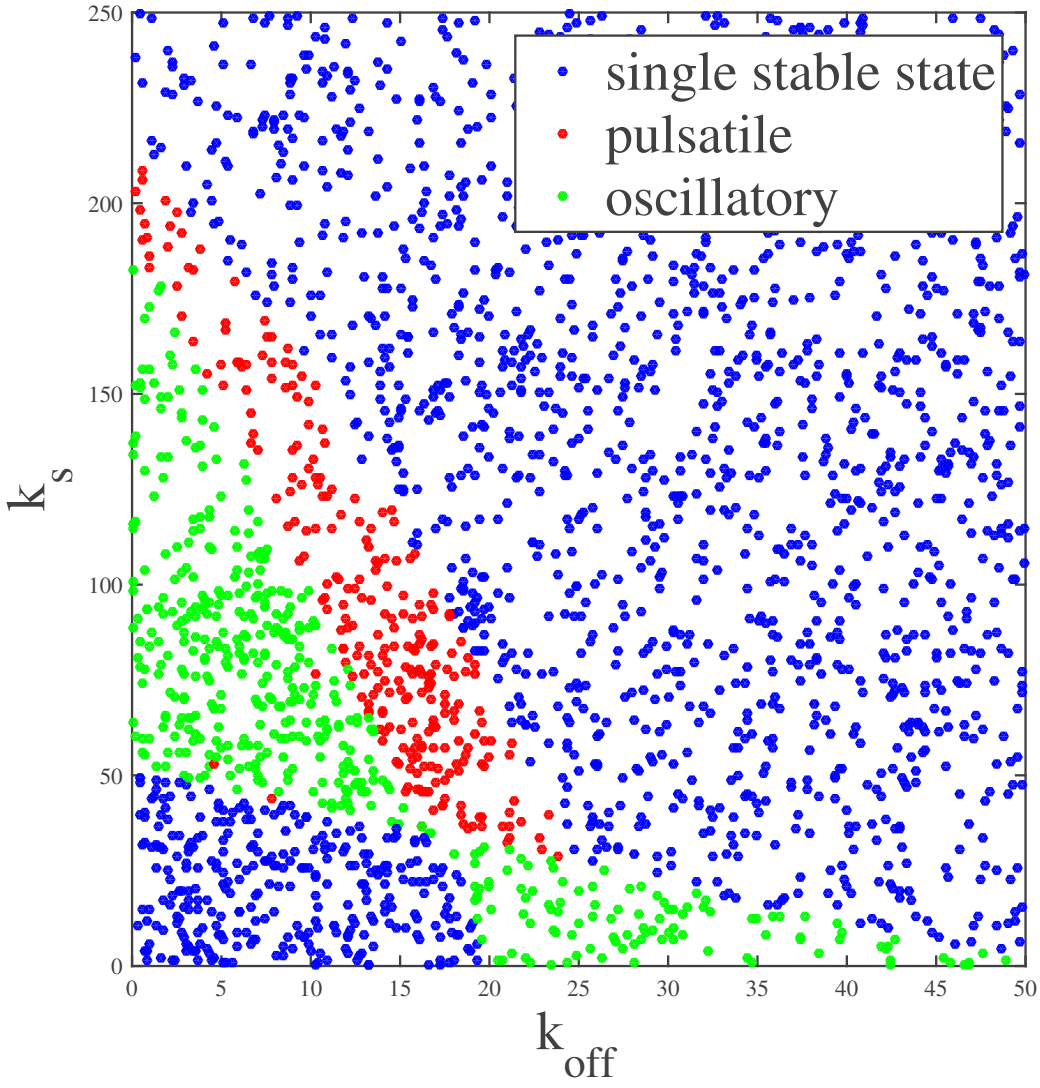


Figure 6.3: Phase diagram of for $k_q = 1$ and $k_q q = 100$.

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

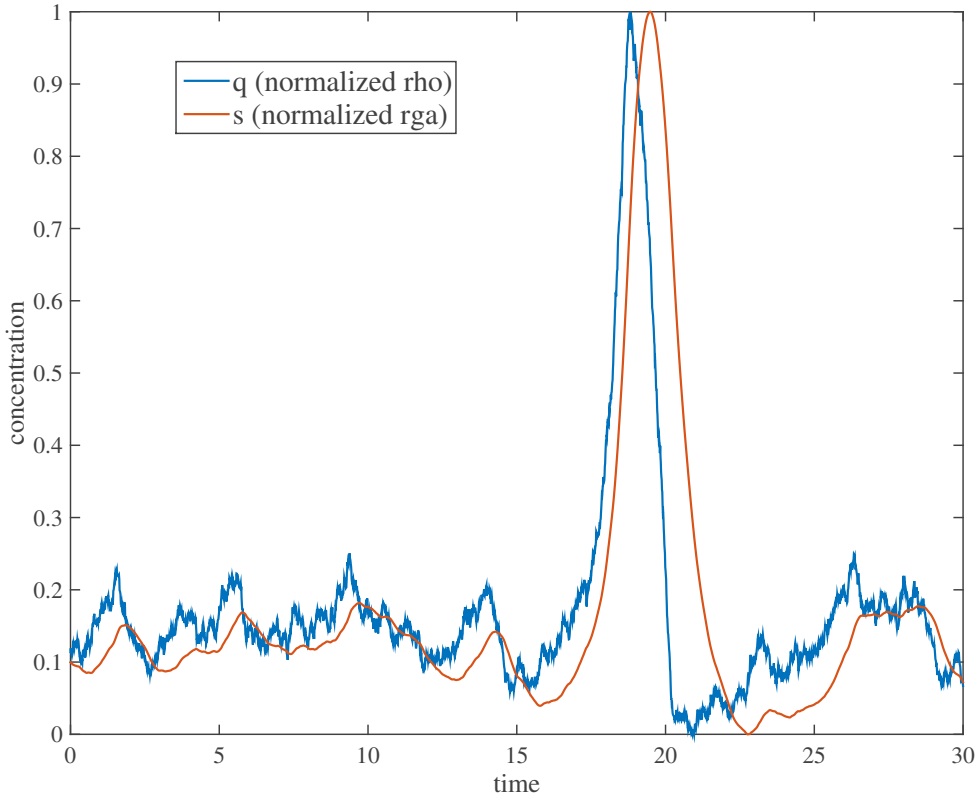


Figure 6.4:

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.

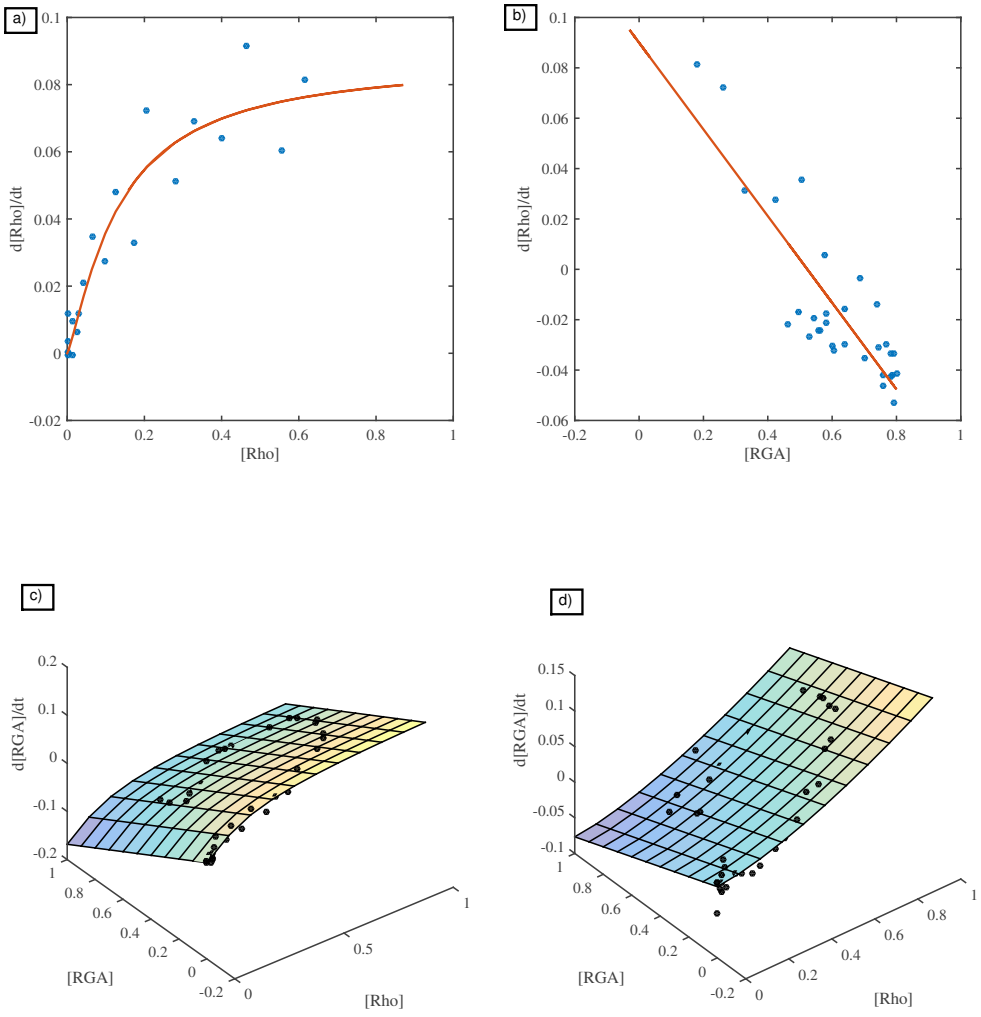


Figure 6.5: Multiple methods of fitting.

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

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

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

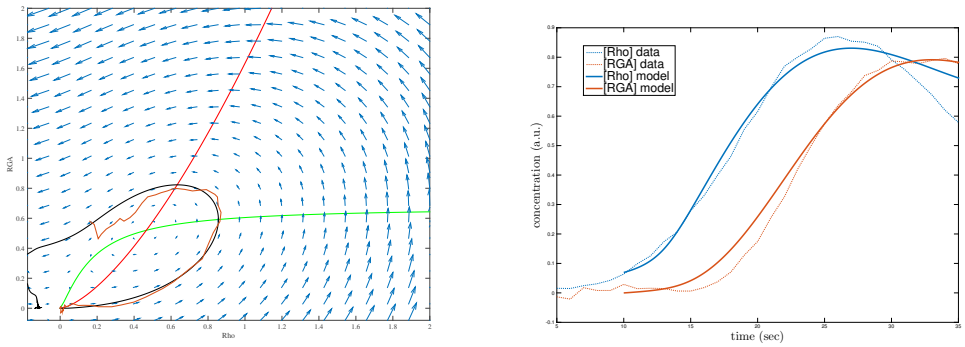


Figure 6.6: Simulation results and fitted data for model with $m = 2$ and $k = 0$. Displaying simulation results and the data used to fit the simulation on a phase plane (left) and as pairs of time plots (right).

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.

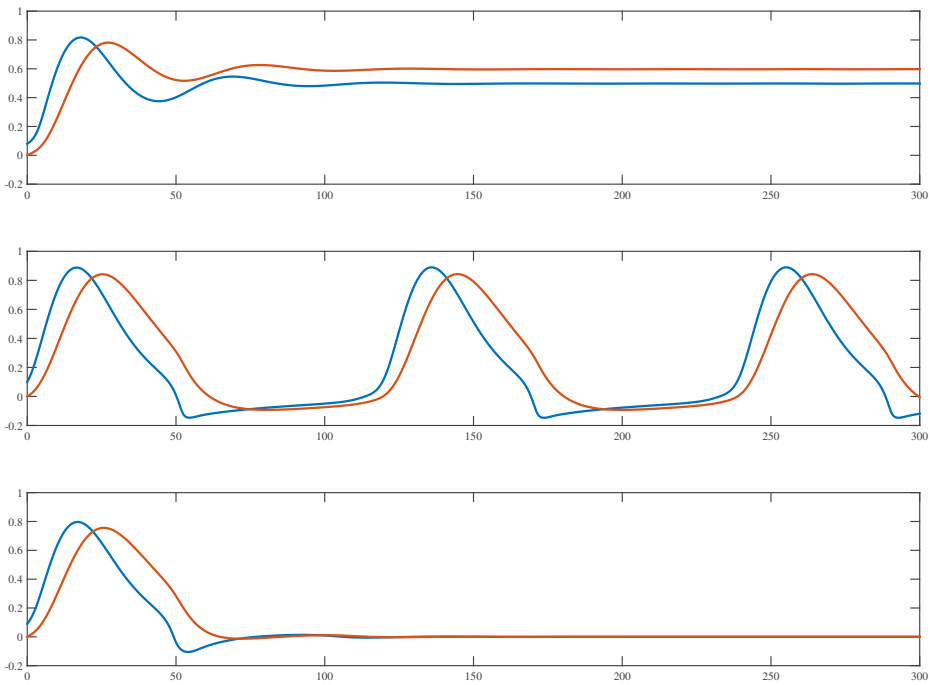


Figure 6.7: Small variation in simulation parameters can have large qualitative effects.

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.

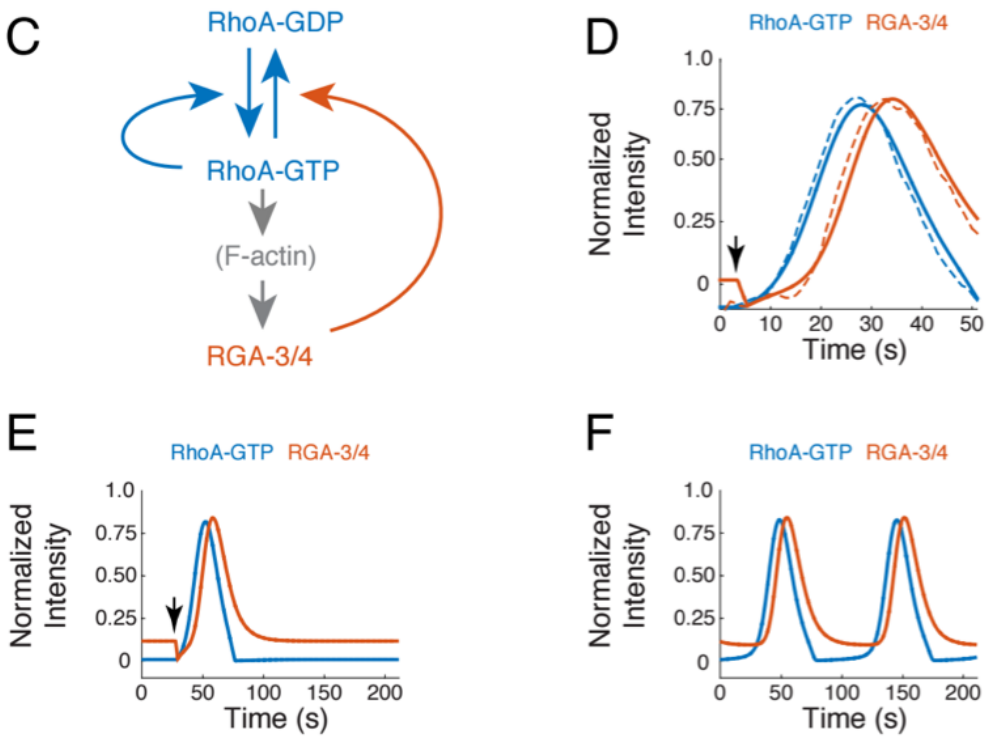


Figure 6.8: Final figure as it appears in Robin et al.

CHAPTER 7
FUTURE DIRECTIONS

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

7.2 Probing other relaxation timescales in more detail

7.3 Incorporating bending degrees of freedom

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:

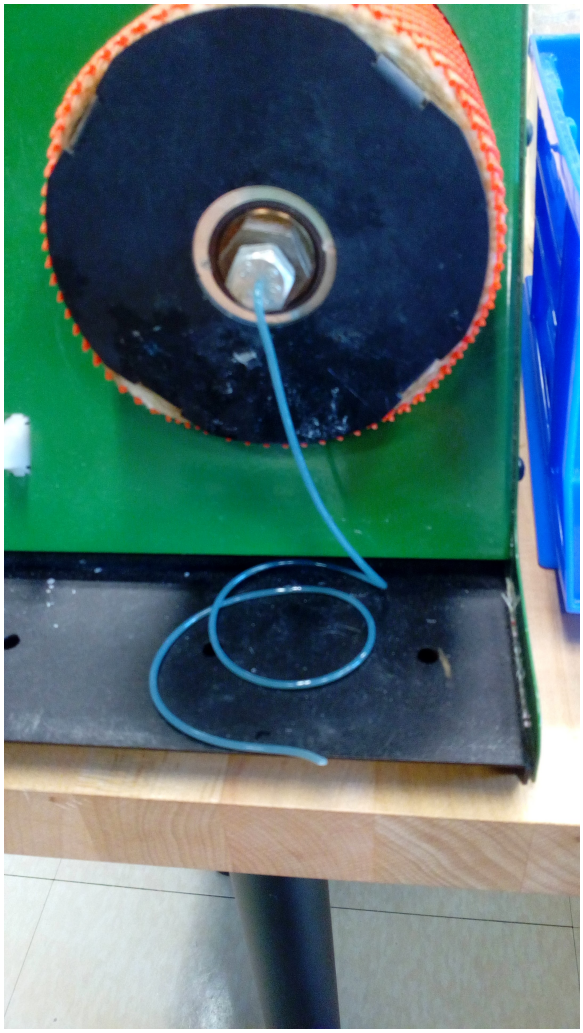


Figure A.3:

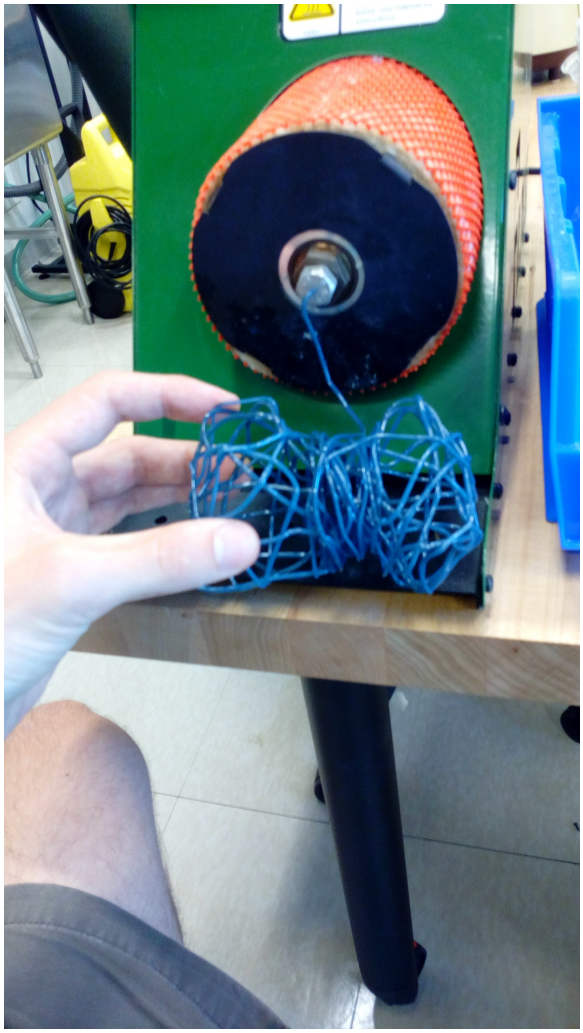


Figure A.4:

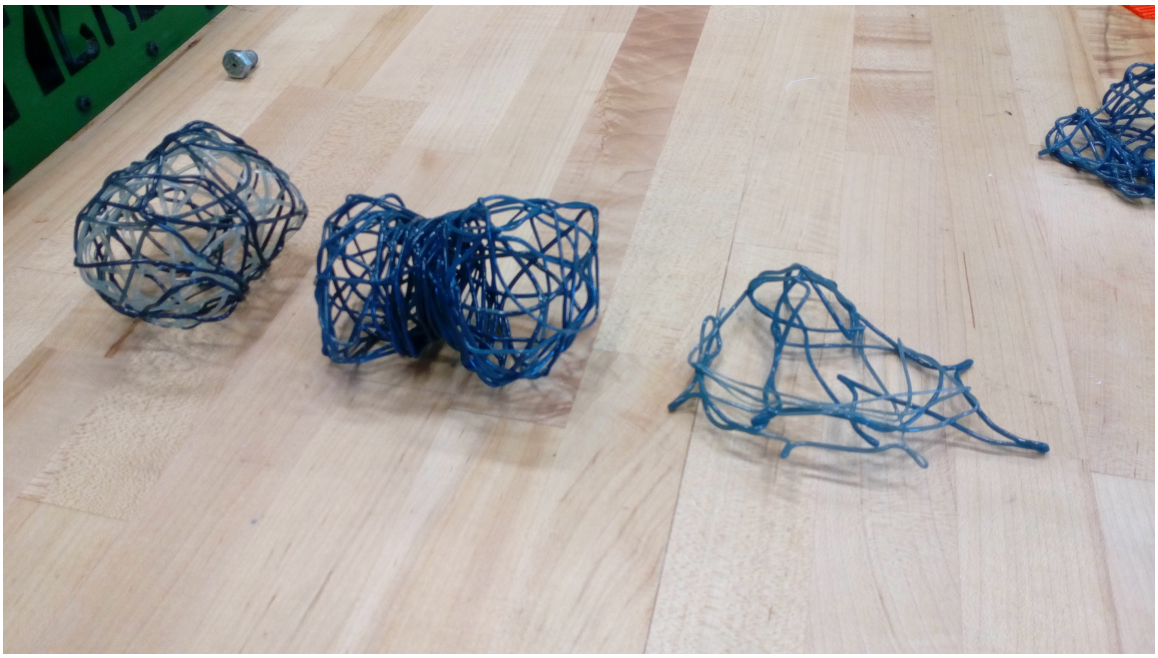


Figure A.5:

Figure A.6:

Figure A.7:

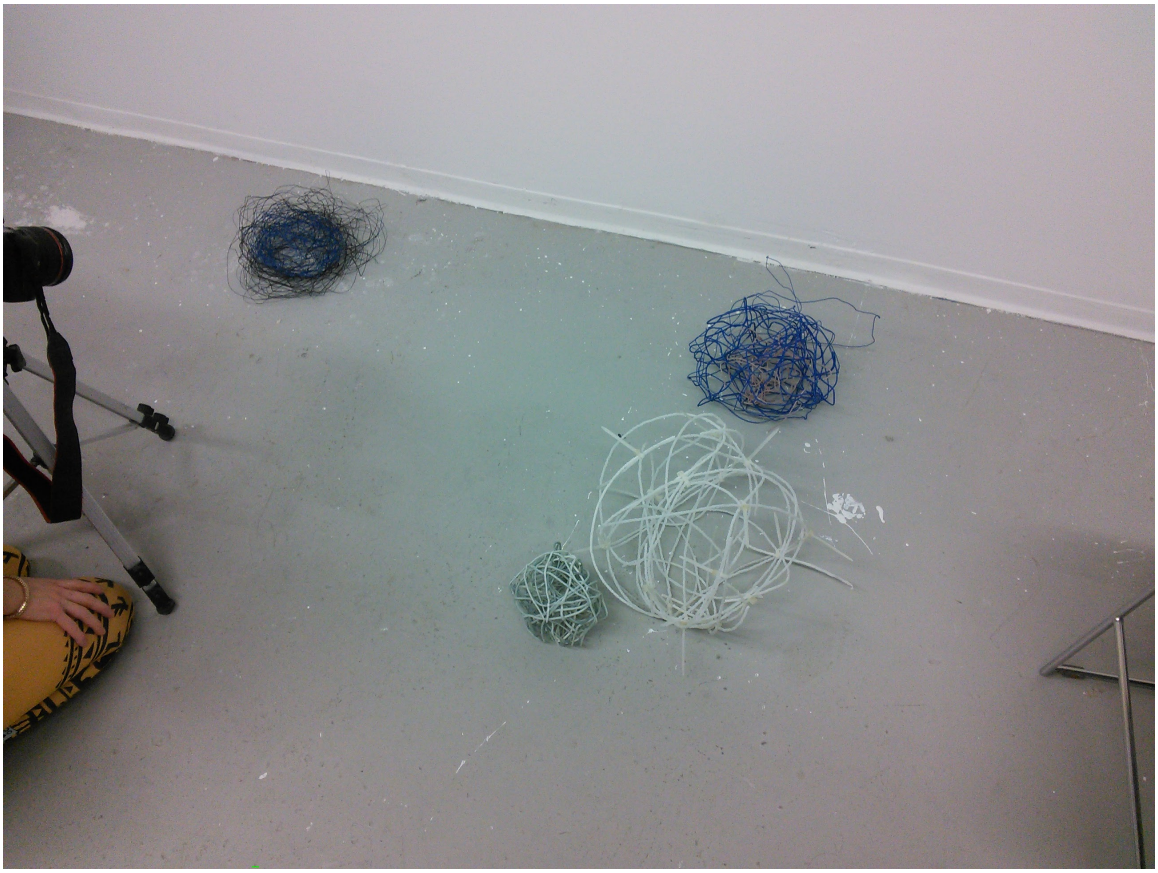


Figure A.8:

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

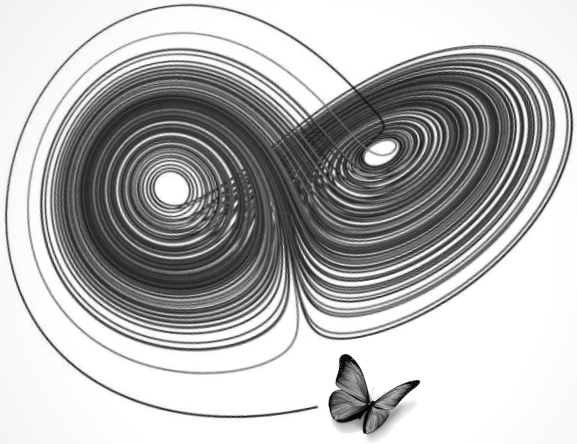
In general, the students reacted positively to the framework that we established with it's emphasis on the three C's outlined above. The student survey responses showed that the focus on classroom participation was welcome. In practice, the students were not required to complete any homework, and the 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”

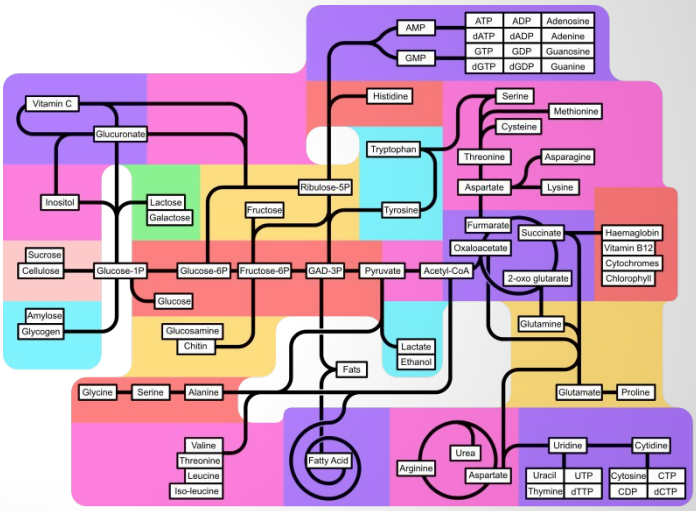
Collect: Part of what we are doing here is learning how to collect information. Every class starts with a simple content quiz over the short reading, followed by a 15 minute review of the day’s focus content.

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

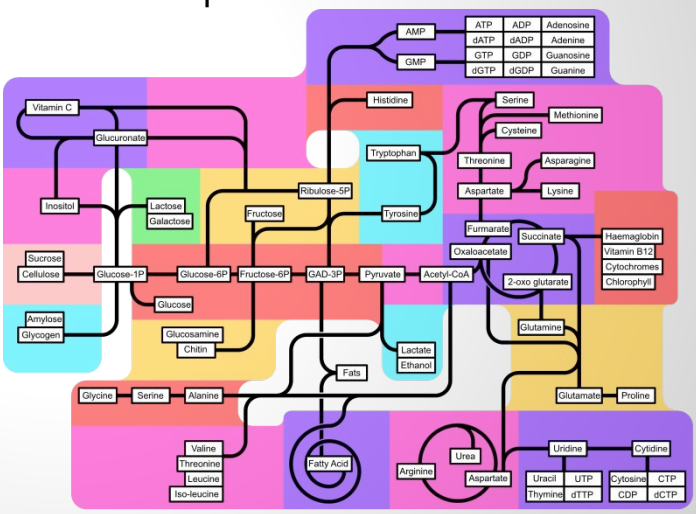


Biology is hard enough on its own. Why study mathematical modeling too?

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

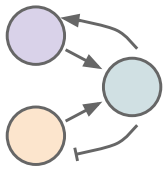
3. Models can clarify key components of reaction pathways

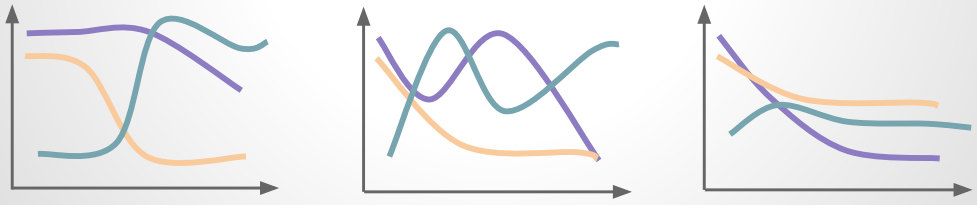
Wouldn't you like to tease apart a parasite's core metabolic pathway to find drug targets using just a system of differential equations?



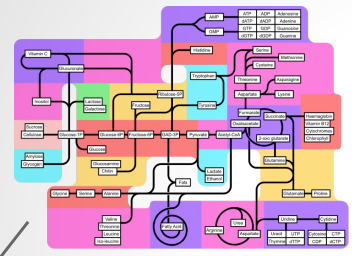
2. Mathematical models make predictions that cartoon models can't

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

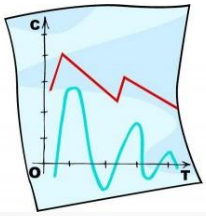




1. Mathematical models aren't that hard



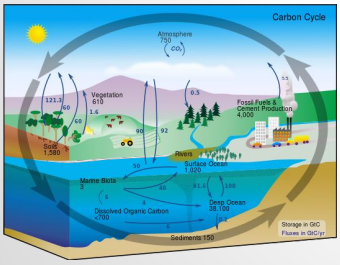
$$\frac{dc}{dt} = k \cdot a$$



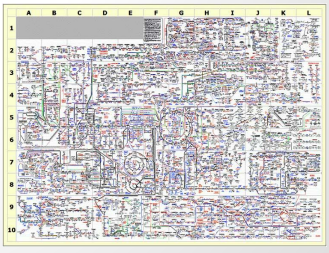
The balancing act behind scientific models

Complex Problem

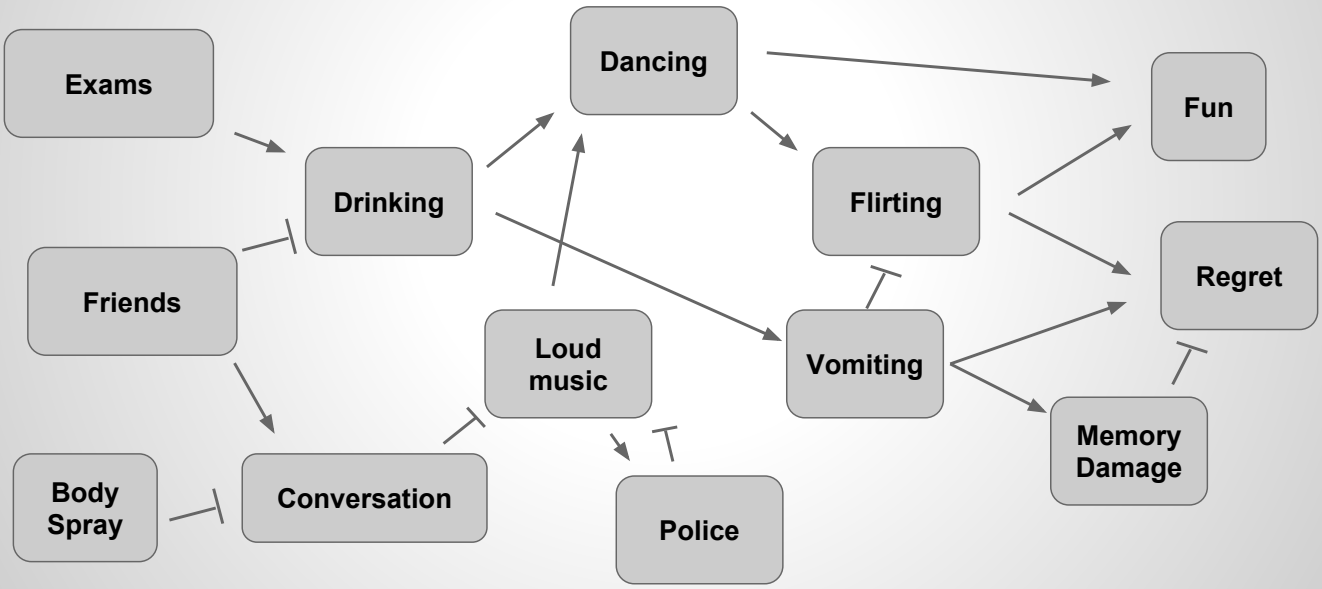
Simplification



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

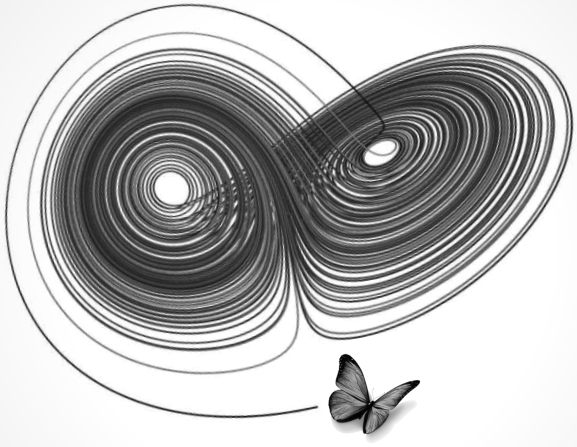
Modelling biological systems is a significant task of systems biology and mathematical biology. It involves the use of computer simulations of biological systems, including cellular subsystems (such as the networks of metabolites and enzymes which comprise metabolism, signal transduction pathways and 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.

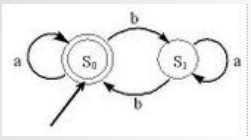
What is a differential equation good for? A differential equation is a simple formula that describes how a system will change in time based on the state of the system right now. You can think of it like a very formal protocol for how concentrations of substances will change over time. For example, a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

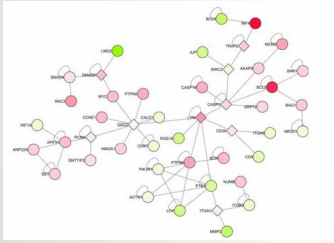
Biology and Dynamical Systems



Week 2: Modeling biological systems with differential equations

Mathematical modeling of biological systems can mean many different things



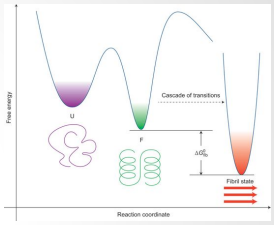


Discrete stochastic

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:

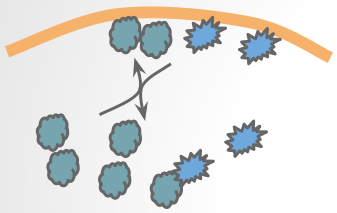
Describing rates of change

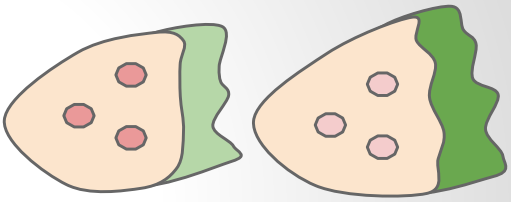
How to interpret reaction diagrams

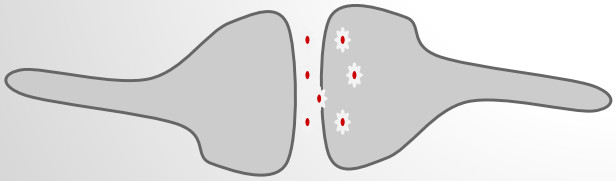
Converting reaction diagrams to differential equations

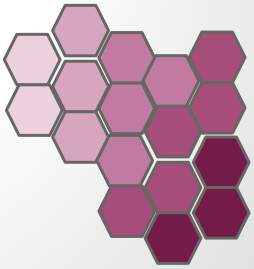
Examples

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

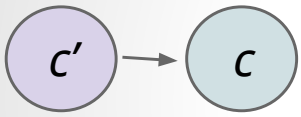




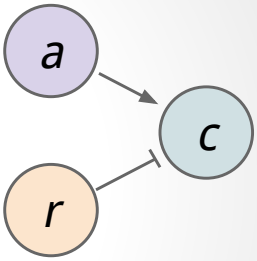




We use reaction diagrams to illustrate how protein levels change based on other cellular components



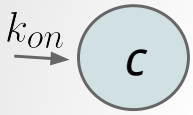
c' is converted into c



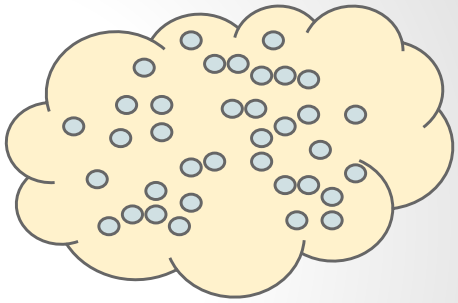
c is upregulated by a
c is downregulated by r

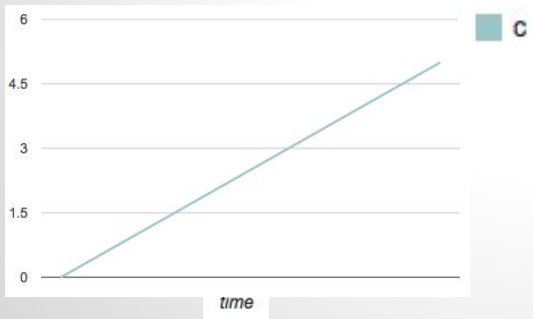
Differential equations formalize these relationships by defining protein level *rates of change*.

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



Example: synthesis of a gene





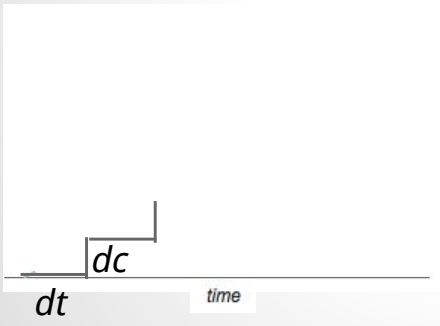
$$\frac{dc}{dt} = k_{on}$$

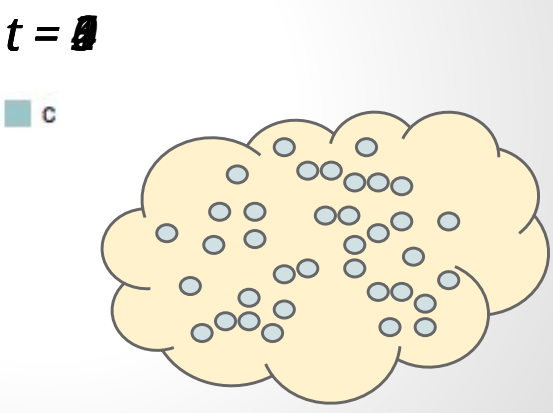
change in concentration → $\frac{dc}{dt} = k_{on}$ ← is a constant

over small time period →

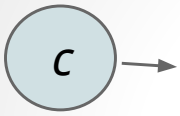
change in concentration → $\frac{dc}{dt} = k_{on}$ ← is a constant (in c/second)

over small time period →

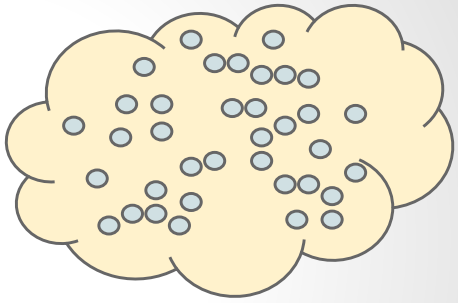




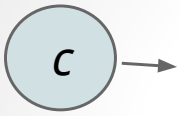
In most situations, a protein's rate of change will depend on how much is around



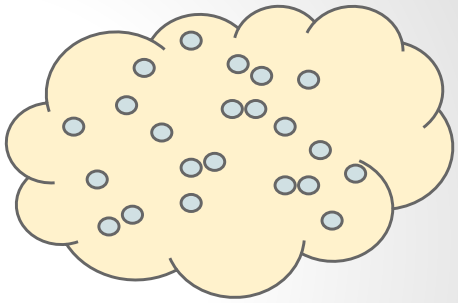
Example: degradation of a gene



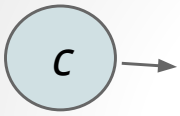
In most situations, a protein's rate of change will depend on how much is around



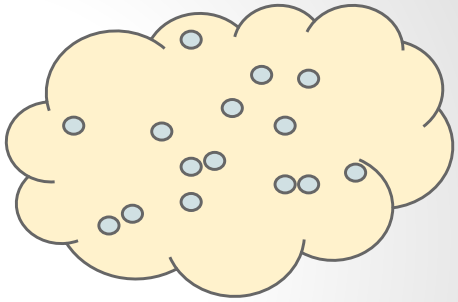
Example: degradation of a gene



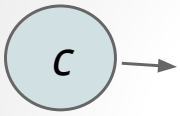
In most situations, a protein's rate of change will depend on how much is around



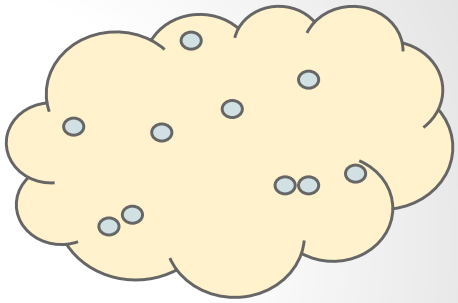
Example: degradation of a gene



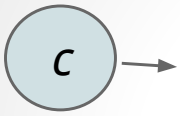
In most situations, a protein's rate of change will depend on how much is around



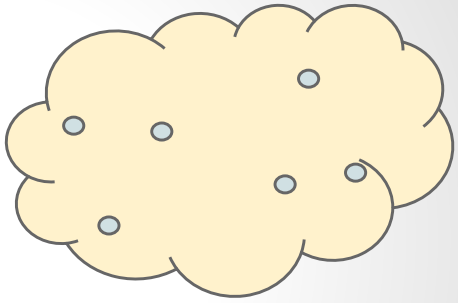
Example: degradation of a gene



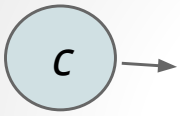
In most situations, a protein's rate of change will depend on how much is around



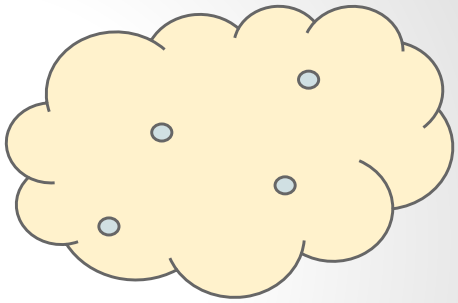
Example: degradation of a gene



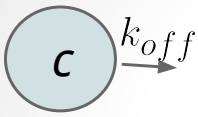
In most situations, a protein's rate of change will depend on how much is around



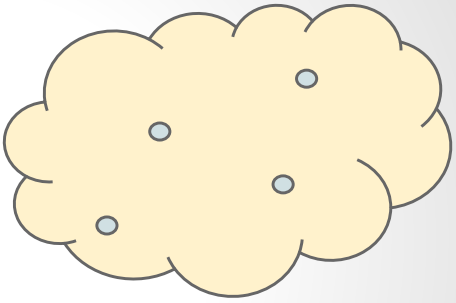
Example: degradation of a gene

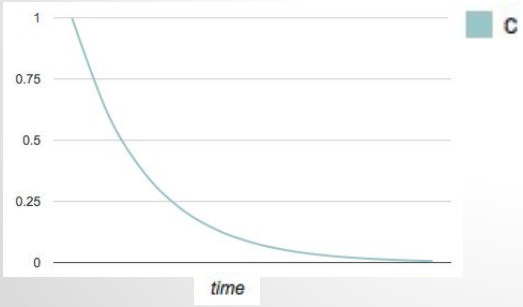


In most situations, a protein's rate of change will depend on how much is around



Example: degradation of a gene



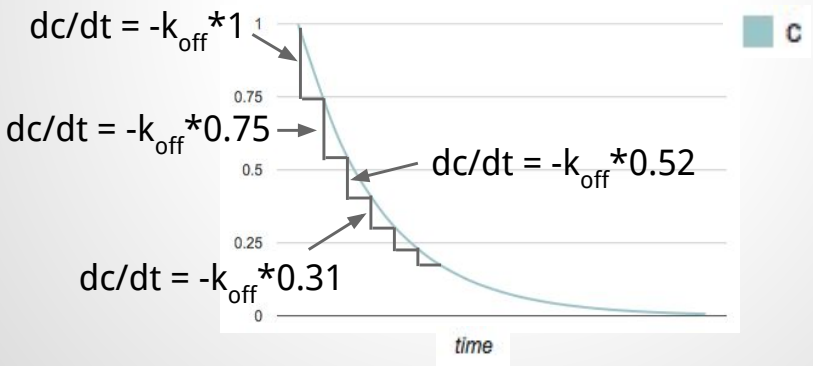


Note that rate depends on c

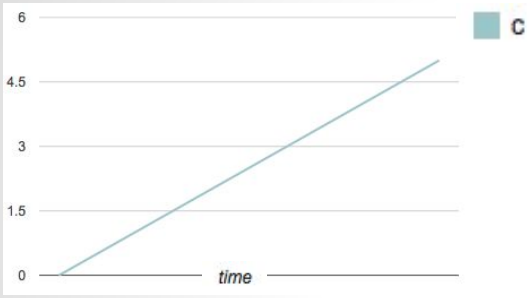
$$\frac{dc}{dt} = -k_{off}c$$

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

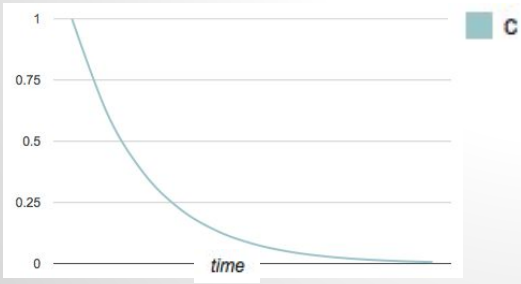
$$\frac{dc}{dt} = -k_{off}c$$



Discussion break:
What do k_on and k_off do?

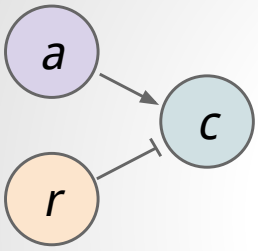


$$\frac{dc}{dt} = k_{on}$$

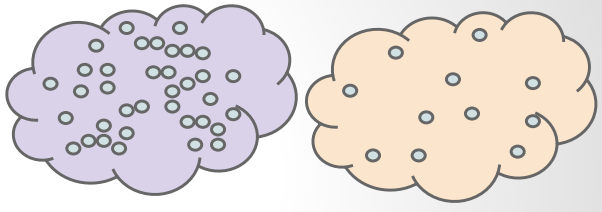


$$\frac{dc}{dt} = -k_{off}c$$

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

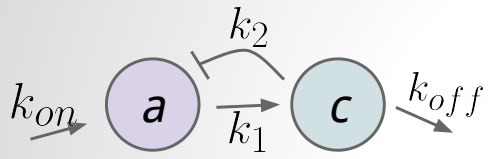


Example: gene regulation



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

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

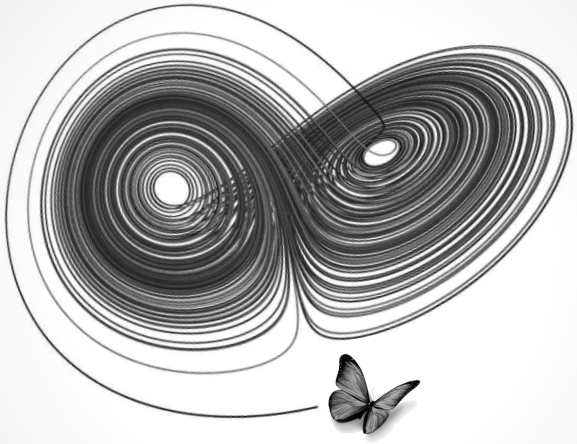
Why is it called differential? One could argue that differential equations could actually be called derivative equations. That?s because the equation itself is really just a way of explicitly writing out the derivative of a variable. For those that don?t remember a derivative is just ?a way to represent 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:

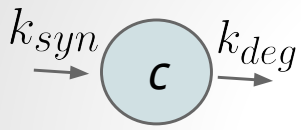
What to do when equations get tricky

Graphically visualizing rates of change

Visualizing the meaning of equilibrium

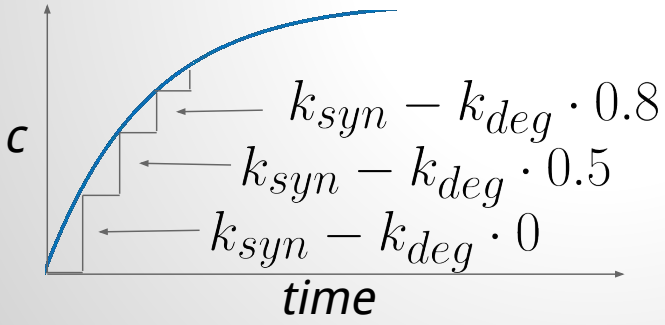
More complicated examples

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



synthesis and degradation of gene

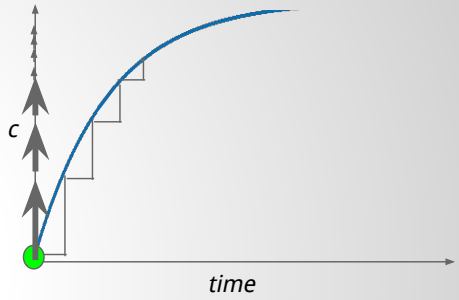
$$\frac{dc}{dt} = k_{syn} - k_{deg}c$$

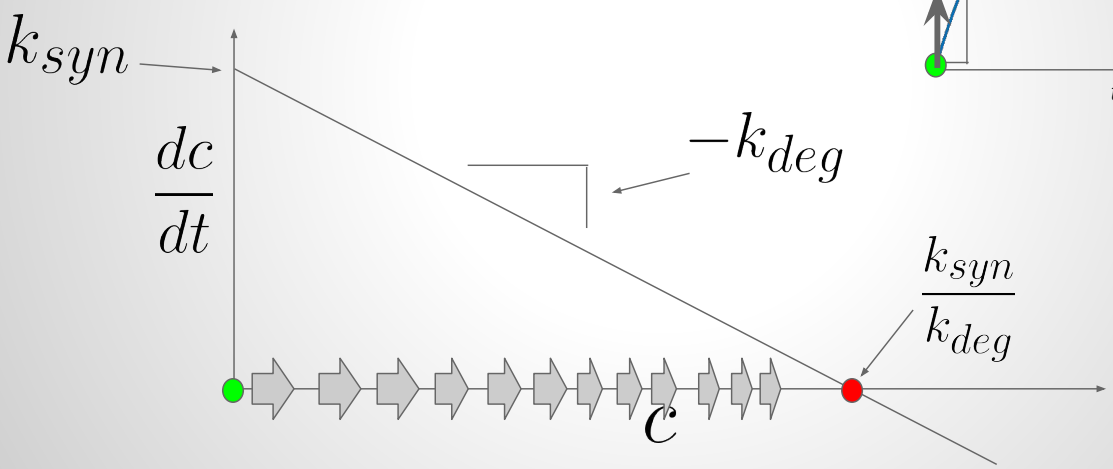


Does it ever stop? When?

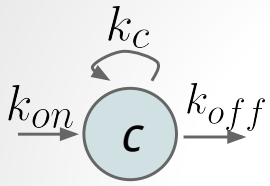
We graph dc/dt to see the protocol for how *c* changes.

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



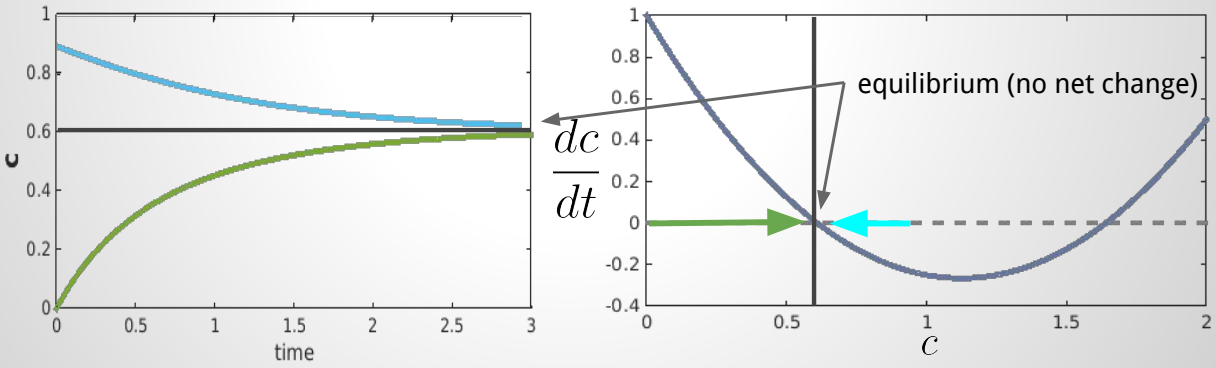


This same graphical presentation can be used to analyze more and more complicated systems

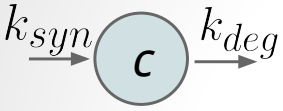


Example: gene auto-regulation

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

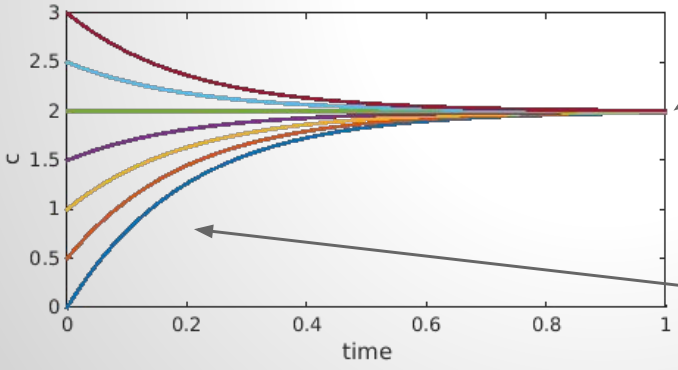


We can find equilibrium points by looking for concentrations at which the net change equals zero



Example: gene auto-regulation

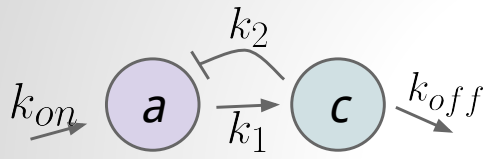
$$\frac{dc}{dt} = k_{syn} - k_{deg} \cdot c = 0$$



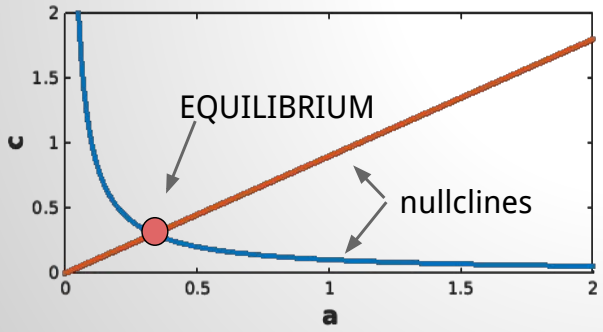
EQUILIBRIUM is important: it is where the system ends up after you wait long enough

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

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



Example: gene feedback



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

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

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

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

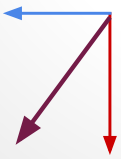
The "protocol" describes how *c* and *a* change depending on their current values

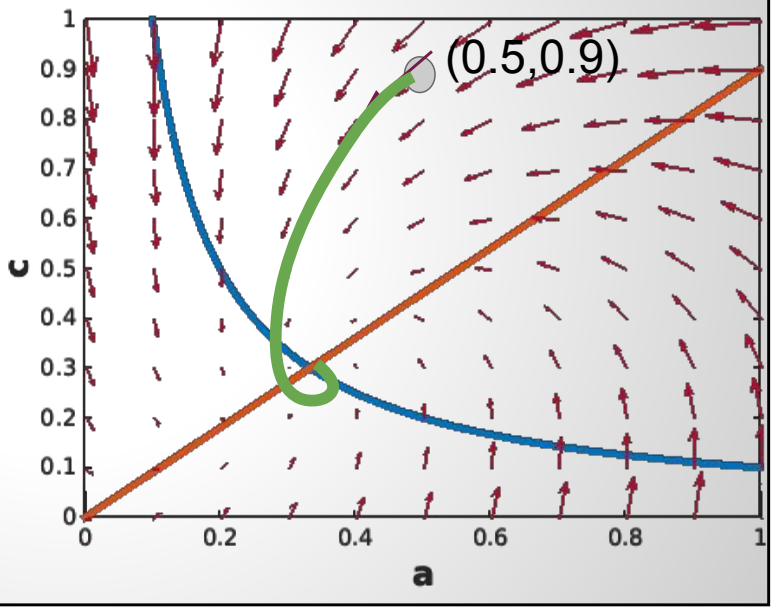
$$\frac{dc}{dt} = \overset{9}{k_1} a - \overset{10}{k_{off}} c$$

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

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

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





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

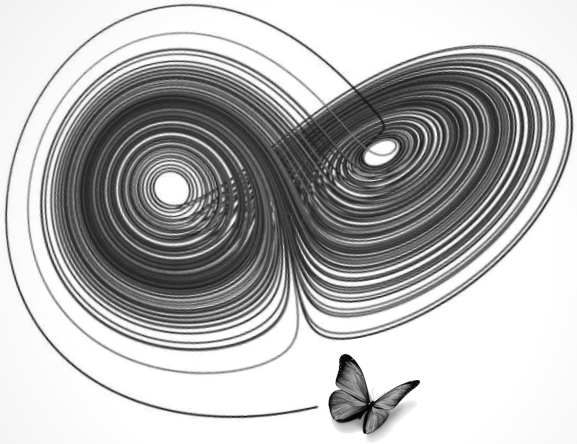
Why simplify before simulating? There are a number of reasons to simplify your equations before you ever start a simulation. Here are my top three. 1) You won?t be able to remember or adequately describe all the parameters that went into a given simulation when somebody asks you. If you have 20 parameters 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

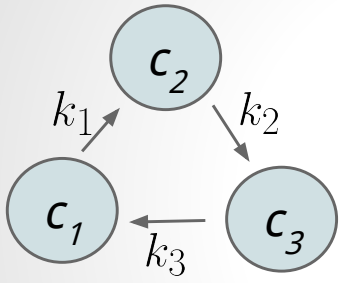
Parameters: the constants that define the system

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

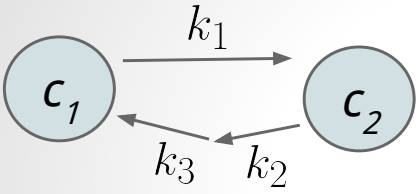
$$c_1 + c_2 + c_3 = C$$

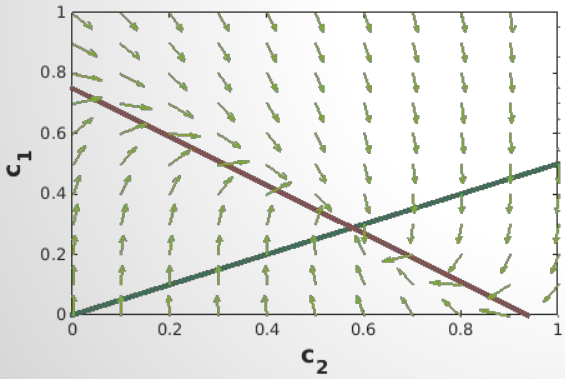
$$\frac{dc_1}{dt} = k_3 c_3 - k_1 c_1$$

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

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

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

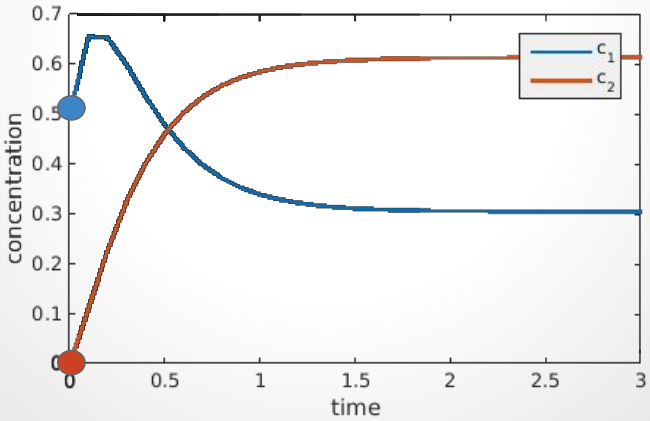
$$\frac{dc_1}{dt} = 1 * 8 - 8 * c_2 - (2 + 8) * c_1 \qquad \frac{dc_2}{dt} = 2 * c_1 - 1 * c_2$$

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

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

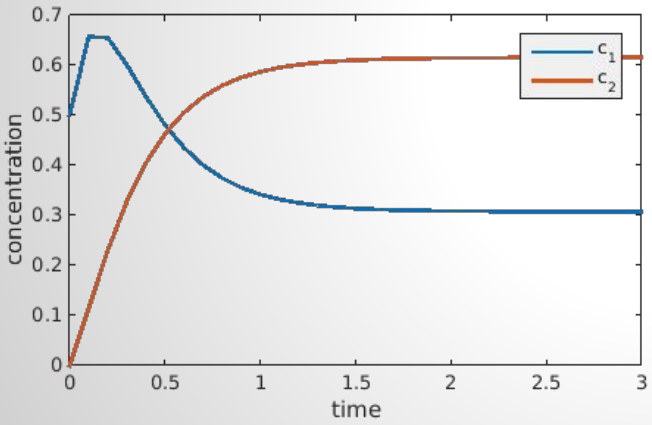
$$\frac{dc_2}{dt} = k_1 c_1 - k_2 c_2$$

```
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
```

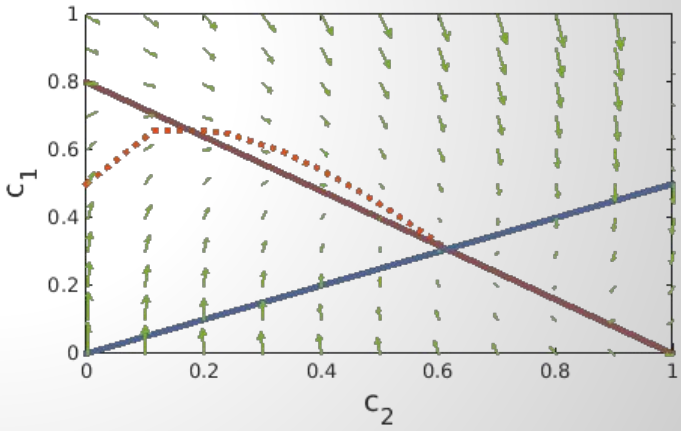
```
t = 0:0.1:10;
c0 = [1,0];

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.

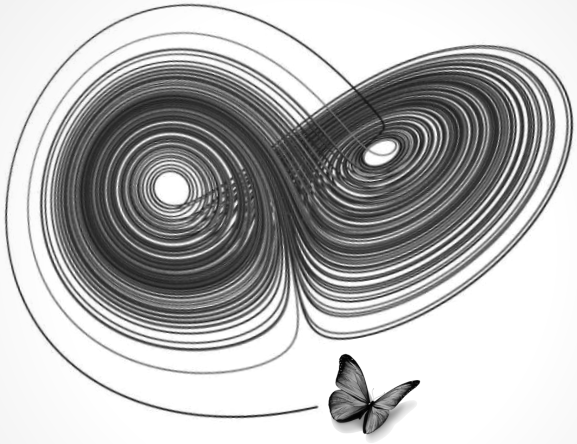
Running simulations : The following two snippets of code is all you need to solve differential equations in MATLAB.

```
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

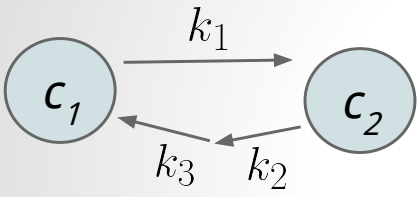


Week 5: Analyzing equations and understanding simulation output

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?



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

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

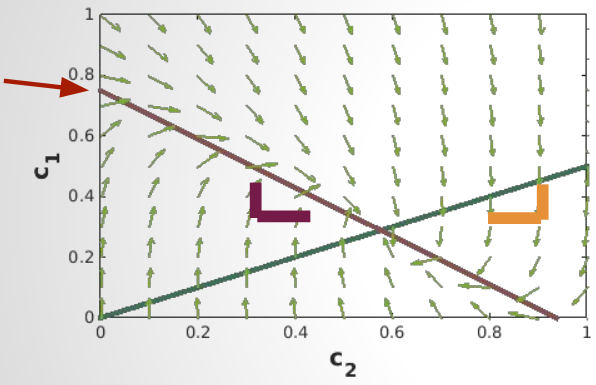
What happens if you double k_3 or k_1?

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

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

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

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

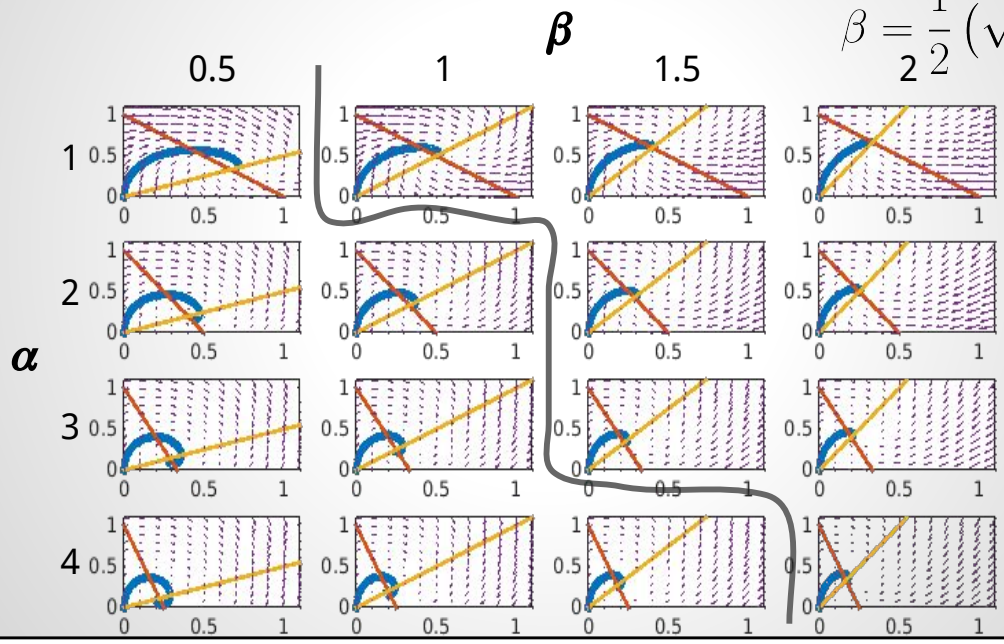
$$\frac{d\rho_1}{dt} = 1 - \alpha\rho_2 - \rho_1$$

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

```
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

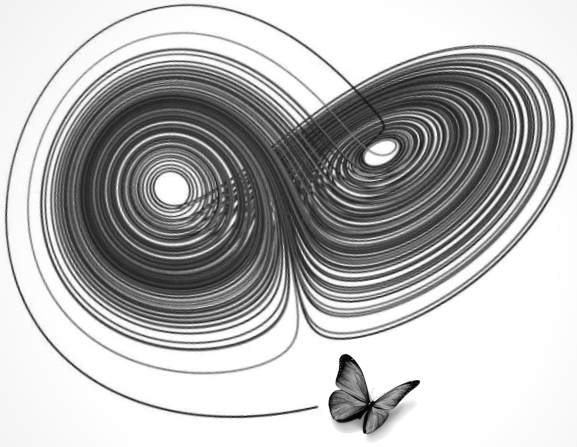
1. Continue with the system you wrote about in the last project assignment
2. Try to simplify your equations again (10 pts)
    a. Try to use tricks like the ones we came up with today
3. Modify your simulation to reflect your simplification (5 pts)
4. Systematically vary parameters to find qualitative 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

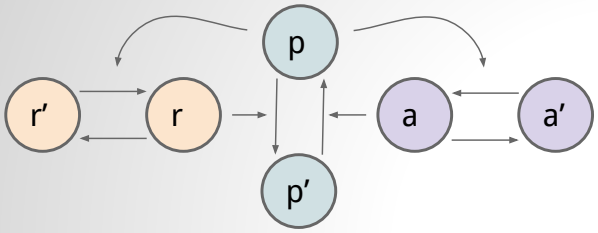
Today's aim is to get us working with systems that have 3 or more components

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

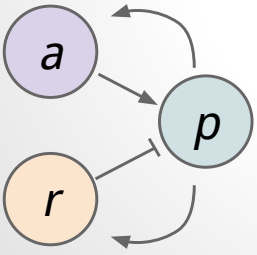
$$\frac{dr'}{dt} = k_1' r + k_1^p r p' - k_1' r' \qquad \frac{dr}{dt} = -k_1' r - k_1^p r p' + k_1' r'$$

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

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

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

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



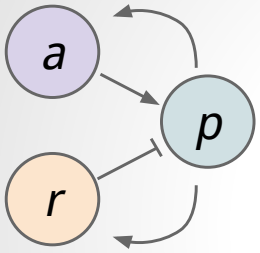
simplified diagram

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

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

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

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

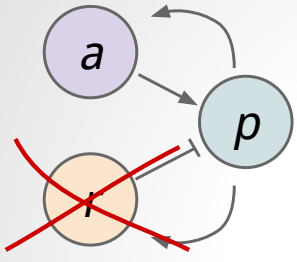
finish: 3 equations + 4 constants

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

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

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

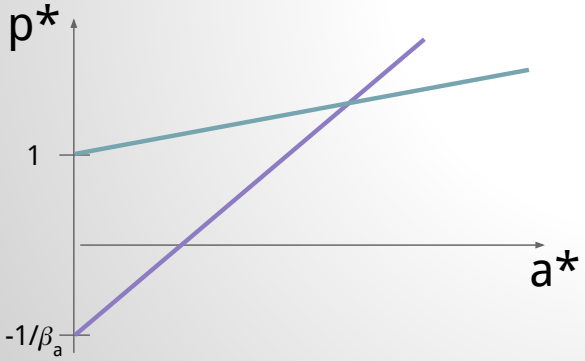
To start, let's see what happens if r = 0



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

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

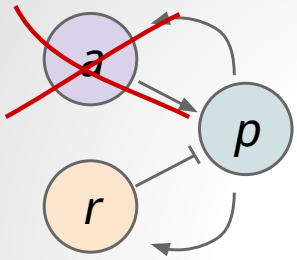
$$\cancel{\frac{dr}{dt} = 1 + \beta_r p - \frac{r}{\kappa_r}}$$



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

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

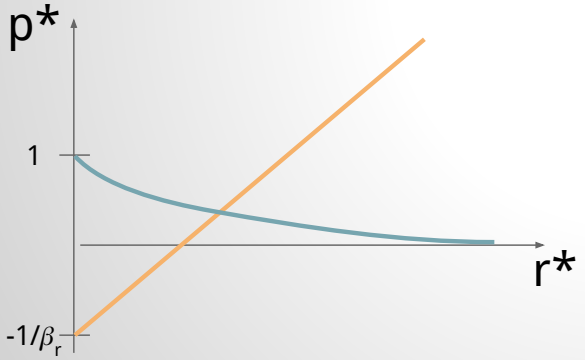
and what if a = 0



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

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

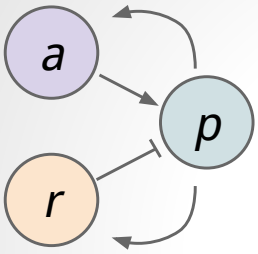
$$\frac{dr}{dt} = 1 + \beta_r p - \frac{r}{\kappa_r}$$



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

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

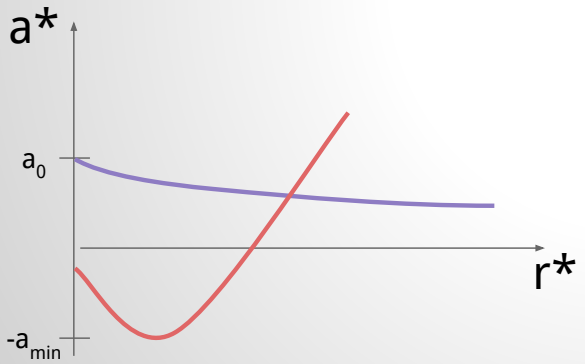
Let's just see what this system looks like if p very close to its equilibrium value.



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

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

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



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

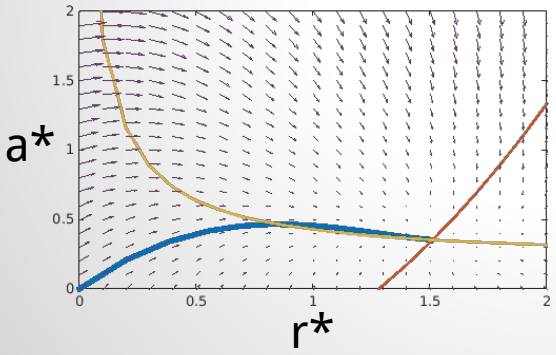
$$a^* = \omega_r (r - r_0)^2 - a_{min}$$

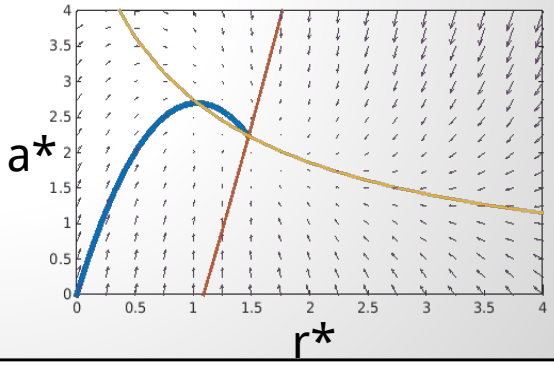
Now that we have simplified and gathered some intuition, we can start analyzing simulations

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

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

```
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.[14] 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.[14] 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.[76] 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.

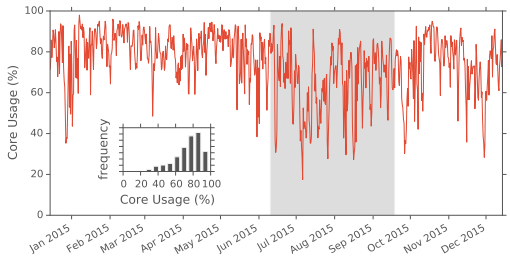


Figure C.1: Average core usage for a 244 node shared HPC partition in the Midway cluster. Insert shows usage statistics histogram.

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[24]. Most data centers have local power due to a backup generator, which could also be used to absorb some load during peak time [45]. More recently, methods of energy storage have been proposed[57] 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[60], 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[80] 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[5] 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.

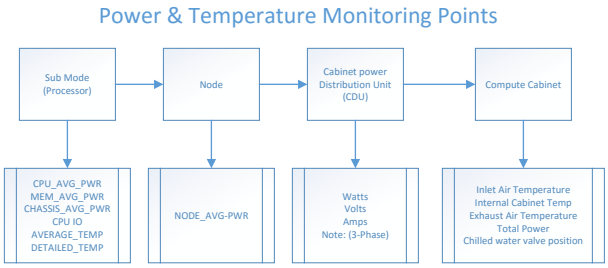


Figure C.2: Energy monitoring framework.

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

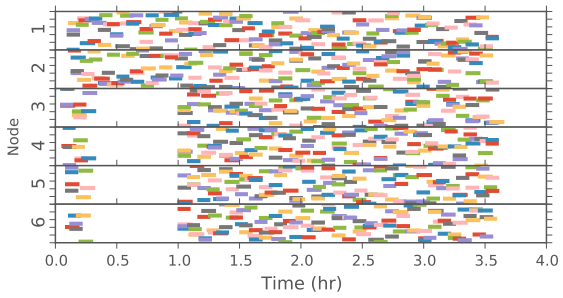


Figure C.3: Diagram of job scheduling during a four node temporary shutdown experiment. Each colored rectangle displays the execution time of a single LAMPPS test job running for approximately 5 minutes.

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

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

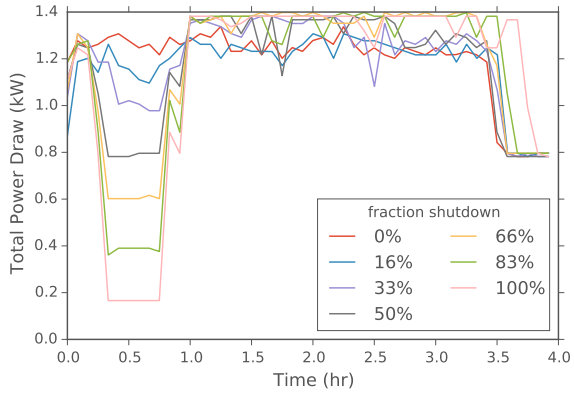


Figure C.4: Total power consumed during experiments where variable numbers of machines were shut down during simulated peak pricing.

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

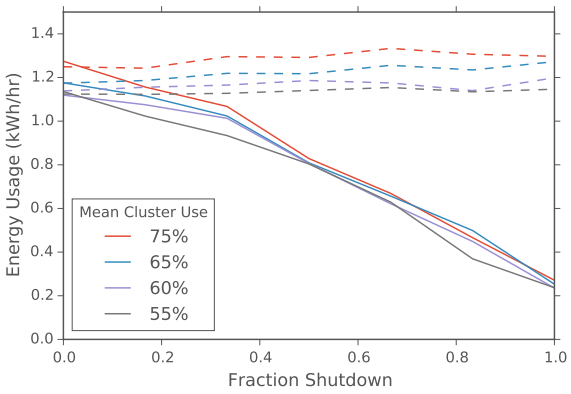


Figure C.5: Energy usage of test cluster during partial shutdown experiments. Solid lines indicate power usage during the shutdown, while dashed lines indicate power usage after returning to full operation.

capacity $r_L = 1$, and n_L is minimized subject to constraints giving $n_L = S/t_s - n_H(T - t_s)/t_s$.

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.

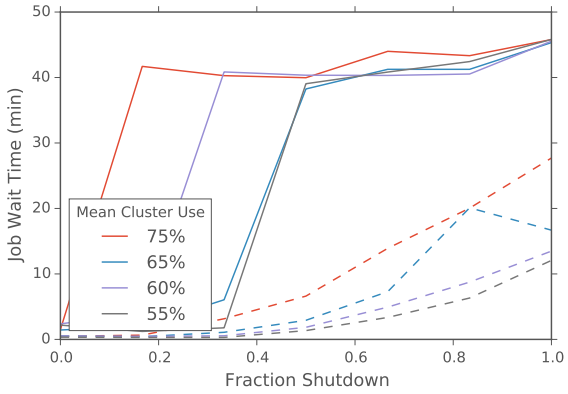


Figure C.6: Maximum (solid) and mean (dashed) job wait times during partial shutdown experiments.

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

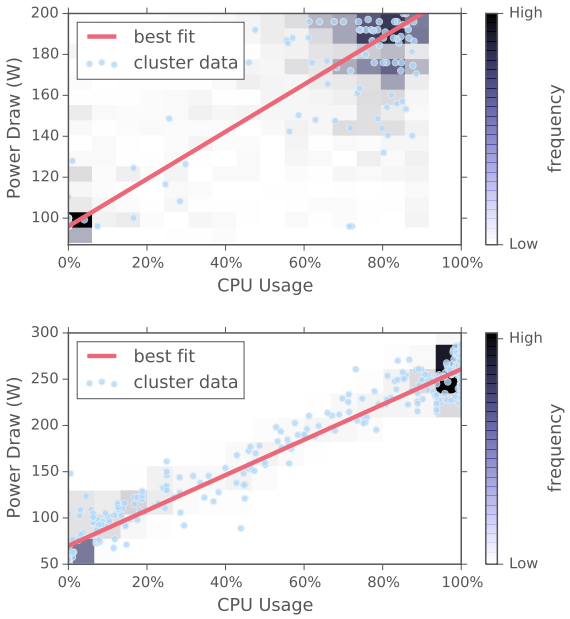


Figure C.7: Power data for test cluster (top) and production cluster (bottom) nodes in presence of variable usage. The slope and intercept of the line are used to determine u_v and u_0 respectively.

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

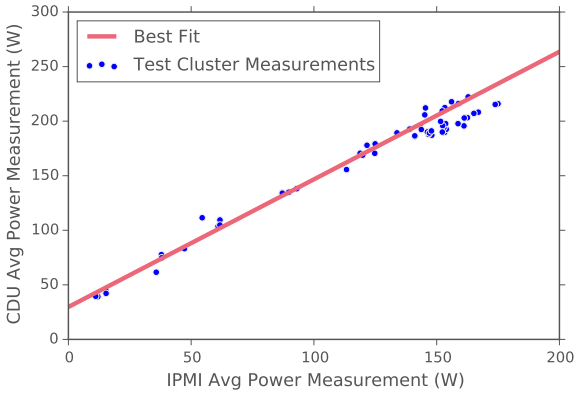


Figure C.8: Comparison between node level IPMI measurements and rack level CDU measurements. Best fit shows the model relationship used to convert IPMI data to estimated total power draw.

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

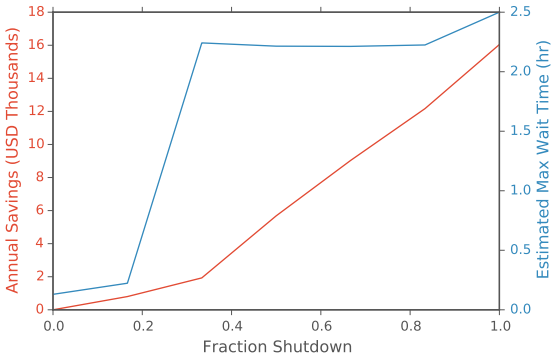


Figure C.9: Estimated savings from partial cluster shutdowns.

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.2 activnet Simulation Source

In case the online source and documentation isn't accessible for the activnet model, I'm including the simulation code here.

D.2.1 Instructions for running MATLAB pre-compiled code

D.2.2 Simulation Source

```
1 function p = activnet_gen(zet,L,mu,kap,lc,xi,ups,phi,psi,r,
    sig,Dx,Dy,Df,Dw,ls,lf,tinc,tfin,nonlin)
2 % generates an active network simulation and prints node
    positions
3 % at time steps.  Parameters are defined as follows:
4 %
5 %   zet - medium viscosity
6 %   L - length of the filament
```

```
7  %   mu - compressional modulus of the filament
8  %   kap - bending modulus of a filament if ls<L
9  %   lc - average distance between filament overlaps
10 %   xi - frictional resistance between two overlapping
       segments
11 %   ups - motor force at filament overlaps
12 %   phi - fraction of overlaps that receive a motor force
13 %   psi - spatial variation in motor force (see below)
14 %   sig - applied force
15 %   Dx - x-dimension of domain
16 %   Dy - y-dimension of domain
17 %   Df -
18 %   Dw - width of window in x-dimension where forces/
       constraints are applied
19 %   ls - length of filament segments
20 %   lf - length of force falloff at end of filament (for
       continuous forces)
21 %   tinc - time increment to return solutions
22 %   tfin - end time of simulation
23 %   nonlin - nonlinear factor by which to make filament
       stiffer by extension
24 %
25 %   See also SUM, PLUS.
26
27
28 %% this mess just ensures that any string input is converted
       to numbers
29     if(ischar(zet)); zet = str2num(zet); end;
30     if(ischar(L)); L = str2num(L); end;
31     if(ischar(mu)); mu = str2num(mu); end;
32     if(ischar(kap)); kap = str2num(kap); end;
```

```
33      if(ischar(lc)); lc = str2num(lc); end;
34      if(ischar(xi)); xi = str2num(xi); end;
35      if(ischar(ups)); ups = str2num(ups); end;
36      if(ischar(phi)); phi = str2num(phi); end;
37      if(ischar(psi)); psi = str2num(psi); end;
38      if(ischar(r)); r = str2num(r); end ;
39      if(ischar(sig)); sig = str2num(sig); end;
40      if(ischar(Dx)); Dx = str2num(Dx); end;
41      if(ischar(Dy)); Dy = str2num(Dy); end;
42      if(ischar(Df)); Df = str2num(Df); end;
43      if(ischar(Dw)); Dw = str2num(Dw); end;
44      if(ischar(ls)); ls = str2num(ls); end;
45      if(ischar(lf)); lf = str2num(lf); end;
46      if(ischar(tinc)); tinc = str2num(tinc); end;
47      if(ischar(tfin)); tfin = str2num(tfin); end;
48      if(ischar(nonlin)); nonlin = str2num(nonlin); end;
49      Dp = 1;
50      if(Df<0);Df=abs(Df);Dp = Df; end;
51  %     rng(abs(nonlin));
52
53      %% use inputs to calculate number of filaments to add
54      ncnt = ceil(L/ls)+1;
55      N = floor(2*Dx*Dy/lc/L);
56
57      nu=[];
58      if(ups>0)
59          nu = ups*double(rand(N,N)<phi).*(ones(N,N)-eye(N,N));
60          nu = (nu+nu')/2;
61      end
62
63      %% initialize network
```

```
64
65      p = zeros(N*ncnt,2);
66      for i=1:N
67          p((i-1)*ncnt+1,:) = [Dp*Dx*rand Dy*rand];
68          thet = rand*2*pi;
69          for j = 2:ncnt
70              p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(ncnt
                    -1.0)*[cos(thet) sin(thet)];
71          end
72      end
73      if(nonlin<0)
74          p = zeros(N*ncnt,2);
75          for i=1:N
76              p((i-1)*ncnt+1,:) = [Dx*(0.2+0.6*rand) Dy
                    *(0.2+0.6*rand)];
77              thet = rand*2*pi;
78              for j = 2:ncnt
79                  p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(
                        ncnt-1.0)*[cos(thet) sin(thet)];
80              end
81          end
82          nonlin=-nonlin;
83      end
84
85      muN = nonlin;
86
87      p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
88
89      fileID = 1;
90      %% solve ode
91      z0 = reshape(p,1,[]);
```

```
92
93      if(tinc>0.05/2/r)
94          tinc = 0.05/2/r;
95      end
96      tt = 0:tinc:tfin;
97
98      fprintf(fileID,'%.3f',0);
99      for i=1:length(z0)
100         fprintf(fileID,' %.4f',z0(i));
101     end
102     fprintf(fileID,'\n');
103
104     activnet(N,tt,z0,zet,L,mu,muN,kap,xi,nu,psi,sig,Dx,Dy,Df,
            Dw,Dp,ncnt,lf,r,tinc,fileID);
105
106
107 end
```

```
1
2 %% this mess just ensures that any string input is converted
      to numbers
3 zet = 0.005;
4 L = 4;
5 mu = 0.1;
6 kap = 0;
7 xi = 1;
8 ups = 1;
9 phi = 1;
10 psi = 1;
11 r = 0;
12 sig = 0;
```

```
Dx = 10;
Dy = 10;
Df = 0.5;
Dw = 0.1;
ls = 4;
lf = 0.05;
tinc = 0.01;
tfin = 10;
seed = 100;

rng(seed);

%% use inputs to calculate number of filaments to add
ncnt = ceil(L/ls)+1;
N = 2;

nu=[];
if(ups>0)
    nu = ups*double(rand(N,N)<phi).*(ones(N,N)-eye(N,N));
    nu = (nu+nu')/2;
end

%% initialize network

p = zeros(N*ncnt,2);
i=1;
p((i-1)*ncnt+1,:) = [2.5  3];
thet = 0;
for j = 2:ncnt
    p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(ncnt-1.0)*[cos
        (thet) sin(thet)];
```

```
43 end
44 i=2;
45 p((i-1)*ncnt+1,:) = [3  2.5];
46 thet = pi/2;
47 for j = 2:ncnt
48     p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(ncnt-1.0)*[cos
       (thet) sin(thet)];
49 end
50
51 p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
52
53
54 fileID = 1;
55 %% solve ode
56 z0 = reshape(p,1,[]);
57
58 if(tinc>0.05/2/r)
59     tinc = 0.05/2/r;
60 end
61 tt = 0:tinc:tfin;
62
63 fprintf(fileID,'%.3f',0);
64 for i=1:length(z0)
65     fprintf(fileID,' %.4f',z0(i));
66 end
67 fprintf(fileID,'\n');
68
69 activnet(N,tt,z0,zet,L,mu,kap,xi,nu,psi,sig,Dx,Dy,Df,Dw,ncnt,
   lf,r,tinc,fileID);
```

```
1 function g = lineSegmentGrid(indL,XY,Dx,Dy,l0)
```

```
2  % find intersections of lines XY
3  % keeps persistent ga in memory to keep from generating it
       every time
4
5  persistent ga;
6
7  if isempty(ga)
8      ga = zeros(10000000,5);
9  end
10
11 dg=Dy/floor(Dy/2/10);
12 XYg = ceil(XY/dg);
13 % n_max = 100;
14 % grid = zeros(2*D/dg,D/dg,n_max);
15
16 xmin = min(XYg(:,1),XYg(:,3));
17 xmax = max(XYg(:,1),XYg(:,3));
18 ymin = min(XYg(:,2),XYg(:,4));
19 ymax = max(XYg(:,2),XYg(:,4));
20
21 str = 1;
22 for xn=1:Dx/dg
23     for yn=1:Dy/dg
24         subXY = xmin<=xn&xmax>=xn&ymin<=yn&ymax>=yn;
25 %         subXY = grid(xn,yn);
26         gsub = lineSegmentIntersect(indL(subXY),XY(subXY,:));
27         if(~isempty(gsub))
28             lst = str+size(gsub,1)-1;
29             ga(str:lst,:) = gsub;
30             str = lst+1;
31         end
```

```
    end
end

g = unique(ga(1:str-1,:),'rows');
end
```

```
function g = lineSegmentIntersect(indL,XY,D,l0)
%LINESEGMENTINTERSECT Intersections of line segments.
%   OUT = LINESEGMENTINTERSECT(XY1,XY2) finds the 2D Cartesian Coordinates of
%   intersection points between the set of line segments given in XY1 and XY2.
%
%   XY1 and XY2 are N1x4 and N2x4 matrices. Rows correspond to line segments.
%   Each row is of the form [x1 y1 x2 y2] where (x1,y1) is the start point and
%   (x2,y2) is the end point of a line segment:
%
%                     Line Segment
%         o-------------------------------------o
%         ^                                     ^
%     (x1,y1)                               (x2,y2)
%
%   OUT is a structure with fields:
%
%   'intAdjacencyMatrix' : N1xN2 indicator matrix where the entry (i,j) is 1 if
%         line segments XY1(i,:) and XY2(j,:) intersect.
%
%   'intMatrixX' : N1xN2 matrix where the entry (i,j) is the
```

```
    X coordinate of the
21  %       intersection point between line segments XY1(i,:) and
     XY2(j,:).
22  %
23  %   'intMatrixY' : N1xN2 matrix where the entry (i,j) is the
    Y coordinate of the
24  %       intersection point between line segments XY1(i,:) and
     XY2(j,:).
25  %
26  %   'intNormalizedDistance1To2' : N1xN2 matrix where the (i,j
    ) entry is the
27  %       normalized distance from the start point of line
    segment XY1(i,:) to the
28  %       intersection point with XY2(j,:).
29  %
30  %   'intNormalizedDistance2To1' : N1xN2 matrix where the (i,j
    ) entry is the
31  %       normalized distance from the start point of line
    segment XY1(j,:) to the
32  %       intersection point with XY2(i,:).
33  %
34  %   'parAdjacencyMatrix' : N1xN2 indicator matrix where the (
    i,j) entry is 1 if
35  %       line segments XY1(i,:) and XY2(j,:) are parallel.
36  %
37  %   'coincAdjacencyMatrix' : N1xN2 indicator matrix where the
     (i,j) entry is 1
38  %       if line segments XY1(i,:) and XY2(j,:) are coincident
    .
39
40  % Version: 1.00, April 03, 2010
```

```
41 % Version: 1.10, April 10, 2010
42 % Author:  U. Murat Erdem
43
44 % CHANGELOG:
45 %
46 % Ver. 1.00:
47 %   -Initial release.
48 %
49 % Ver. 1.10:
50 %   - Changed the input parameters. Now the function accepts
       two sets of line
51 %   segments. The intersection analysis is done between these
       sets and not in
52 %   the same set.
53 %   - Changed and added fields of the output. Now the
       analysis provides more
54 %   information about the intersections and line segments.
55 %   - Performance tweaks.
56
57 % I opted not to call this 'curve intersect' because it would
       be misleading
58 % unless you accept that curves are pairwise linear
       constructs.
59 % I tried to put emphasis on speed by vectorizing the code as
       much as possible.
60 % There should still be enough room to optimize the code but
       I left those out
61 % for the sake of clarity.
62 % The math behind is given in:
63 %   http://local.wasp.uwa.edu.au/~pbourke/geometry/lineline2d
       /
```

```
64 % If you really are interested in squeezing as much horse
       power as possible out
65 % of this code I would advise to remove the argument checks
       and tweak the
66 % creation of the OUT a little bit.
67
68 [n_rows,~] = size(XY);
69
70
71 %%% Prepare matrices for vectorized computation of line
       intersection points.
72 %
   ------------------------------------------------------------
73 X1 = repmat(XY(:,1),1,n_rows);
74 X2 = repmat(XY(:,3),1,n_rows);
75 Y1 = repmat(XY(:,2),1,n_rows);
76 Y2 = repmat(XY(:,4),1,n_rows);
77
78 X4_X3 = (X2'-X1');
79 Y1_Y3 = (Y1-Y1');
80 Y4_Y3 = (Y2'-Y1');
81 X1_X3 = (X1-X1');
82 X2_X1 = (X2-X1);
83 Y2_Y1 = (Y2-Y1);
84
85
86 numerator_a = X4_X3 .* Y1_Y3 - Y4_Y3 .* X1_X3;
87 numerator_b = X2_X1 .* Y1_Y3 - Y2_Y1 .* X1_X3;
88 denominator = Y4_Y3 .* X2_X1 - X4_X3 .* Y2_Y1;
89
```

```
90  u_a = numerator_a ./ denominator;
91  u_b = numerator_b ./ denominator;
92
93  % Find the adjacency matrix A of intersecting lines.
94  INT_B = (u_a >= 0.0000000001) & (u_a <= 0.9999999999) & (u_b
        >= 0.0000000001) & (u_b <= 0.9999999999);
95  INT_X = X1+X2_X1.*u_a;
96
97  % Arrange output.
98  [ns, ms] = find(INT_B);
99  if(~isempty(ns))
100     g = [u_a(INT_B) u_b(INT_B) indL(ns)' indL(ms)' INT_X(
            INT_B)];
101 else
102     g = [];
103 end
104 end
```

```
1  function v = mydiff(p1,p2,Dx,Dy)
2  % specialized vector differences in the periodic domain
3      v = [mysub(p1(:,1),p2(:,1),Dx) mysub(p1(:,2),p2(:,2),Dy)
           ];
4  end
```

```
1  function d = mysub(x1,x2,S)
2  % specialized subtraction in the periodic domain
3      d = x2-x1;
4      d(d>S/2) = d(d>S/2) - S;
5      d(d<-S/2) = d(d<-S/2) + S;
6  end
```

```
1  function activnet(N,tt,z0,zet,L,mu,muN,kap,xi,nu,psi,sig,Dx,
       Dy,Df,Dw,Dp,ncnt,lf,r,tinc,fileID)
2  % evalutates a simulation under the input conditions
3
4      pull = (isempty(nu)&&sig~=0);
5      if(pull)
6          options = odeset('Mass',@activnet_mass_sp,'AbsTol'
               ,0.001,'RelTol',0.001);
7      else
8          options = odeset('Mass',@activnet_mass,'AbsTol'
               ,0.001,'RelTol',0.001);
9      end
10
11     ind = 2;
12     istep = length(tt)-1;
13     if(r>0)
14         istep = 2;
15     end
16     while(ind<length(tt))
17         % solve for one timestep
18         if(pull)
19             [~,z] = ode15s(@activnet_pull_ode,tt(ind-1:ind-1+
                   istep),z0,options,zet,L,mu,muN,kap,xi,nu,psi,
                   sig,Dx,Dy,Df,Dw,ncnt,lf);
20         else
21             [~,z] = ode23(@activnet_act_ode,tt(ind-1:ind-1+
                   istep),z0,options,zet,L,mu,muN,kap,xi,nu,psi,
                   sig,Dx,Dy,Df,Dw,ncnt,lf);
22         end
23
24         % output to file
```

```
25          for is=1:size(z,1)-1
26              fprintf(fileID,'%.3f',tt(ind-1+is));
27              for i=1:size(z,2)
28                  fprintf(fileID,' %.4f',z(is+1,i));
29              end
30              fprintf(fileID,'\n');
31          end
32
33          % setup for next iteration
34          z0 = z(end,:);
35          if(r>0)
36              p = reshape(z0,[],2);
37              p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
38
39              i = randi(N,floor(r*istep*tinc*N)+(rand<mod(r*
                    istep*tinc*N,1)),1);
40              p((i-1)*ncnt+1,:) = [Dp*Dx*rand(size(i)) Dy*rand(
                    size(i))];
41              thet = rand(size(i))*2*pi;
42              for j = 2:ncnt
43                  p((i-1)*ncnt+j,:) = p((i-1)*ncnt+j-1,:)+L/(
                        ncnt-1.0)*[cos(thet) sin(thet)];
44              end
45              z0 = reshape(p,1,[]);
46          end
47
48          ind = ind+istep;
49      end
50  end
```

```
1  function dz = activnet_act_ode(t,z,zet,L,mu,muN,kap,xi,nu,psi
```

```
    ,sig ,Dx,Dy,Df,Dw,ncnt,lf)
% return right side of diff equation A*x'=f for node position
    , x


    %% compute intrafilament forces
    l0 = L/(ncnt-1);
    p = reshape(z,[],2);
    p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
    dp = zeros(size(p));
    for n=1:ncnt:length(p)
        va_orth=[0 0];
        va = [0 0];
        la = 0;
        for i=0:ncnt-2
            vb = mydiff(p(n+i,:),p(n+i+1,:),Dx,Dy);
            lb = sqrt(vb*vb');
            gam = (lb-l0)/l0;
            f = mu*vb/lb*gam;
            if(mu<0)
                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;
            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;
```

```
31                  end
32                  tor = kap/l0^2*acos(max(min(va*vb'/la/lb,1)
                        ,0));
33                  dp(n+i-1,:)=dp(n+i-1,:)+tor*va_orth/la;
34                  dp(n+i,:)=dp(n+i,:)-tor*va_orth/la;
35                  dp(n+i+1,:)=dp(n+i+1,:)+tor*vb_orth/lb;
36                  dp(n+i,:)=dp(n+i,:)-tor*vb_orth/lb;
37              end
38              va = vb;
39              va_orth = vb_orth;
40              la = lb;
41          end
42      end
43
44
45      %% add active force from motors at crosslinking points
46      if(~isempty(nu))
47          indL = 1:length(p);
48          indL = indL(mod(indL,ncnt)~=0);
49
50          subpL=p(mod(1:length(p),ncnt)~=0,:);
51          subpR=p(mod(1:length(p),ncnt)~=1,:);
52
53          subpL(subpL(:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)=subpL(
                subpL(:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)+Dx;
54          subpL(subpL(:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)=subpL(
                subpL(:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)+Dy;
55          subpR(subpR(:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)=subpR(
                subpR(:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)+Dx;
56          subpR(subpR(:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)=subpR(
                subpR(:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)+Dy;
```

```
57
58          subv = subpR-subpL;
59          subv = subv./repmat(sqrt(subv(:,1).^2+subv(:,2).^2)
                ,1,2);
60          subpL = subpL - l0*lf/2*subv;
61          subpR = subpR + l0*lf/2*subv;
62
63          XY = [subpL subpR];
64
65          subXY = XY(:,1)>Dx|XY(:,2)>Dy|XY(:,3)>Dx|XY(:,4)>Dy;
66
67          extXY = XY(subXY, :);
68
69          tsub = extXY(:,1)>Dx;
70          extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(
                tsub),1);
71          tsub = extXY(:,3)>Dx;
72          extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(
                tsub),1);
73          tsub = extXY(:,2)>Dy;
74          extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(
                tsub),1);
75          tsub = extXY(:,4)>Dy;
76          extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(
                tsub),1);
77
78          XY = [XY; extXY];
79
80          indL = [indL indL(subXY)];
81
82          g = lineSegmentGrid(indL,XY,Dx,Dy,l0);
```

```
83
84          f = min(1,max(0,(g-lf/2)/(1-lf)));
85          for ind=1:size(g,1)
86              i = g(ind,3);
87              j = g(ind,4);
88
89              vm = mydiff(p(j,:),p(j+1,:),Dx,Dy);
90              lm = sqrt(vm*vm');
91
92              edg = 1;
93
94              if(g(ind,1)<lf)
95                  edg = edg*g(ind,1)/lf;
96              elseif((1-g(ind,1))<lf)
97                  edg = edg*(1-g(ind,1))/lf;
98              end
99
100             if(g(ind,2)<lf)
101                 edg = edg*g(ind,2)/lf;
102             elseif((1-g(ind,2))<lf)
103                 edg = edg*(1-g(ind,2))/lf;
104             end
105             mul = 1;
106             if(psi>0)
107                 mul = double(g(ind,5)>=psi*abs(-Dx*Df));
108             end
109             tnu = nu(ceil(i/ncnt),ceil(j/ncnt))*mul;
110             dp(i:i+1,:) = dp(i:i+1,:) + edg*tnu/lm*[vm*(1-f(
                    ind,1));vm*f(ind,1)];
111             dp(j:j+1,:) = dp(j:j+1,:) - edg*tnu/lm*[vm*(1-f(
                    ind,2));vm*f(ind,2)];
```

```
112
113         end
114     end
115
116
117     %% and bring it all home
118
119     dz = reshape(dp,[],1);
120
121
122 end
```

```
1  function Mz = activnet_mass(t,z,zet,L,mu,muN,kap,xi,nu,psi,
       sig,Dx,Dy,Df,Dw,ncnt,lf,ext)
2  % return right side of diff equation A*x'=f for node position
       , x
3  % uses a non sparse matrix
4
5      %% create velocity coupling matrix
6      l0 = L/(ncnt-1);
7
8      p = reshape(z,[],2);
9      p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
10
11     Mo = zeros(length(p));
12
13
14     indL = 1:length(p);
15     indL=indL(mod(indL,ncnt)~=0);
16
17     subpL=p(mod(1:length(p),ncnt)~=0,:);
```

```
18      subpR=p(mod(1:length(p),ncnt)~=1,:);
19
20      subpL(subpL(:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)=subpL(subpL
            (:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)+Dx;
21      subpL(subpL(:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)=subpL(subpL
            (:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)+Dy;
22      subpR(subpR(:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)=subpR(subpR
            (:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)+Dx;
23      subpR(subpR(:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)=subpR(subpR
            (:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)+Dy;
24
25      % extend ends slightly for smooth force falloff
26      % this entire block is just to ensure that our diffeq is
            continuous
27      % as segments move past each other
28      subv = subpR-subpL;
29      subv = subv./repmat(sqrt(subv(:,1).^2+subv(:,2).^2),1,2);
30      subpL = subpL - l0*lf/2*subv;
31      subpR = subpR + l0*lf/2*subv;
32
33      XY = [subpL subpR];
34
35      subXY = XY(:,1)>Dx|XY(:,2)>Dy|XY(:,3)>Dx|XY(:,4)>Dy;
36
37      extXY = XY(subXY, :);
38
39      tsub = extXY(:,1)>Dx;
40      extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(tsub)
            ,1);
41      tsub = extXY(:,3)>Dx;
42      extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(tsub)
```

```
    ,1);
43  tsub = extXY(:,2)>Dy;
44  extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(tsub)
    ,1);
45  tsub = extXY(:,4)>Dy;
46  extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(tsub)
    ,1);
47
48  XY = [XY; extXY];
49
50  indL = [indL indL(subXY)];
51
52  g = lineSegmentGrid(indL,XY,Dx,Dy,10);
53
54  f = min(1,max(0,(g-lf/2)/(1-lf)));
55  for ind=1:size(g,1)
56      i = g(ind,3);
57      j = g(ind,4);
58
59      edg = 1;
60
61      if(g(ind,1)<lf)
62          edg = edg*g(ind,1)/lf;
63      elseif((1-g(ind,1))<lf)
64          edg = edg*(1-g(ind,1))/lf;
65      end
66
67      if(g(ind,2)<lf)
68          edg = edg*g(ind,2)/lf;
69      elseif((1-g(ind,2))<lf)
70          edg = edg*(1-g(ind,2))/lf;
```

```
71          end
72
73          Mo(i:i+1,i:i+1) = Mo(i:i+1,i:i+1) + edg*xi*[[1-f(ind
                ,1) f(ind,1)]*(1-f(ind,1)); [1-f(ind,1) f(ind,1)]*
                f(ind,1)];
74          Mo(i:i+1,j:j+1) = Mo(i:i+1,j:j+1) - edg*xi*[[1-f(ind
                ,2) f(ind,2)]*(1-f(ind,1)); [1-f(ind,2) f(ind,2)]*
                f(ind,1)];
75
76      end
77
78
79      %% generate medium viscosity with assymetry for filament
            orientation
80
81      vt = mydiff(p(2:end,:),p(1:end-1,:),Dx,Dy);
82      v = [0 0; (vt(1:end-1,:)+vt(2:end,:))/2; 0 0];
83      v(1:ncnt:end,:)=vt(1:ncnt:end,:);
84      v(ncnt:ncnt:end,:)=vt(ncnt-1:ncnt:end,:);
85      den = sqrt(v(:,1).^2+v(:,2).^2);
86      gx = L*zet/ncnt*(1+abs(v(:,2))./den);
87      gy = L*zet/ncnt*(1+abs(v(:,1))./den);
88
89      %% and bring it all home
90
91      Mz = (blkdiag(Mo+diag(gx),Mo+diag(gy)));
92
93  end
```

```
1  function Mz = activnet_mass_sp(t,z,zet,L,mu,muN,kap,xi,nu,psi
       ,sig,Dx,Dy,Df,Dw,ncnt,lf,ext)
```

```
2  % return right side of diff equation A*x'=f for node position
       , x
3  % uses a sparse matrix
4      %% create velocity coupling matrix
5      l0 = L/(ncnt-1);
6
7      p = reshape(z,[],2);
8      p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
9
10     Mo = sparse(length(p),length(p));
11
12
13     indL = 1:length(p);
14     indL=indL(mod(indL,ncnt)~=0);
15
16     subpL=p(mod(1:length(p),ncnt)~=0,:);
17     subpR=p(mod(1:length(p),ncnt)~=1,:);
18
19     subpL(subpL(:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)=subpL(subpL
           (:,1)<Dx/3&subpR(:,1)>2*Dx/3,1)+Dx;
20     subpL(subpL(:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)=subpL(subpL
           (:,2)<Dy/3&subpR(:,2)>2*Dy/3,2)+Dy;
21     subpR(subpR(:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)=subpR(subpR
           (:,1)<Dx/3&subpL(:,1)>2*Dx/3,1)+Dx;
22     subpR(subpR(:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)=subpR(subpR
           (:,2)<Dy/3&subpL(:,2)>2*Dy/3,2)+Dy;
23
24     % extend ends slightly for smooth falloff
25     subv = subpR-subpL;
26     subv = subv./repmat(sqrt(subv(:,1).^2+subv(:,2).^2),1,2);
27     subpL = subpL - l0*lf/2*subv;
```

```
28      subpR = subpR + l0*lf/2*subv;
29
30
31      XY = [subpL subpR];
32
33      subXY = XY(:,1)>Dx|XY(:,2)>Dy|XY(:,3)>Dx|XY(:,4)>Dy;
34
35      extXY = XY(subXY, :);
36
37      tsub = extXY(:,1)>Dx;
38      extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(tsub)
            ,1);
39      tsub = extXY(:,3)>Dx;
40      extXY(tsub,:)=extXY(tsub,:)-repmat([Dx 0 Dx 0],sum(tsub)
            ,1);
41      tsub = extXY(:,2)>Dy;
42      extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(tsub)
            ,1);
43      tsub = extXY(:,4)>Dy;
44      extXY(tsub,:)=extXY(tsub,:)-repmat([0 Dy 0 Dy],sum(tsub)
            ,1);
45
46      XY = [XY; extXY];
47
48      indL = [indL indL(subXY)];
49
50      g = lineSegmentGrid(indL,XY,Dx,Dy,l0);
51
52      f = min(1,max(0,(g-lf/2)/(1-lf)));
53      for ind=1:size(g,1)
54          i = g(ind,3);
```

```
55          j = g(ind,4);
56
57          edg = 1;
58
59          if(g(ind,1)<lf)
60              edg = edg*g(ind,1)/lf;
61          elseif((1-g(ind,1))<lf)
62              edg = edg*(1-g(ind,1))/lf;
63          end
64
65          if(g(ind,2)<lf)
66              edg = edg*g(ind,2)/lf;
67          elseif((1-g(ind,2))<lf)
68              edg = edg*(1-g(ind,2))/lf;
69          end
70
71          Mo(i:i+1,i:i+1) = Mo(i:i+1,i:i+1) + edg*xi*[[1-f(ind
                ,1)  f(ind,1)]*(1-f(ind,1)); [1-f(ind,1)  f(ind,1)]*
                f(ind,1)];
72          Mo(i:i+1,j:j+1) = Mo(i:i+1,j:j+1) - edg*xi*[[1-f(ind
                ,2)  f(ind,2)]*(1-f(ind,1)); [1-f(ind,2)  f(ind,2)]*
                f(ind,1)];
73
74      end
75
76
77      %% decouple nodes as they enter the edge zone
78      if(sig~=0)
79          sz = size(Mo,2);
80
81          subp = p(:,1)<Dw*Dx;
```

```
82          multi = repmat(4*p(subp,1)/Dw/Dx-3,1,sz);
83          Mo(subp,:)=Mo(subp,:).*multi;
84
85          subp = p(:,1)>Dx*(1-Dw);
86          multi = repmat(4*abs(p(subp,1)-Dx)/Dw/Dx-3,1,sz);
87          Mo(subp,:)=Mo(subp,:).*multi;
88
89          subp = p(:,1)<3*Dx/4*Dw|p(:,1)>Dx-3*Dx/4*Dw;
90          Mo(subp,:)=zeros(sum(subp),size(Mo,2));
91      end
92
93      %% generate medium viscosity with assymetry for filament
            orientation
94
95      vt = mydiff(p(2:end,:),p(1:end-1,:),Dx,Dy);
96      v = [0 0; (vt(1:end-1,:)+vt(2:end,:))/2; 0 0];
97      v(1:ncnt:end,:)=vt(1:ncnt:end,:);
98      v(ncnt:ncnt:end,:)=vt(ncnt-1:ncnt:end,:);
99      den = sqrt(v(:,1).^2+v(:,2).^2);
100     gx = zet*(1+abs(v(:,2))./den);
101     gy = zet*(1+abs(v(:,1))./den);
102
103     %% and bring it all home
104
105     Mz = sparse(blkdiag(Mo+diag(gx),Mo+diag(gy)));
106 end
```

```
1  function dz = activnet_pull_ode(t,z,zet,L,mu,muN,kap,xi,nu,
       psi,sig,Dx,Dy,Df,Dw,ncnt,lf)
2  % return right side of diff equation A*x'=f for node position
       , x
```

```
% leaves out inefficient O(~N^2) intersection step

    %% compute intrafilament forces
    l0 = L/(ncnt-1);
    p = reshape(z,[],2);
    p = [mod(p(:,1),Dx),mod(p(:,2),Dy)];
    dp = zeros(size(p));
    for n=1:ncnt:length(p)
        va_orth=[0 0];
        va = [0 0];
        la = 0;
        for i=0:ncnt-2
            vb = mydiff(p(n+i,:),p(n+i+1,:),Dx,Dy);
            lb = sqrt(vb*vb');
            gam = (lb-l0)/l0;
            f = mu*vb/lb*gam;
            if(mu<0)
                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;
            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));
```

```
33                  dp(n+i-1,:)=dp(n+i-1,:)+tor*va_orth/la;
34                  dp(n+i,:)=dp(n+i,:)-tor*va_orth/la;
35                  dp(n+i+1,:)=dp(n+i+1,:)+tor*vb_orth/lb;
36                  dp(n+i,:)=dp(n+i,:)-tor*vb_orth/lb;
37              end
38              va = vb;
39              va_orth = vb_orth;
40              la = lb;
41          end
42      end
43
44
45
46      %% add external force at centerline and constrain edges
47      if(psi>0)
48          val = sig*sin(psi*t);
49      elseif(psi<0)
50          val = sig*round(mod(0.5+-psi*t,1)).*(round(mod(0.55+-
                psi*t/2,1))-0.5)*2;
51      else
52          val = sig;
53      end
54
55      subp = p(:,1)>(Df-Dw)*Dx&p(:,1)<(Df+Dw)*Dx;
56      ff = 1-abs(p(subp,1)-Df*Dx)/Dw/Dx;
57      if(sig<0)
58          dp(subp,1)=dp(subp,1) - Dy*val.*ff/sum(ff);
59      else
60          dp(subp,2)=dp(subp,2) - Dy*val.*ff/sum(ff);
61      end
62
```

```
63      subp = p(:,1)<Dw*Dx;
64      dp(subp,:)=dp(subp,:).*repmat(4*p(subp,1)/Dw/Dx-3,1,2);
65
66      subp = p(:,1)>Dx*(1-Dw);
67      dp(subp,:)=dp(subp,:).*repmat(4*abs(p(subp,1)-Dx)/Dw/Dx
            -3,1,2);
68
69      subp = p(:,1)<3*Dx/4*Dw|p(:,1)>Dx-3*Dx/4*Dw;
70      dp(subp,:)=0;
71
72      %% and bring it all home
73
74      dz = reshape(dp,[],1);
75
76
77  end
```

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 and M. Cristina Marchetti. Instabilities and oscillations in isotropic active gels. *Soft Matter*, 7:463–473, 2011.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] Carl-Philipp Heisenberg and Yohanns Bellaïche. Forces in tissue morphogenesis and patterning. *Cell*, 153(5):948 – 962, 2013.

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

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

[29] 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.

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

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] Simone Köhler and Andreas R. Bausch. Contraction mechanisms in composite active actin networks. *PLoS ONE*, 7(7):e39869, 07 2012.

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

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] T. B. Liverpool, M. C. Marchetti, J.-F. Joanny, and J. Prost. Mechanical response of active gels. *EPL (Europhysics Letters)*, 85(1):18007, 2009.

[47] 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.

[48] 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.

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

[50] 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.

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

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

[53] 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.

[54] 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.

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

[56] 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.

[57] 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.

[58] F. J. Nedelec, T. Surrey, A. C. Maggs, and S. Leibler. Self-organization of microtubules and motors. *Nature*, 389(6648):305–308, 09 1997.

[59] 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.

[60] 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.

[61] 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.

[62] Guillaume Salbreux, Guillaume Charras, and Ewa Paluch. Actin cortex mechanics and cellular morphogenesis. *Trends in Cell Biology*, 22(10):536 – 545, 2012.

[63] Guillaume Salbreux, Guillaume Charras, and Ewa Paluch. Actin cortex mechanics and cellular morphogenesis. *Trends in Cell Biology*, 22(10):536 – 545, 2012.

[64] 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.

[65] 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.

[66] Dimitrije Stamenovic. Cell mechanics: Two regimes, maybe three? *Nat Mater*, 5(8):597–598, 08 2006.

[67] 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.

[68] Thomas Surrey, François Nédélec, Stanislas Leibler, and Eric Karsenti. Physical properties determining self-organization of motors and microtubules. *Science*, 292(5519):1167–1171, 2001.

[69] 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.

[70] Hervé Turlier, Basile Audoly, Jacques Prost, and Jean-François Joanny. Furrow constriction in animal cell cytokinesis. *Biophysical Journal*, 106(1):114 – 123, 2014.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] Jan Wilhelm and Erwin Frey. Elasticity of stiff polymer networks. *Phys. Rev. Lett.*, 91:108103, Sep 2003.

[78] M. Wyart, H. Liang, A. Kabla, and L. Mahadevan. Elasticity of floppy and stiff random networks. *Phys. Rev. Lett.*, 101:215501, Nov 2008.

[79] 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.

[80] 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.

[81] 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.