

1-1-2011

3-D Computational Investigation of Viscoelastic Biofilms using GPUs

Paisa Seeluangsawat

University of South Carolina - Columbia

Follow this and additional works at: <https://scholarcommons.sc.edu/etd>

 Part of the [Mathematics Commons](#)

Recommended Citation

Seeluangsawat, P.(2011). *3-D Computational Investigation of Viscoelastic Biofilms using GPUs*. (Doctoral dissertation). Retrieved from <https://scholarcommons.sc.edu/etd/4648>

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact dillarda@mailbox.sc.edu.

3-D COMPUTATIONAL INVESTIGATION OF VISCOELASTIC BIOFILMS USING GPUS

By

Paisa Seeluangsawat

Bachelor of Science
Massachusetts Institute of Technology 2002
Master of Science
University of North Texas 2006

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Mathematics

College of Arts and Sciences

University of South Carolina

2011

Accepted by:

Qi Wang, Major Professor

Peter Binev, Committee Member

Hong Wang, Committee Member

Xiaofeng Yang, Committee Member

Guiren Wang, Committee Member

Lacy Ford, Vice Provost and Dean of Graduate Studies

© Copyright by Paisa Seeluangsawat, 2011

All Rights Reserved.

DEDICATION

This dissertation is dedicated to my mother–the best mother.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my adviser Dr. Qi Wang, who has guided me through the transition from classroom to research. He is always available for discussion and regularly provides insights that turn a hard problem into a manageable one. Without his patient support, this work would be impossible.

I also would like to thank Dr. Hong Wang, Dr. Peter Binev, Dr. Xiaofeng Yang, and Dr. Guiren Wang for volunteering their time to be on my Dissertation Committee. Several of them have also taught me tools and theorems that I use in this research.

I am indebted to all my teachers, both at USC and from prior. Their collective knowledge and valuable advices play an integral role in shaping me into who I am.

I appreciate friendships from USC Math faculty, fellow graduate students, and the staff. Together, they provide a home away from home, and help keep me sane through the ups and downs of my graduate school life.

Last but not the least, I would like to thank Xiao Xiao for taking care of me when I do not have time to take care of myself.

ABSTRACT

A biofilm is a slimy colony of bacteria and the materials they secrete, collectively called “extracellular polymeric substances (EPS)”. The EPS consists mostly of bio-polymers, which cross link into a network that behave viscoelastically under deformation. We propose a single-fluid multi-component phase field model of biofilms that captures this behavior, then use numerical simulations on GPUs to investigate the biofilm’s growth and its hydrodynamics properties.

We model a biofilm immersed in a solution as a two-phase fluid, consisting of the solution, which is modeled as a viscous fluid, and the biomass, which is modeled as a viscoelastic solution with viscosity much higher than that of the solution. Each fluid has its own velocity field, but the important quantity is their combined volume-averaged velocity, which is the main physically observable quantity. The theory is developed with this average velocity in mind, while tracking the individual velocities using the excessive velocity of each fluid, which is calculated from the given mixing free energy density.

By using the phase field model, the whole domain is governed by a single set of governing equations, simplifying the numerical procedure significantly. The model accounts for Cahn-Hilliard phase mixing and nucleation, biomass growth from nutrient consumption, nutrient diffusion, fluid flow interaction, viscous stress due to bacteria and the solution, and elastic stress due to the EPS.

We use a finite difference scheme based on a staggered grid in 2-D and 3-D geometry. The incompressible Navier-Stokes equation is solved by the Gauge-Uzawa method, modified so that it can be quickly solved using Fast Fourier Transform (FFT). The elastic stress is governed by a modified Giesekus constitutive equation valid trivially in the solvent

region, which we solve by a backward interpolation and an explicit updating scheme. The remaining equations are discretized using a semi-explicit scheme and solved iteratively by the BiCG-stab method.

The numerical scheme is implemented on graphics processing units (GPUs), which offers up to a hundred fold speed up over a traditional single-thread CPU. Our numerical implementation is carried out such that only a small amount of key parameters are passed between CPU and GPU, while large data are kept in GPU at all time in order to avoid the relatively low bandwidth and high latency of the CPU-GPU data transfer. They are copied to the CPU memory only occasionally in order to output to a file. Data are laid out in the GPU memory in such a way that GPU threads can fetch them in a coalesced manner to increase the speed of data access. We use the CUFFT package for the fast Fourier transform, and the Thrust and CUSP libraries for BiCG-stab and data management.

We carry out numerical simulations in both two and three spatial dimensions. The viscoelastic results are compared with those from the viscous model at two distinct timescales relevant to biomass growth and an imposed shear flow. In the growth timescale, measured in days, which is much longer than the elastic relaxation time, both models predict nearly identical results. This is simply because the viscoelastic model behaves like a viscous model since the elastic relaxation time is so short that the elastic effect is not felt strongly at this time scale. In the shear timescale, measured in seconds, which is shorter than the elastic relaxation time, the viscoelastic model predicts biofilms that deform under shear more than those predicted by the viscous model. In 3-D, the viscoelastic model predicts that a portion of the biomass can be pulled into a nose shaped and stream along with the flow. After the external shear ceases, the viscoelastic model predicts that the biofilms partially recoil back toward their original position, while the viscous model predicts that the biofilms stop moving. Nutrient distribution and its effect to biofilm growth is investigated by the numerical solver revealing inherent hydrodynamic interaction in the material's transport. This hydrodynamic model together with the GPU based numerical codes pro-

vide a valuable predicative tool for biofilm research, in particular, for the investigation of biofilm-solution interaction under flowing conditions.

CONTENTS

DEDICATION	iii
ACKNOWLEDGMENTS	iv
ABSTRACT	v
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER 1 INTRODUCTION	1
1.1 Biofilm	1
1.2 Biofilm models	6
1.3 Viscoelastic models	11
CHAPTER 2 BIOFILM MODEL	16
2.1 Constitutive equations	17
2.2 Nondimensionalization	20
2.3 Remark about the stress constitutive equation	22
CHAPTER 3 NUMERICAL SCHEMES AND GPU IMPLEMENTATION	24
3.1 Numerical scheme	24
3.2 Momentum equation	30
3.3 Elastic stress equation	46
3.4 Biomass volume fraction equation	47
3.5 GPU implementation	51

3.6	Mesh refinement	60
3.7	Visualizing a stress field	69
CHAPTER 4 NUMERICAL SIMULATION AND DISCUSSIONS		71
4.1	Growth dynamics of biofilms	73
4.2	Biofilm dynamics in shear flows	76
CHAPTER 5 CONCLUSION		94
BIBLIOGRAPHY		96
APPENDIX A SOLVING THE HELMHOLTZ EQUATION BY DFT		103
A.1	One dimension	104
A.2	Two and three dimensions	109
A.3	Verification	110
APPENDIX B GRID REFINEMENT ANALYSIS		112
B.1	Convergence rate	112
B.2	Global Error	113
B.3	Nonconstant refinement factors	114

LIST OF TABLES

Table 1.1	Linear viscoelastic models	12
Table 3.1	Decay rate of the $n + 1 - \epsilon$ interpolation scheme	39
Table 3.2	2-D spatial refinement result for growing a biofilm with $\Delta t = 10^{-4}$. . .	63
Table 3.3	2-D temporal refinement result for growing a biofilm with $\Delta x = 1/1024$. . .	64
Table 3.4	3-D spatial refinement result for growing a biofilm with $\Delta t = 10^{-4}$. . .	65
Table 3.5	3-D temporal refinement result for growing a biofilm with $\Delta x = 1/240$. . .	66
Table 3.6	2-D spatial refinement result for shearing a biofilm with $\Delta t = 10^{-5}$. . .	67
Table 3.7	2-D temporal refinement result for growing a biofilm with $\Delta x = 1/1024$. . .	67
Table 3.8	3-D spatial refinement result for shearing a biofilm with $\Delta t = 10^{-5}$. . .	68
Table 3.9	3-D temporal refinement result for growing a biofilm with $\Delta x = 1/240$. . .	68
Table 4.1	Parameter values used in the simulations	72
Table A.1	Spatial refinement of the Navier-Stokes solver	111
Table A.2	Temporal refinement of the Navier-Stokes solver	111

LIST OF FIGURES

Figure 1.1	Biofilm developmental stages.	2
Figure 1.2	Model of biofilm EPS.	2
Figure 1.3	Cell specialization within a biofilm colony.	4
Figure 3.1	Variable locations on the 2-D staggered grid	27
Figure 3.2	Solution of Eq.(3.81). The real part of each λ is plotted as a function of c	40
Figure 3.3	The incorrect assumption that $\mathbf{v}_{y=1} = 0$ can introduce a boundary error of order 10^{-7} on each time step	45
Figure 3.4	Shear profile for 1-D flow in a 2-D domain.	45
Figure 3.5	The bulk free energy \tilde{f} and its second derivative	49
Figure 3.6	Inflection points ϕ_1, ϕ_2 of the bulk free energy $\tilde{f}(\phi_n)$. The biomass nucleates when $\phi_n \in (\phi_1, \phi_2)$	50
Figure 3.7	2-D mesh refinement of growing a biofilm. Profiles at $t = 0$ and $t = 2$	63
Figure 3.8	3-D mesh refinement of growing a biofilm. Profiles at $t = 0$ and $t = 2$	65
Figure 3.9	2-D mesh refinement of shearing a biofilm. Profiles at $t = 0$ and $t = 2$	67
Figure 3.10	3-D mesh refinement of shearing a biofilm. Profiles at $t = 0$ and $t = 2$	68
Figure 4.1	Initial biomass volume fraction ϕ_n profile.	79
Figure 4.2	Biomass volume fraction ϕ_n , nutrient concentration c , nutrient con- sumption rate g_c , and biomass production rate g_n in the simulation of a growing bud of biofilm at $t = 300$	80
Figure 4.3	Velocity \mathbf{v} , biomass flux $\phi_n \mathbf{v}_n$, and pressure p	81
Figure 4.4	Elastic stress distributions.	82

Figure 4.5	Snapshots of the simulation of a growing randomly scattered bits of biofilm	83
Figure 4.6	The left and right colonies grow faster than those in the middle.	84
Figure 4.7	Biomass flux, elastic stress, and viscous stress due to bacteria	85
Figure 4.8	Net forces in a growing colony of biofilm.	86
Figure 4.9	Growing a bud of biofilms in 3-D	87
Figure 4.10	Snapshots of growing scattered bits of biofilms in 3-D	87
Figure 4.11	Biomass volume fraction ϕ_n , average velocity \mathbf{v} , and pressure p	88
Figure 4.12	Elastic stress and viscous stress distributions	88
Figure 4.13	Forces in the momentum equation	89
Figure 4.14	Shearing the biofilm grown from a small bud	89
Figure 4.15	Shearing of a 3-D biofilm grown from a bud	90
Figure 4.16	Shearing of a 3-D biofilm grown from scattered bits	91
Figure 4.17	Shearing of a 3-D colony of biofilm with fewer buds	92
Figure 4.18	Shearing an biofilm lump with a thin neck. The biofilm colony detaches at high shear rates.	93
Figure A.1	Odd/even extensions for grids with 4 data points	105

CHAPTER 1

INTRODUCTION

1.1 BIOFILM

Biofilms are the slimy materials commonly found on moist surfaces. They are ubiquitously found on plants and in river beds, kitchen sinks, water pipes, water filters, medical implants, in body tissues, to name a few. A biofilm is a mixture of bacteria, the slimy materials they produced, which are made up of polysaccharides, proteins, and other biomaterials collectively called “extracellular polymeric substances” (EPS), and water. The EPS harbors bacteria from surrounding environments, allowing them to communicate via chemical signals and cooperate their self-defense against harsh chemicals. Many biofilms are cooperative ecosystems of several species of bacteria, as well as fungi, algae, yeasts, protozoa, and other microorganisms.

More information about biofilm can be found in the review articles [20] [11] [45]. We give a brief overview here. Figure.1.1 illustrates developmental stages of a biofilm colony. Planktonic cells first attach to a surface, also called substratum. They then start producing EPS and multiply their quantity. The biofilm colony grows as cells multiply and more EPS are produced. Once matured, the colony releases planktonic cells, which disperse on new substrate to form a new colony. Each stage in the figure is accompanied by a photomicrograph of *Pseudomonas aeruginosa*, a model organism for biofilm study [20] [74]. Biofilms can form layers with thickness that ranges from a few microns to a few centimeters.

Under limited nutrient supplies, growth of biofilm colonies typically follows four dis-

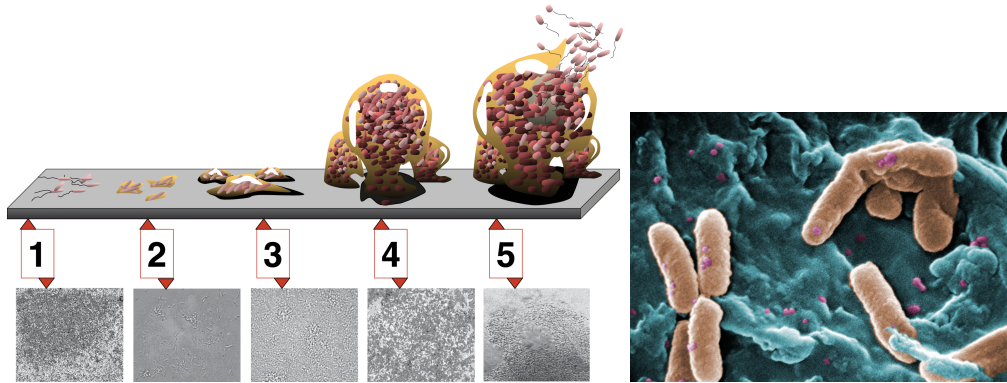


Figure 1.1: Left: Biofilm developmental stages. Each stage is illustrated by a drawing (top) and a photo of a *P. aeruginosa* biofilm (bottom). (1) Planktonic cells land. (2) Cells irreversibly attach to the site. (3) Cells start secreting EPS. (4) The biofilm grows into a mature colony. (5) Some cells disperse back into the solution. (Figure by D. Davies in [62] [40]) Right: a scanning electron micrograph (SEM) of *P. aeruginosa* (by Janice Haney Carr, Centers for Disease Control and Prevention).

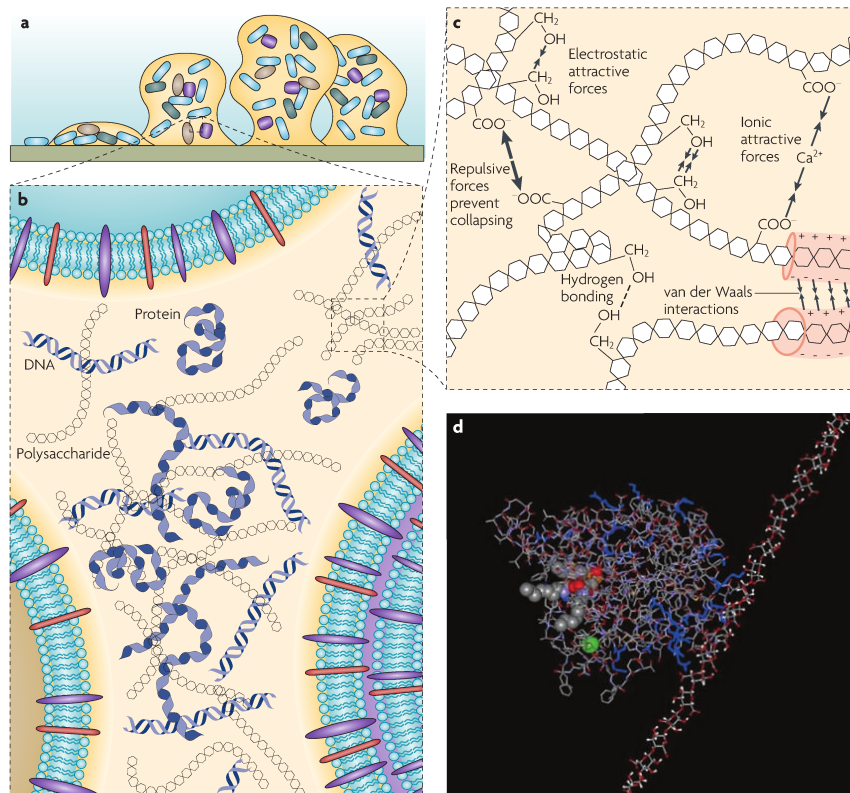


Figure 1.2: A closer look on the EPS. (a) Drawing of a biofilm colony. (b) The EPS consists of polysaccharides, proteins, and DNA as major components. (c) The EPS is stabilized by weak chemical forces and entanglement. (d) Extracellular enzyme lipase using its positively charged amino acids (blue) to weakly bind to anions in an EPS strand. (From Flemming & Wingender [20])

tinctive phases [71] [66],

1. Lag phase. Organism undergoes phenotypic change to adapt to the environment and produces necessary RNA and enzymes to get ready for cell division.
2. Exponential phase (also known as logarithmic phase). Cells multiplies (a.k.a. divide), causing exponential growth. The growth rate depends on the species and environment. For example, the doubling time of *P. aeruginosa* is about 20 minutes in mice lung [59] and 100 minutes in human lung [73]. On the other hand, *T. pallidum* in rabbit testes takes about 30 hours to double [42].
3. Stationary phase. Nutrient starts to be scarce, and waste products accumulate. The growth slows down, and is balanced out by the death rate.
4. Death phase. Nutrient diminishes.

Aside from lack of nutrient, cells can also die from drugs and harsh chemical treatments. However, drugs do not affect all cells equally. Even within the same species, some cells are more resistant to drugs than others. These are known as persistors. They neither grow nor die in the presence of a specific drug. When the drug subsides, these persistors divide again. However the rejuvenated colony does not inherit this persistence. They still show the same level of susceptibility to the same drug [4] [37].

Inside the biofilm, cells are tangled inside the EPS network, greatly reducing their mobility. This allows chemical gradient to develop, thus encourages cells inside biofilm to specialize in different functions and benefit from each other. In an example from [67] (Fig.1.3), cells on the EPS-solution interface specialize in efflux pump, which keep drug out of the EPS while allowing nutrient to pass, while younger cells divide and grow in the interior.

Several experiments study the hydrodynamics properties of biofilms [63] [61] [32] [56]. Phenomena commonly observed in flows include streaming, sloughing, detachment (shed-

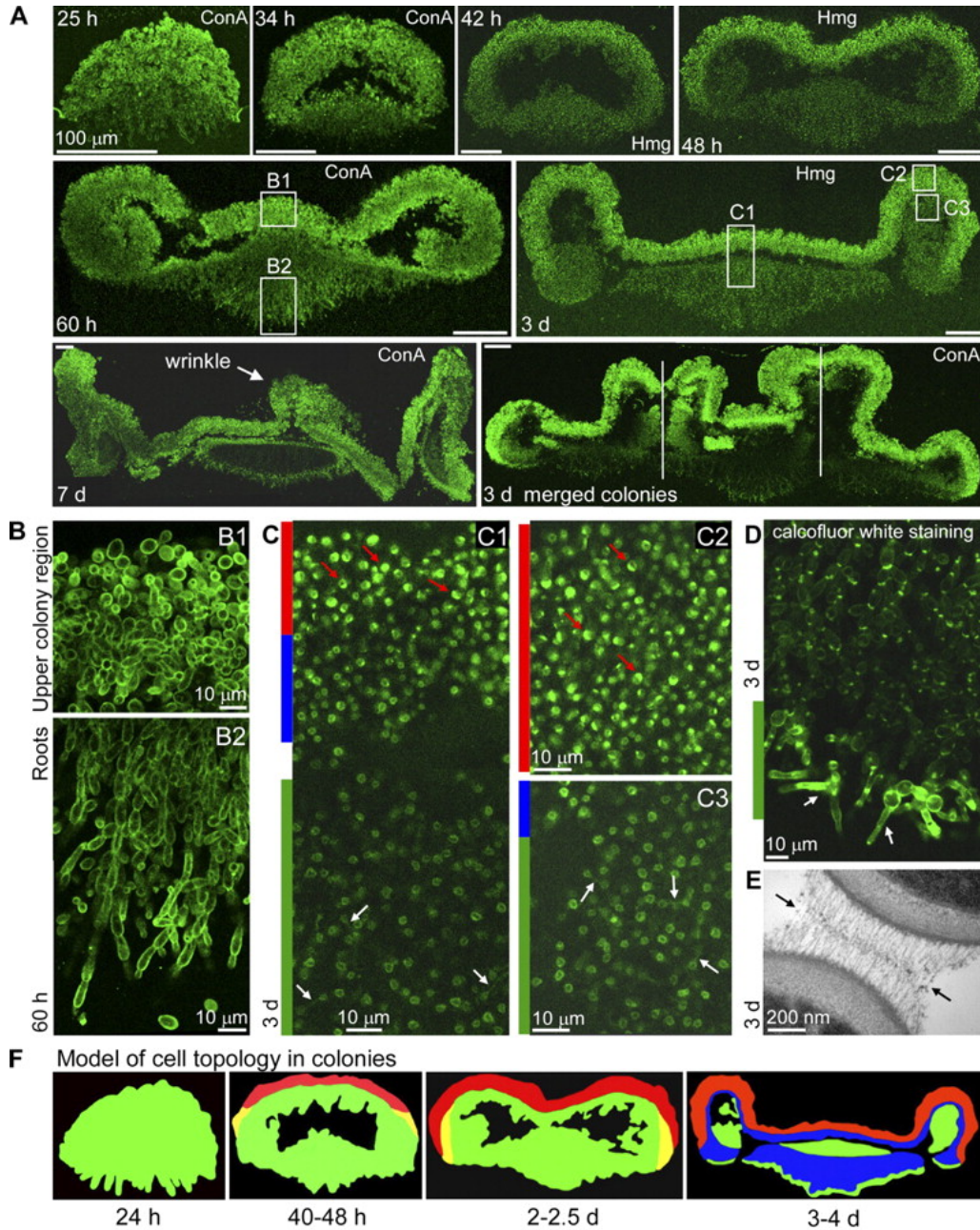


Figure 1.3: Cell specialization within a biofilm colony of *S. cerevisiae*. (A) Vertical cross sections of the colony. (B) Zoom-in on an upper region and a root region, showing different cell morphologies. The magnified regions are marked in A. (C) Spatial heterogeneity of cells in the biofilm. The color bars mark regions with stationary cells (red), young nondividing cells (blue), and dividing cells (green). Arrows mark examples of stationary cells (red) and dividing cells (white). (D) Dividing cells in root tips. Arrows mark examples of cells reaching a terminal phenotype. (E) Velcro-like interconnection between cells. (F) Topology of the colony evolving over time, showing regions with dividing (green), early stationary (yellow), stationary (red), and younger with no apparent division activity (blue) cells. (From Vachova et al [67])

ding), rolling, and viscoelastic recoil. Outside labs, biofilms have been studied in its natural habitat such as river sediments [21] and rocks under river falls [29].

Key parameters of viscoelastic materials are the elastic modulus and relaxation time. There is a wide range of the biofilm elastic shear modulus, from 10^{-2} to 10^5 Pa [57]. The value depends on the species of bacteria. Even within the same species, different papers report values that differ by 1-2 order of magnitudes. This is partly because the biofilm is a live heterogeneous material, thus its elasticity differs from specimen to specimen. Furthermore, chemical compositions of the environment can alter the strength of the EPS [34]. Additionally, some discrepancy is attributed to the different measurement methods and the difficulties in interpreting the experimental results [1]. The relaxation time of various species of biofilms are reported to be about 18 minutes [57]. However, a newer measurement technique using a microfluidic device come up with the relaxation time of *S. epidermidis* at 14 seconds [26]. This could be because a biofilm has multiple relaxation times.

Biofilms can cause many industrial problems. They corrode surfaces, clog pipes, increase fluid drag, and reduce heat transfer. They make surfaces harder to clean and disinfect, posing important risks in food processing and medical settings. Inside human body, biofilms shield pathogens from antibiotics and the host's immune system. Rather than sparsely spreading out throughout the body, bacteria cooperate and thrive together inside a biofilm. Detachment process allows a group of pathogens to migrate to a new location together [24]. On the positive side, biofilms can be harnessed to some industrial benefits in waste treatment, bio-barrier, and microbial fuel cells. A better understanding of biofilms will not only provide scientific insights into the intriguing biomaterial system but also make important economic impacts to the society.

1.2 BIOFILM MODELS

It is a challenge to model the live microorganism in biofilms and their transient growth and transport behavior. There have been many mathematical models for studying biofilms, some dating back to early 1980's. They differ in the phenomena of interest, key variables, physical effect considered, and numerical methods. Recent reviews of these models include [48], [16] [33] and [68].

The most commonly studied phenomena are growth at the expense of nutrient consumption, and biomass movement. Other phenomena of interest are cell death, quorum sensing, and phenotypic shift. These phenomena occur on different timescales,

- seconds: advection, diffusion, cell motility
- minutes: elasticity
- hours: phenotypic shift, cell division, nutrient consumption, death by drug
- days: colony growth, death by starvation

The set of phenomena we desire to capture partially dictates the list of key quantities that we need to keep track of. The most common ones are representations of biomass, nutrient, and their hydrodynamics. Other quantities include drug, signaling chemical, phenotypic ratio, and elastic stress.

Biomass

A modeler may choose to keep track of the biomass as a single material entity [52] [13] [27] [47], or separate them into two separate materials of distinct material's properties: bacteria and EPS parts [38]. More sophisticated models may keep track of several species or several phenotypes of a specie [69] [51] [46]. Models differ in how these quantity are represented. For example, the location and amount of biofilms in water have been represented by these methods,

- Location of biofilm-water interface [54] [69] [13]
- Cellular automata on a lattice [52] [28] [51]
- Individual bacteria in the continuum [35] [46]
- Scalar field of biofilm concentration [13] [17] [75] [76] [47]

A hybrid approach [50] [49] [47] is to represent the biomass as a concentration field, but once the concentration reaches a threshold, the biomass spreads to adjacent cells using cellular automation algorithm.

Nutrients, drugs, and other chemicals

While a biofilm growth might require several types of nutrient, it is typical to have one nutrient that is the most essential, thus predominantly determines the enzymatic reaction rate. Many numerical simulations pick this rate-limiting nutrient to be dissolved oxygen (DO) [50] [49] [27] [47]. However, some models use a different rate-limiting nutrient such as glucose [28]. Some models keep track of multiple nutrients [69] [35] [41] [46].

In addition to nutrient, some models incorporate drugs or harsh chemicals which inhibit growth, kill bacteria, or erode the EPS [38]. Some allow bacteria to produce waste products which might hamper growth or become a nutrient for other species [46]. Others yet include a chemical signal that bacteria release as a part of quorum sensing. We will refer to these components collectively as chemicals.

Chemicals are usually modeled as a scalar field of concentration, along with Fick's laws of diffusion [69] [50] [49] [13] [17] [52] [47]. For low nutrient concentration, an alternative approach is to apply a diffusion-limited aggregation model [72] [65] [51], in which each individual nutrient molecule does a random walk until it hits a biofilm and gets consumed. This tends to produce biofilms that grow into fractal-like branches instead of a round lump [39].

In models where biomass and solutions can mix, there are two approaches in keeping track of the chemicals,

- Chemicals exist in both biomass and solution. With the chemical concentration c per volume and diffusion constant D_s , the diffusion equation is $\frac{\partial c}{\partial t} = D_s \nabla^2 c$.
- Chemicals exist only in the solution. If a chemical has dissolved concentration c inside the solution of volume fraction ϕ_s , then its spatial concentration is $c\phi_s$. The appropriate diffusion equation is $\frac{\partial c\phi_s}{\partial t} = \nabla \cdot D_s \phi_s (\nabla c)$.

Note that D_s can be modeled as a constant, or a function of concentrations c and ϕ_s .

The source of chemicals can also be modeled differently,

- Fixed concentration at the domain boundary [13] [17] [35] [41] [75] [76]. Thus nutrient reaches the biomass by advection and diffusion.
- Fixed concentration at the biofilm-solution interface [69] [41]. This follows from the assumption that the solution is well-mixed outside the biomass.
- Fixed concentration at the top biofilm height [46].
- Flux balance at the biofilm-solution interface [69], [52]. Let D_{bulk} and $D_{biofilm}$ be the diffusion constants in the bulk fluid and inside biofilm respectively. Let L be the width of diffusion layer. This condition is imposed at the interface,

$$D_{bulk}(c_{bulk} - c_{interface})/L = D_{biofilm} \left. \frac{\partial c}{\partial x} \right|_{interface} . \quad (1.1)$$

- Fixed concentration in the inflow solution [53] [47].
- Fixed boundary flux [51].
- Chemicals produced inside the domain.

The choice of chemical sources partially determines the handling of boundary conditions.

The boundary condition of the bottom side depends on the substratum on which to model.

- Impermeable (zero flux: $\vec{n} \cdot \nabla c = 0$).
- Permeable. The boundary condition depends on specific settings.
- Reactive ($c = 0$).

Rate of growth/consumption

The rates of biofilm growth and nutrient consumption are often modeled by the Monod equation $rate = \frac{rate_{max}c}{K+c}$, where c is the nutrient concentration, $rate_{max}$ is the maximum consumption rate that can be a function of bacterial concentration, and K is a constant known as the half saturation constant. The Monod model gives a nearly linear consumption rate at small c and becomes independent of c as it is large. Note that Monod equation looks exactly the same as Michaelis-Menten equation, thus hinting that enzyme kinetics underlies the growth and consumption process.

If the biomass is represented by cellular automata, growth usually means increasing the number of automata, representing cell division [52] [51]. If the model keeps track of individual bacteria, each cell can grow in volume, shoving its neighbors aside, and split into two smaller cells [35] [46]. If biomass is demarcated by biofilm-water interface, growth is represented by increased biomass volume. The interface location can be directly modified to account for additional volume [69]. Alternatively, the growth can modify local pressure, which drives a flow that eventually expands the biomass volume [13]. If the biomass is represented by a concentration field, growth can be represented by an increase in concentration. In such model, one still needs a mechanism for the biomass to spread out to neighboring regions. Otherwise, the regions that start off without biomass will continue to have no biomass. This spreading can be done by ad-hoc diffusion [17], Cahn-Hilliard dynamics [75] [76], or cellular automata rule [50] [49] [47].

Models can also account for death of bacteria, modeled as an ad-hoc decay [52] [51] [41], killed by drugs, or death by starvation [17] [28] [27] [53].

Hydrodynamics

Some models consider biofilm growth in static medium [69] [50] [49] [52] [51]. Such models consider diffusion, but often ignore advection. When advection is considered, the velocity field can be described by,

- One velocity field which convects everything [13].
- One velocity field for water. Biomass is assumed to be stationary [17].
- Separate velocity fields for water and biofilm [9] [10].
- One mass-averaged velocity field, together with excessive velocities $\mathbf{v}_{water} - \mathbf{v}_{average}$ and $\mathbf{v}_{biofilm} - \mathbf{v}_{average}$ [75] [76] [38].

The velocity field is usually modeled by Darcy's law [13] or the incompressible Navier-Stokes equation [75] [76].

Other features

In many models, one can find an analytic solution or partial solution in one dimension. Higher spatial dimensions usually require numerical simulations. The advantage of modeling in two spatial dimensions or higher lies in the ability to model heterogeneous biofilm-solution surface. More importantly, it is close to the "real thing". Some phenomena like channel flow requires a 3-D simulation. On the other hand, modeling in higher dimensions incurs significantly more computational overhead.

Most simulations are done on a rectangular domain due to its simplicity. Some simulations were carried out on more complicated domain using the finite element method [47].

Some authors incorporate a user interface so that their program can be used by other people [18] [53] [70].

1.3 VISCOELASTIC MODELS

There are many in-depth books on viscoelasticity, which sits at the intersection of rheology, polymer physics, and mechanical engineering, and chemical engineering. We give a very brief overview here. For further details, please consult [5], [3], [8], [25]. Physical theories behind these constitutive models are discussed in [15] and [14].

Viscosity is a generalization of friction. When two flat surfaces rub against each other at a constant normal force, it experiences a friction force $F = -\text{const } \mathbf{v}$. Both friction and viscosity convert kinetic energy into heat. Elasticity is epitomized by a spring. When one pulls a spring by a distance x away from the equilibrium, one experiences an elastic force $F = -\text{const } x$. Elastic force converts kinetic energy to potential energy.

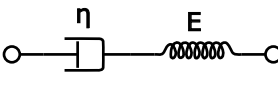
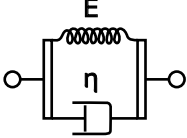
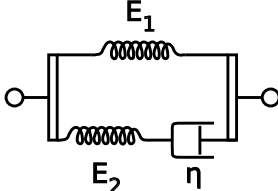
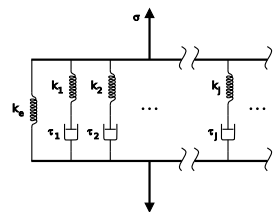
Viscoelastic materials exhibit both viscous and elastic behavior. Kinetic energy is converted partly to potential energy and partly to heat. There are many viscoelastic models. In 1-D, the key variables are stress σ and strain ε . For linear models, the elastic component is represented by a spring with constitutive equation $\sigma = E\varepsilon$ where E is the elastic modulus. The viscous component is represented by a dashpot with constitutive equation $\sigma = \eta \frac{d\varepsilon}{dt}$.

Dashpot and spring can be combined to form a viscoelastic model in several ways, as shown in Table.1.1. Maxwell model accurately describes stress relaxation, while Kelvin-Voigt better describes creep flows. Other models combined more dashpots and springs to improve models' accuracy, at the cost of increased parameters and mathematical complexity. Linear models are suitable for small strain. At larger strain, materials can exhibit non-linear damping or fracture. Alternatively, one can start with the *general linear viscoelastic model*

$$\sigma(t) = \int_{-\infty}^t G(t-t')\dot{\varepsilon}(t')dt' \quad (1.2)$$

where $\dot{\varepsilon} = \frac{d\varepsilon}{dt}$ and $G : [0, \infty) \rightarrow \mathbb{R}$ is called the relaxation modulus. When $G(t) = \eta\delta(t)$ where δ is the Dirac delta function, we get a viscous fluid $\sigma = \eta\dot{\varepsilon}$. When $G(t) = E$ is a constant function, we recover the elastic model $\sigma = E\varepsilon$. For viscoelastic materials, we

Table 1.1: Linear viscoelastic models (Diagrams by Wikipedia user Pekaje)

Model	Schematic	Constitutive equation
Maxwell		$\frac{d\varepsilon}{dt} = \frac{d\varepsilon_{dashpot}}{dt} + \frac{d\varepsilon_{spring}}{dt} = \frac{\sigma}{\eta} + \frac{1}{E} \frac{d\sigma}{dt}$
Kelvin-Voigt		$\sigma = E\varepsilon + \eta \frac{d\varepsilon}{dt}$
Standard linear solid		$\frac{d\varepsilon}{dt} = \frac{E_2}{\eta} \left(\frac{\eta}{E_2} \frac{d\sigma}{dt} + \sigma - E_1\varepsilon \right)$
Generalized Maxwell		$\sigma = \sum_{j=1}^n \sigma_j,$ $\sigma_1 = E_1\varepsilon,$ $\frac{d\varepsilon}{dt} = \frac{\sigma_j}{\eta_j} + \frac{1}{E_j} \frac{d\sigma_j}{dt}, \text{ for } j = 2, \dots, n.$

generally want the stress to be affected mostly by recent shear history. Thus we set $G(t)$ to be a decreasing function tending to zero, for example $G(t) = Ee^{-t/\lambda}$.

Convected derivatives

Before we generalize these viscoelastic models into higher spatial dimensions, we first have to learn about *convected derivatives*. Recall that, for any scalar field ϕ , the time derivative $\frac{\partial \phi}{\partial t}$ in material frame can be converted into the Eulerian frame as the material derivative $\frac{D\phi}{Dt} := \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi$. The extra term originates from the change in material's location due to the flow field \mathbf{v} . In addition to this, the flow field also rotates and bends the basis of the tensor. Scalar quantities don't feel this, but tensor quantities do. Given a basis that convects, rotates, and deforms along with the flow, the time derivative of a second order contravariant tensor τ^{ij} in that frame is translated into the Eulerian frame as

the upper convected time derivative,

$$\overset{\nabla}{\tau} := \frac{D\tau}{Dt} - (\nabla\mathbf{v}) \cdot \tau - \tau \cdot (\nabla\mathbf{v})^T. \quad (1.3)$$

Here $(\nabla\mathbf{v})_{ij} := \frac{dv_i}{dx_j}$. Some books define $(\nabla\mathbf{v})_{ij}$ differently as $\partial_i v_j$, thus gives $\overset{\nabla}{\tau} = \frac{D\tau}{Dt} - (\nabla\mathbf{v})^T \cdot \tau - \tau \cdot (\nabla\mathbf{v})$.

The time derivative of a second order covariant tensor τ_{ij} in that frame translates into Eulerian frame as the lower convected time derivative,

$$\overset{\Delta}{\tau} := \frac{D\tau}{Dt} + (\nabla\mathbf{v})^T \cdot \tau + \tau \cdot (\nabla\mathbf{v}). \quad (1.4)$$

Alternatively, one can think of $\overset{\Delta}{\tau}$ as the time derivative of a contravariant tensor in a frame whose *dual basis* deforms along with the flow.

We now define the Gordon-Schowalter convected derivative. This has one parameter $0 \leq \xi \leq 1$ called slippage. Equivalently, some literature use $a = 1 - 2\xi$, thus $-1 \leq a \leq 1$.

$$\overset{\square}{\tau} := \frac{\partial\tau}{\partial t} := \frac{D\tau}{Dt} + \xi \overset{\nabla}{\tau} + (1 - \xi) \overset{\Delta}{\tau}, \quad (1.5)$$

$$= \frac{D\tau}{Dt} - (\nabla\mathbf{v}) \cdot \tau - \tau \cdot (\nabla\mathbf{v})^T + \xi(\mathbf{D} \cdot \tau + \tau \cdot \mathbf{D}), \quad (1.6)$$

$$= \frac{D\tau}{Dt} - \mathbf{W} \cdot \tau + \tau \cdot \mathbf{W} - a(\mathbf{D} \cdot \tau + \tau \cdot \mathbf{D}), \quad (1.7)$$

where \mathbf{D} and \mathbf{W} are the rate-of-strain tensor and the vorticity tensor respectively,

$$\mathbf{D} = \frac{1}{2}[\nabla\mathbf{v} + \nabla\mathbf{v}^T], \quad \mathbf{W} = \frac{1}{2}[\nabla\mathbf{v} - \nabla\mathbf{v}^T]. \quad (1.8)$$

When $\xi = 0$ ($a = 1$) we recover $\overset{\nabla}{\tau}$. When $\xi = 1$ ($a = -1$) we recover $\overset{\Delta}{\tau}$. When $\xi = \frac{1}{2}$ ($a = 0$) we have $\overset{\square}{\tau} = \frac{D\tau}{Dt} - \mathbf{W} \cdot \tau + \tau \cdot \mathbf{W}$, which is called co-rotational derivative or Jaumann derivative. It is the time-invariant derivative for second order tensors. This arises when the basis rotates but does not deform along with the flow. By comparing (1.6) to (1.3), we can see that $\overset{\square}{\tau}$ is equivalent to taking the upper convected derivative under the effective velocity gradient $\nabla\mathbf{v} - \xi\mathbf{D}$.

For quantities ϕ that have unit ‘‘per volume’’, the material derivative is $\frac{D\phi}{Dt} + \mathbf{v} \cdot \nabla\phi = \frac{\partial\phi}{\partial t} + \mathbf{v} \cdot \nabla\phi + \phi\nabla \cdot \mathbf{v} = \frac{\partial\phi}{\partial t} + \nabla \cdot (\mathbf{v}\phi)$. Thus one often sees the term $\nabla \cdot (\mathbf{v}\phi)$ instead of $\mathbf{v} \cdot \nabla\phi$ in fluid dynamics constitutive equations. The two are equivalent when \mathbf{v} is divergence-free.

Going from 1-D to 3-D

Imagine an infinitesimal cube of fluid, surrounded by fluid of the same type. As fluid flows, neighboring fluid exerts forces on each face of the cube. This force is proportional to surface area: $d\mathbf{F} = \boldsymbol{\tau} \cdot \mathbf{n}dS$, where \mathbf{n} is the unit external normal and dS is the surface element. The quantity $\boldsymbol{\tau}$ is called stress, which is a second order tensor. The angular momentum balance implies that $\boldsymbol{\tau}$ is a symmetric tensor. Divergence theorem yields the force on the cube $d\mathbf{F} = (\nabla \cdot \boldsymbol{\tau})dV$. In continuum mechanics, the word “force” usually is a shorthand for force density per volume $\mathbf{f} := \frac{d\mathbf{F}}{dV} = \nabla \cdot \boldsymbol{\tau}$.

The simplest model of liquid, known as incompressible Newtonian fluid, stipulates that $\boldsymbol{\tau} = 2\eta\mathbf{D}$, where $\mathbf{D} = \frac{1}{2}(\nabla\mathbf{v} + \nabla\mathbf{v}^T)$ is the rate-of-strain tensor and η is the fluid’s viscosity. Assuming homogeneity in the viscosity $\eta = \text{const}$, and using of the incompressibility condition $\nabla \cdot \mathbf{v} = 0$, one can work out the viscous force,

$$\mathbf{f} = \nabla \cdot \boldsymbol{\tau} = \eta\nabla^2\mathbf{v} + \eta\nabla(\nabla \cdot \mathbf{v}) = \eta\nabla^2\mathbf{v} \quad (1.9)$$

The Maxwell constitutive equation from Table.1.1 can be rewritten as $\boldsymbol{\sigma} + \lambda\frac{d\boldsymbol{\sigma}}{dt} = \eta\frac{d\boldsymbol{\varepsilon}}{dt}$, where $\lambda = \frac{\eta}{E}$ is the relaxation time. This generalizes into higher spatial dimensions as the upper convected Maxwell (UCM) model when we replace the time derivative by the convective derivative for second order tensors

$$\boldsymbol{\tau} + \lambda\overset{\nabla}{\boldsymbol{\tau}} = 2\eta\mathbf{D}, \quad (1.10)$$

which is the key ingredient for several viscoelastic models. The commonly used single relaxation time constitutive equation models are given below [5].

- Johnson-Segalman,

$$\boldsymbol{\tau} + \lambda\overset{\square}{\boldsymbol{\tau}} = 2\eta\mathbf{D} \quad (1.11)$$

- Giesekus,

$$\frac{\alpha\lambda}{\eta}\boldsymbol{\tau}^2 + \boldsymbol{\tau} + \lambda\overset{\nabla}{\boldsymbol{\tau}} = 2\eta\mathbf{D}, \quad (1.12)$$

where α is an adjustable model parameter.

- Phan-Thien-Tanner,

$$\frac{\alpha\lambda}{\eta} \text{tr}(\tau)\tau + \tau + \lambda \overset{\nabla}{\tau} = 2\eta\mathbf{D} \quad \text{linear form} \quad (1.13)$$

$$\tau \exp\left(\frac{\alpha\lambda}{\eta} \text{tr}(\tau)\right) + \lambda \overset{\nabla}{\tau} = 2\eta\mathbf{D} \quad \text{exponential form} \quad (1.14)$$

- White-Metzner

$$\tau + \lambda I_2 \overset{\nabla}{\tau} = 2\eta I_2 \mathbf{D} \quad (1.15)$$

where $I_2 = \frac{1}{2}(\mathbf{D} : \mathbf{D} - \text{tr}(\mathbf{D})^2)$ is the second invariant of \mathbf{D} . Incompressibility yields $\text{tr}(\mathbf{D}) = 0$, hence $I_2 = \frac{1}{2}\mathbf{D} : \mathbf{D} = \frac{1}{2}\sum_i \sum_j \mathbf{D}_{ij}^2$.

When Gordon-Schowalter derivative is used, the slippage parameter a is usually set close to 1, that is $\overset{\square}{\tau} \approx \overset{\nabla}{\tau}$. It is justified physically by assuming some slippage in the microscopic network, causing them to travel at a different velocity than \mathbf{v} but not far away from it.

If we assume that the total stress is a combination of UCM and Newtonian stress, we have $\tau = \tau_n + \tau_{ps}$ where $\tau_n + \lambda_1 \overset{\nabla}{\tau} = 2\eta_n \mathbf{D}$ and $\tau_{ps} = 2\eta_{ps} \mathbf{D}$. Then,

$$\tau + \lambda_1 \overset{\nabla}{\tau} = 2(\eta_n + \eta_{ps})\mathbf{D} + 2\lambda_1 \eta_{ps} \overset{\nabla}{\mathbf{D}} \quad (1.16)$$

Let $\eta = \eta_n + \eta_{ps}$ be the total viscosity. Let $\lambda_2 = \lambda_1 \frac{\eta_{ps}}{\eta_n + \eta_{ps}}$ be the retardation time [5]. We have,

$$\tau + \lambda_1 \overset{\nabla}{\tau} = 2\eta(\mathbf{D} + \lambda_2 \overset{\nabla}{\mathbf{D}}). \quad (1.17)$$

This is known as convected Jeffreys model or Oldroyd's fluid type B. Since η_n and η_{ps} are nonnegative, we have $0 \leq \lambda_2 < \lambda_1$. We recover the UCM model when $\lambda_2 = 0$ and Newtonian model when $\lambda_2 = \lambda_1$.

Let \mathbf{I} be the identity. Notice that $\overset{\square}{\mathbf{I}} = 2a\mathbf{D}$. We define the conformation tensor $\mathbf{c} = \tau + \frac{\eta}{a\lambda}\mathbf{I}$. Then the Johnson-Segalman model can be rewritten as,

$$\mathbf{c} + \lambda \overset{\square}{\mathbf{c}} = \frac{\eta}{a\lambda}\mathbf{I}. \quad (1.18)$$

Notice that if \mathbf{c} is initially nonnegative definite, then it will stay nonnegative definite. This reformulation of the Johnson-Segalman model will be exploited to design our numerical method later.

CHAPTER 2

BIOFILM MODEL

We model biofilms using a phase-field based hydrodynamic theory formulation, in which the EPS production and nutrient consumption are effectively accounted for. We treat the biofilm and the ambient fluid as a unified mixture system, in which the biomass, consisted of the bacteria and EPS, is modeled collectively as the polymer solution phase; and the other components, mainly solvent and nutrients, are effectively modeled as an effective solvent phase. The effective solvent is modeled by a Newtonian fluid which is governed by the incompressible Navier-Stokes equation.

The volume inside the mixture domain is divided into two components, the biomass and the solution. A scalar field is introduced to keep track of their volume fraction at each point in the domain. In the pure solvent region in the ambient fluid, the volume fraction of the biomass vanishes. Each component has its own velocity field. However, the boundary condition for each individual velocity field is hard to define and physically measure. To circumvent this, we use a single fluid model, in which a single mass averaged velocity serves as the only measurable macroscopic velocity while the individual velocities are calculated from the intermixing fluxes.

Zhang, Cogan, and Wang [75] [76] have developed, analyzed and simulated such a model in 1-D and 2-D viscous settings. Here, we extend this model to include elastic effect and simulate it in both 2-D and 3-D.

2.1 CONSTITUTIVE EQUATIONS

We model biofilms growing inside a tank of liquid solution. The scalar field ϕ_n denotes the volume fraction of the biomass, which includes bacteria and EPS. Let ϕ_s denote the volume fraction of the liquid solution. We have $\phi_n + \phi_s = 1$, thus only one of the volume fractions needs to be tracked. Let c denote the nutrient concentration level, which floats only inside the solution. Thus, its density per volume is $c\phi_s$.

Let \mathbf{v} be the average velocity, and p be the hydrostatic pressure. The phase field theory for biofilms consists of four sets of equations.

Momentum and continuity equation

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot (a\tau_n + \phi_n\tau_{ps} + \phi_s\tau_s) - [\nabla p + \gamma_1 k_B T \nabla \cdot (\nabla \phi_n \nabla \phi_n)], \quad (2.1)$$

$$\nabla \cdot \mathbf{v} = 0.$$

Here, $\rho = \phi_n\rho_n + \phi_s\rho_s$ is the effective material density for the fluid mixture, where ρ_n and ρ_s are the densities for the biomass and the solvent respectively. The extra stress in the solvent is $\phi_s\tau_s$. In the biomass, there is a viscoelastic stress $a\tau_n$ due to the EPS polymer network and a viscous stress $\phi_n\tau_{ps}$ due to the bacteria. The constant a is the slip coefficient in the Giesekus model. There is no ϕ_n in front of τ_n because we already fold that biomass volume fraction term into τ_n for the reason explained in Section 3.3.

The remaining term is due to the extended Flory-Huggin's mixing free energy density given by,

$$f = \frac{\gamma_1}{2} k_B T |\nabla \phi_n|^2 + \gamma_2 k_B T \left[\frac{\phi_n}{N} \ln(\phi_n + \epsilon) + (1 - \phi_n) \ln(1 - \phi_n) + \chi \phi_n (1 - \phi_n) \right], \quad (2.2)$$

where k_B is the Boltzmann constant, T is the absolute temperature, γ_1 is a parameter measures the strength of the conformation entropy and γ_2 is the strength of the bulk mixing free energy. We note that γ_2 is proportional to the reciprocal of the volume of the solvent molecule. N is an extended polymerization index for the biomass, χ is the mixing parameter, and $\epsilon = 10^{-12}$ is a small dimensionless parameter used to regularize the potential in the pure solvent region.

Transport equation for the volume fraction of the polymer network

$$\frac{\partial \phi_n}{\partial t} + \nabla \cdot (\phi_n \mathbf{v}) = \nabla \cdot \left[\lambda \phi_n \nabla \frac{\delta f}{\delta \phi_n} \right] + g_n, \quad (2.3)$$

where λ is the mobility parameter. The polymer network production rate is given by a modified Michaelis-Menton kinetics,

$$g_n = \mu \phi_n \frac{c}{k_c + c} \left(\frac{\phi_n}{\phi_n + \phi_{min}} \right) \left(1 - \frac{\phi_n}{\phi_{max}} \right) \quad \text{with } \phi_{min} = 0.01 \text{ and } \phi_{max} = 0.20 \quad (2.4)$$

where μ is the maximum production rate, k_c is the half-saturation constant. The purpose of ϕ_{min} is to cap the growth of biomass in the very dilute limit. This represents either stray EPS or planktonic bacteria, neither of which produce EPS. Without the ϕ_{min} term, diluted biomass can outgrow the main buds of the biofilm since it floats closer to the nutrient feeding boundary. The ϕ_{max} term stops the growth once the biomass get very dense. Both are model parameters and can be calibrated through well controlled experiments. The transport equation is a modified or singular Cahn-Hilliard equation with a biomass volume fraction dependent mobility λ . For simplicity, we use a constant λ . In general, it should be proportional to $\phi_s = 1 - \phi_n$. However, ψ_s is always greater than ϕ_n in biofilms. So, the current assumption on λ works fine.

Transport equation for the nutrient

$$\frac{\partial}{\partial t} (\phi_s c) + \nabla \cdot (c \mathbf{v} \phi_s - D_s \phi_s \nabla c) = -g_c, \quad (2.5)$$

where c is the nutrient concentration and the nutrient consumption rate is given by

$$g_c = A \phi_n \frac{c}{k_1 + c}. \quad (2.6)$$

A is the maximum consumption rate, k_1 is the half saturation rate, and D_s is the diffusion constant for the nutrient substrate. Again a Michealis-Menton kinetic is assumed for the decay of the nutrient due to biomass consumption.

Constitutive equations for stress tensors

We model the stress as a sum of three components: viscous stress due to the bacteria $\phi_n \tau_{ps}$, viscous stress due to the solution $\phi_s \tau_s$, and viscoelastic stress due to the EPS network $\tau_n = \phi_n \tilde{\tau}_n$. The first two parts are modelled as Newtonian fluid,

$$\tau_{ps} = 2\eta_{ps}\mathbf{D}_n, \quad \tau_s = 2\eta_s\mathbf{D}_s, \quad (2.7)$$

The elastic stress is modeled by the Giesekus constitutive equation (1.12),

$$\square \tilde{\tau}_n + \frac{\alpha}{\eta} \tilde{\tau}_n^2 + \frac{\tilde{\tau}_n}{\lambda_1} = \frac{2\eta_n}{\lambda_1} \mathbf{D}_n. \quad (2.8)$$

The viscosity coefficients η_n , η_{ps} and η_s are for EPS, bacteria, and solvent respectively. Recall that λ_1 is the relaxation time, $0 \leq \alpha \leq 1$ is the mobility parameter, and a is the slip coefficient. A direct calculation of $\tilde{\tau}_n$ will lead to a big loss in numerical accuracy, as will be explained in Section 3.3. Thus we opt to keep track of the quantity $\tau_n = \phi_n \tilde{\tau}_n$ instead. Use the fact that $\frac{\partial \phi_n}{\partial t} + \nabla \cdot (\mathbf{v}_n \phi_n) \approx \tilde{g}_n \phi_n$ where $\tilde{g}_n = \frac{g_n}{\phi_n}$. The Giesekus constitutive equation becomes,

$$\square \tau_n + \frac{\alpha}{\phi_n \eta_n} \tau_n^2 + \frac{\tau_n}{\lambda_1} = \tilde{g}_n \tau_n + \frac{2\phi_n \eta_n}{\lambda_1} \mathbf{D}_n. \quad (2.9)$$

Expanding the Gordon-Schowalter convected derivative yields,

$$\frac{\partial \tau_n}{\partial t} + \nabla \cdot (\mathbf{v}_n \tau_n) - \mathbf{W}_n \cdot \tau_n + \tau_n \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_n + \tau_n \cdot \mathbf{D}_n) + \frac{\alpha}{\phi_n \eta_n} \tau_n^2 + \frac{\tau_n}{\lambda_1} = \tilde{g}_n \tau_n + \frac{2\phi_n \eta_n}{\lambda_1} \mathbf{D}_n. \quad (2.10)$$

The infinite relaxation time limit $\lambda_1 \rightarrow \infty$ yields the pure elastic theory. In this model, we assume the EPS and bacteria are transported by the same velocity. The biomass velocity is defined by

$$\mathbf{v}_n = \mathbf{v} - \lambda \nabla \frac{\delta f}{\delta \phi_n}, \quad (2.11)$$

which is identified from the transport equation for ϕ_n . Analogously, we can identify the solvent velocity as

$$\mathbf{v}_s = \mathbf{v} + \frac{\lambda \phi_n}{\phi_s} \nabla \frac{\delta f}{\delta \phi_n}. \quad (2.12)$$

The rate of deformation tensor and the vorticity tensor with respect to the average velocity are given by

$$\mathbf{D} = \frac{1}{2}[\nabla \mathbf{v} + \nabla \mathbf{v}^T], \quad \mathbf{W} = \frac{1}{2}[\nabla \mathbf{v} - \nabla \mathbf{v}^T]. \quad (2.13)$$

$\mathbf{D}_n, \mathbf{W}_n, \mathbf{D}_s, \mathbf{W}_s$ are defined analogously by using \mathbf{v}_n and \mathbf{v}_s .

We investigate the dynamics of the biofilm in both 2 and 3 space dimensions. In 2-D, the domain is $(x, y) \in \Omega = [0, L_x] \times [0, L_y]$. The x direction is periodic. In y direction, we impose no-flux boundary conditions.

$$\begin{aligned} [c\mathbf{v}_s\phi_s - D_s\phi_s\nabla c] \cdot \mathbf{n}|_{y=0} &= 0, \\ \nabla\phi_n \cdot \mathbf{n}|_{y=0, L_y} &= 0, \\ \left[\mathbf{v}\phi_n - \lambda\nabla \frac{\delta f}{\delta\phi_n} \right] \cdot \mathbf{n}|_{y=0, L_y} &= 0, \\ \mathbf{v}|_{y=0} &= 0, \quad \mathbf{v}|_{y=L_y} = \mathbf{v}_0. \end{aligned} \quad (2.14)$$

At the top of the domain, the flow velocity \mathbf{v}_0 is specified for shearing flows. We also impose a nutrient feeding condition $c|_{y=L_y} = c^*$ in place of the zero-flux condition there. Our 3-D domain is similar, but has an extra dimension with the periodic boundary condition.

2.2 NONDIMENSIONALIZATION

We use a characteristic time scale t_0 and length scale h to nondimensionalize the variables

$$\tilde{t} = \frac{t}{t_0}, \quad \tilde{\mathbf{x}} = \frac{\mathbf{x}}{h}, \quad \tilde{\mathbf{v}} = \frac{\mathbf{v}t_0}{h}, \quad \tilde{\tau} = \frac{\tau t_0^2}{\rho_0 h^2}, \quad \tilde{p} = \frac{p t_0^2}{\rho_0 h^2}, \quad \tilde{c} = \frac{c}{c_0}, \quad (2.15)$$

where c_0 is a characteristic substrate concentration. The length scale h is determined by the computational geometry while the time scale is done by either the growth time scale of the biofilm or the flow induced time scale. The following dimensionless equations arise

$$\begin{aligned} \Lambda &= \frac{\lambda\rho_0}{t_0}, \quad \Gamma_1 = \frac{\gamma_1 k T t_0^2}{\rho_0 h^4}, \quad \Gamma_2 = \frac{\gamma_2 k T t_0^2}{\rho_0 h^2}, \\ Re_s &= \frac{\rho_0 h^2}{\eta_s t_0}, \quad Re_n = \frac{\rho_0 h^2}{\eta_n t_0}, \quad Re_{ps} = \frac{\rho_0 h^2}{\eta_{ps} t_0}, \quad \tilde{\rho} = \phi_s \frac{\rho_s}{\rho_0} + \phi_n \frac{\rho_n}{\rho_0}, \\ \tilde{D}_s &= \frac{D_s t_0}{h^2}, \quad \tilde{A} = \frac{A t_0}{c_0}, \quad \tilde{\mu} = \mu t_0, \quad \tilde{K}_c = \frac{k_c}{c_0}, \quad \tilde{K}_1 = \frac{k_1}{c_0}, \quad \Lambda_1 = \frac{\lambda_1}{t_0}, \end{aligned} \quad (2.16)$$

where Re_n , Re_{ps} and Re_s are the Reynolds numbers for the EPS, bacteria, and solvent flow respectively. The constant ρ_0 is an averaged density, Λ_1 is the Deborah number. We use the extended Newtonian model for the polymeric stress tensor, and the singular Cahn-Hilliard equation for the biomass volume fraction. For simplicity, we drop the \sim on the dimensionless variables and the parameters. The system of governing equations for the viscoelastic biofilm in these dimensionless variables is given by,

$$\rho \frac{d\mathbf{v}}{dt} = \nabla \cdot (a\phi_n\tau_n + \phi_n\tau_{ps} + \phi_s\tau_s) - [\nabla p + \Gamma_1 \nabla \cdot (\nabla\phi_n \nabla\phi_n)], \quad (2.17)$$

$$\nabla \cdot (\mathbf{v}) = 0, \quad (2.18)$$

$$\frac{\partial\phi_n}{\partial t} + \nabla \cdot (\phi_n\mathbf{v}) = \nabla \cdot (\Lambda\phi_n \nabla \frac{\delta f}{\delta\phi_n}) + g_n, \quad (2.19)$$

$$\frac{\partial}{\partial t}(\phi_s c) + \nabla \cdot (c\mathbf{v}_s\phi_s - D_s\phi_s \nabla c) = -g_c. \quad (2.20)$$

where

$$g_n = \frac{\mu\phi_n c}{k_c + c} \left(\frac{\phi_n}{\phi_n + \phi_{min}} \right) \left(1 - \frac{\phi_n}{\phi_{max}} \right), \quad g_c = \frac{A\phi_n c}{K_1 + c}, \quad (2.21)$$

$$\begin{aligned} \frac{\partial\tau_n}{\partial t} + \nabla \cdot (\mathbf{v}_n\tau_n) - \mathbf{W}_n \cdot \tau_n + \tau_n \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_n + \tau_n \cdot \mathbf{D}_n) \\ + \frac{\tau_n}{\Lambda_1} + \frac{\alpha Re_n \tau_n^2}{\phi_n} = \tilde{g}_n \tau_n + \frac{2\phi}{\Lambda_1 Re_n} \mathbf{D}_n, \end{aligned} \quad (2.22)$$

$$\tau_s = \frac{2}{Re_s} \mathbf{D}_s, \quad \tau_{ps} = \frac{2}{Re_{ps}} \mathbf{D}_n. \quad (2.23)$$

The mixing free energy density is now given by

$$f = \frac{\Gamma_1}{2} |\nabla\phi_n|^2 + \Gamma_2 \left[\frac{\phi_n}{N} \ln(\phi_n + \epsilon) + (1 - \phi_n) \ln(1 - \phi_n) + \chi\phi_n(1 - \phi_n) \right], \quad (2.24)$$

which yields the following component of the excessive network velocity,

$$\nabla \frac{\delta f}{\delta \phi_n} = \nabla \left(\frac{\partial f}{\partial \phi} - \nabla \cdot \left(\frac{\partial f}{\partial \phi_x}, \frac{\partial f}{\partial \phi_y}, \frac{\partial f}{\partial \phi_z} \right) \right) \quad (2.25)$$

$$= \nabla \left(-\Gamma_1 \nabla^2 \phi_n + \Gamma_2 \left(\frac{1}{N} \ln(\phi_n + \epsilon) - \ln(1 - \phi_n) - 2\chi \phi_n + \frac{1}{N} - 1 + \chi \right) \right) \quad (2.26)$$

$$= -\Gamma_1 \nabla (\nabla^2 \phi_n) + \Gamma_2 \left(\frac{1}{N} \frac{1}{\phi_n + \epsilon} + \frac{1}{1 - \phi_n} - 2\chi \right) \nabla \phi_n. \quad (2.27)$$

2.3 REMARK ABOUT THE STRESS CONSTITUTIVE EQUATION

We introduce a new stress tensor

$$\tau_p = \tau_n + Bi\phi_n \mathbf{I} \quad \text{where } Bi = \frac{\eta_n}{a\Lambda_1}. \quad (2.28)$$

The constitutive equation for the new elastic stress tensor becomes,

$$\begin{aligned} \frac{\partial \tau_p}{\partial t} + \nabla \cdot (\mathbf{v}_n \tau_p) - \left(\frac{\partial \phi_n}{\partial t} + \nabla \cdot (\mathbf{v}_n \phi_n) \right) Bi \mathbf{I} - \mathbf{W}_n \cdot \tau_p + \tau_p \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_p + \tau_p \cdot \mathbf{D}_n) \\ + \frac{\alpha Re_n}{\phi_n} (\tau_p - Bi\phi_n \mathbf{I})^2 + \frac{1}{\Lambda_1} (\tau_p - Bi\phi_n \mathbf{I}) = \tilde{g}_n (\tau_p - Bi\phi_n \mathbf{I}) \end{aligned} \quad (2.29)$$

Note that $\left(\frac{\partial \phi_n}{\partial t} + \nabla \cdot (\mathbf{v}_n \nabla \phi_n) \right) = \tilde{g}_n \phi_n$. This let us cancel out two terms and are left with,

$$\begin{aligned} \frac{\partial \tau_p}{\partial t} + \nabla \cdot (\mathbf{v}_n \tau_p) - \mathbf{W}_n \cdot \tau_p + \tau_p \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_p + \tau_p \cdot \mathbf{D}_n) \\ + \frac{\alpha Re_n}{\phi_n} (\tau_p - Bi\phi_n \mathbf{I})^2 + \frac{1}{\Lambda_1} (\tau_p - Bi\phi_n \mathbf{I}) = \tilde{g}_n \tau_p. \end{aligned} \quad (2.30)$$

At any time t , we can approximate the constitutive equation by a difference equation up to $O(\Delta t^2)$,

$$\begin{aligned} \tau_p(\mathbf{x}, t + \Delta t) - Bi\phi_n(\mathbf{x}, t + \Delta t) \mathbf{I} = \left[\left(I + \left(\frac{a+1}{2} \nabla \mathbf{v}_n + \frac{a-1}{2} \nabla \mathbf{v}_n^T \right) \Delta t \right) \right. \\ \cdot \tau_p(\mathbf{x} - \mathbf{v}_n(\mathbf{x}, t) \Delta t, t) \cdot \left(I + \left(\frac{a+1}{2} \nabla \mathbf{v}_n + \frac{a-1}{2} \nabla \mathbf{v}_n^T \right)^T \Delta t \right) \\ \left. - Bi\phi_n(\mathbf{x}, t + \Delta t) \mathbf{I} \right] e^{-\frac{\Delta t}{\Lambda_1}} - \frac{\alpha Re_n}{\phi_n} (\tau_p(\mathbf{x}, t) - Bi\phi_n \mathbf{I})^2 \Delta t \\ - (\nabla \cdot \mathbf{v}_n) \tau_p \Delta t + \tilde{g}_n \tau_p \Delta t + O(\Delta t^2). \end{aligned} \quad (2.31)$$

Alternatively, we have the difference equation in another form,

$$\begin{aligned} \tau_p(\mathbf{x}, t + \Delta t) - Bi\phi_n(\mathbf{x}, t + \Delta t)\mathbf{I} &= \left[\left(I + \left(\frac{(a+1)}{2} \nabla \mathbf{v}_n + \frac{(a-1)}{2} \nabla \mathbf{v}_n^T \right) \Delta t \right) \right. \\ \cdot \tau_p(\mathbf{x} - \mathbf{v}_n(\mathbf{x}, t)\Delta t, t) \cdot &\left(I + \left(\frac{(a+1)}{2} \nabla \mathbf{v}_n + \frac{(a-1)}{2} \nabla \mathbf{v}_n^T \right)^T \Delta t \right) e^{-\frac{\Delta t}{\Lambda_1} + \int_t^{t+\Delta t} (\bar{g}_n - \nabla \cdot \mathbf{v}_n) dt} \\ &\left. - Bi\phi_n(\mathbf{x}, t + \Delta t)\mathbf{I} e^{-\frac{\Delta t}{\Lambda_1}} \right] - \frac{\alpha Re_n}{\phi_n} (\tau_p(\mathbf{x}, t) - Bi\phi_n\mathbf{I})^2 \Delta t + O(\Delta t^2). \quad (2.32) \end{aligned}$$

If we take the derivative of the equation with respect to Δt and evaluate it at $\Delta t = 0$, we recover the Giesekus equation. This difference equation will be the basis for us to design the numerical method to solve the Giesekus equation coupled with the momentum transport. These difference equations are in fact the result of conducting first order backward differencing of the convected derivative along the streamline. For comparison purpose, we also conduct simulations where we replace \mathbf{v}_n by the average velocity \mathbf{v} in the Giesekus constitutive equation.

To avoid the singularity in the damping term, numerically in our simulations, we replace $\frac{\alpha Re_n}{\phi_n} (\tau_p - Bi\phi_n\mathbf{I})^2$ by $\frac{\alpha Re_n}{\max(\epsilon_\phi, \phi_n)} (\tau_p - Bi\phi_n\mathbf{I})^2$, where ϵ_ϕ is a numerical parameter. A smaller ϵ_ϕ will make the equation stiffer, necessitating a smaller time step Δt . To run at a practical time step, we use $\epsilon_\phi = 10^{-2} - 10^{-3}$.

This adjustment reduces the damping effect in the region where $\phi_n < \epsilon_\phi$ by a factor of ϵ_ϕ/ϕ_n . In regions where $\phi_n \ll \epsilon_\phi$, the model provides almost no damping at all, even though Sec.3.3 shows that these are the regions where damping is most needed. To compensate for it, we let the stress decay quickly in those regions by reducing the relaxation time, $\Lambda_1(\vec{x}, t) := \Lambda_1 \left(\frac{\phi_n}{\phi_n + \epsilon_\phi} \right)^2$. Since this numerical modification of the constitutive equation occurs in the region with dilute EPS, it should not effect the overall biomass dynamics.

We note that the elastic contribution to the force can be calculated from either τ_p or τ_n since $\mathbf{F} = \nabla \cdot \tau_n = \nabla \cdot (\tau_p - Bi\phi_n\mathbf{I}) = \nabla \cdot \tau_p - Bi\nabla\phi_n$. The last term is a potential force, which can be absorbed into the pressure.

CHAPTER 3

NUMERICAL SCHEMES AND GPU IMPLEMENTATION

3.1 NUMERICAL SCHEME

We use the finite difference method to solve the coupled flow, the phase field equation, the elastic stress constitutive equation, and the nutrient transport equation. We solve the coupled momentum transport equation and the continuity equation using a Uzawa-Gauge scheme developed by Guermond et al. [23]. In order to apply the fast Fourier transform (FFT), the momentum transport equation is rewritten as,

$$\rho\left(\frac{\partial}{\partial t}\mathbf{v} + \mathbf{v} \cdot \nabla\mathbf{v}\right) - \frac{1}{Re_a}\nabla^2\mathbf{v} = -\nabla p + \mathbf{R} - \frac{1}{Re_a}\nabla^2\mathbf{v}, \quad (3.1)$$

with,

$$\mathbf{R} = -\Gamma_1\nabla^2\phi_n\nabla\phi_n + \nabla \cdot (a\tau_n + \phi_n\tau_{ps} + \phi_s\tau_s), \quad (3.2)$$

where Re_a is an averaged Reynolds number. Our choice of Re_a will be discussed later in this section.

We use a uniform time step Δt . For simplicity, the second order extrapolation of any function f in time is denoted by $\bar{f}^{n+1} = 2f^n - f^{n-1}$. We calculate \mathbf{v} and the pressure in three steps.

Step 1:

$$\begin{cases} \rho^{n+1}\left[\frac{3\mathbf{u}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t}\right] + \rho^{n+1}\bar{\mathbf{v}}^{n+1} \cdot \nabla\bar{\mathbf{v}}^{n+1} + \frac{1}{2}(\nabla \cdot (\rho^{n+1}\bar{\mathbf{v}}^{n+1}))\bar{\mathbf{v}}^{n+1} \\ \quad + \frac{1}{Re_s}\nabla s^n - \frac{1}{Re_a}\nabla^2\mathbf{u}^{n+1} = \mathbf{R}^{n+1} - \frac{1}{Re_a}\nabla^2\mathbf{v}^{n+1-\epsilon}, \\ \mathbf{u}^{n+1}|_{y=0} = 0, \quad \mathbf{u}^{n+1}|_{y=L_y} = \mathbf{v}_0. \end{cases} \quad (3.3)$$

Step 2: We implement the projection step by solving a Poisson equation with the Neumann boundary condition:

$$\begin{cases} -\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla \psi^{n+1} \right) = \nabla \cdot \mathbf{u}^{n+1}, \\ \frac{\partial \psi^{n+1}}{\partial n} \Big|_{y=0,H} = 0. \end{cases} \quad (3.4)$$

Step 3: We correct the velocity, pressure and the auxiliary variable s .

$$\begin{cases} \mathbf{v}^{n+1} = \mathbf{u}^{n+1} + \frac{1}{\rho^{n+1}} \nabla \psi^{n+1}, \\ s^{n+1} = s^n - \nabla \cdot \mathbf{u}^{n+1}. \end{cases} \quad (3.5)$$

Here $s^0 = 0$ and $\mathbf{v}^1, s^1, \phi_n^1, c^1$ are computed by a first order scheme. Note that (3.3) and (3.4) can be solved quickly by FFT, as explained in section A. Most terms in (3.3) are the temporally second order discretization of (3.1) at step $n + 1$. The pressure scheme and the term $\frac{1}{Re_a} \overline{\nabla^2 \mathbf{v}^{n+1-\epsilon}}$ are kept first order in time due to stability reasons, which we will discuss in section 3.2. We use $\epsilon = 0.05$. The pressure does not show up explicitly in the scheme, but can be approximated by $p^{n+1} = -\frac{3\psi^{n+1}}{2\Delta t} + \frac{1}{Re_s} s^{n+1}$.

The phase field equation for the biomass volume fraction ϕ_n is discretized by

$$\begin{aligned} \frac{3\phi_n^{n+1} - 4\phi_n^n + \phi_n^{n-1}}{2\Delta t} + \mathbf{v}^{n+1} \cdot \nabla \phi_n^{n+1} = g_n^{n+1} + \Lambda \nabla \cdot \left[-\Gamma_1 \overline{\phi_n^{n+1}} \nabla (\nabla^2 \phi_n^{n+1}) \right. \\ \left. + \Gamma_2 \overline{\phi_n^{n+1}} \left(\frac{1}{N(\overline{\phi_n^{n+1}} + \epsilon)} + \frac{1}{1 - \overline{\phi_n^{n+1}}} - 2\chi \right) \nabla \phi_n^{n+1} \right]. \end{aligned} \quad (3.6)$$

The substrate concentration transport equation is discretized by

$$\frac{3\phi_s^{n+1} c^{n+1} - 4\phi_s^n c^n + \phi_s^{n-1} c^{n-1}}{2\Delta t} + \mathbf{v}^{n+1} \cdot \nabla (c^{n+1} \phi_s^{n+1}) = g_c^{n+1} + \nabla \cdot (D_s \phi_s^{n+1} \nabla c^{n+1}). \quad (3.7)$$

Both ϕ_n and c are solved after the velocity is updated. Since these equations are coupled through the forcing terms, they are supposed to be solved simultaneously. However, we could decouple them by solving one in front of the other. In this case, we first solve the equation for ϕ_n in which the nutrient concentration is extrapolated to $n + 1$ time step and then we solve for c .

When shearing, the Peclet number is very high. Thus, we advect ϕ_n in a separate step by WENO, and then couple the result with the phasefield equation through a first-order operator splitting method,

$$\phi_n^* = \text{WENO}(\phi_n^n, \mathbf{v}^n, \mathbf{v}^{n+1}) \quad (3.8)$$

$$\frac{\phi_n^{n+1} - \phi_n^*}{\Delta t} = g_n^{n+1} + \Lambda \nabla \cdot \left[-\Gamma_1 \phi_n^* \nabla (\nabla^2 \phi_n^{n+1}) \right. \quad (3.9)$$

$$\left. + \Gamma_2 \phi_n^* \left(\frac{1}{N(\phi_n^* + \epsilon)} + \frac{1}{1 - \phi_n^*} - 2\chi \right) \nabla \phi_n^{n+1} \right]. \quad (3.10)$$

The WENO step uses the third-order TVD Runge-Kutta time discretization,

$$\phi_n^{(1)} = \phi_n^n + \Delta t L(\phi_n^n, \mathbf{v}^n) \quad (3.11)$$

$$\phi_n^{(2)} = \frac{3}{4} \phi_n^n + \frac{1}{4} \phi_n^{(1)} + \frac{1}{4} \Delta t L(\phi_n^{(1)}, \mathbf{v}^{n+1}) \quad (3.12)$$

$$\phi_n^* = \frac{1}{3} \phi_n^n + \frac{2}{3} \phi_n^{(2)} + \frac{2}{3} \Delta t L(\phi_n^{(2)}, \frac{\mathbf{v}^n + \mathbf{v}^{n+1}}{2}), \quad (3.13)$$

where $L(\phi_n, \mathbf{v}) \approx -\nabla \cdot (\mathbf{v} \phi_n)$ is an evaluation of the advection term. We use the third-order WENO scheme. For clarity, we first describe the discretization for a 1-D advection in x direction, given by $(L_x(\phi_n, v_x))_i = -\frac{1}{\Delta x} \left((\overline{\phi_n v_x})_{i+\frac{1}{2}} - (\overline{\phi_n v_x})_{i-\frac{1}{2}} \right)$. The numerical flux $(\overline{\phi_n v_x})_{i+\frac{1}{2}} = \frac{\omega_1}{\omega_1 + \omega_2} (\overline{\phi_n v_x})_{n,i+\frac{1}{2}}^{(1)} + \frac{\omega_2}{\omega_1 + \omega_2} (\overline{\phi_n v_x})_{n,i+\frac{1}{2}}^{(2)}$ is a weighted average of two second-order accurate fluxes, discretized on two different stencils. In regions where $v_x \geq 0$, they are given by,

$$(\overline{\phi_n v_x})_{n,i+\frac{1}{2}}^{(1)} = -\frac{1}{2} (\phi_n v_x)_{i-1} + \frac{3}{2} (\phi_n v_x)_i, \quad (\overline{\phi_n v_x})_{n,i+\frac{1}{2}}^{(2)} = \frac{1}{2} (\phi_n v_x)_i + \frac{1}{2} (\phi_n v_x)_{i+1}. \quad (3.14)$$

The nonlinear weights are $\omega_k = \frac{\gamma_k}{(\epsilon + \beta_k)^2}$ for $k = 1, 2$, where $\epsilon = 10^{-6}$ and the linear weights $\gamma_1 = \frac{1}{3}$ and $\gamma_2 = \frac{2}{3}$. The smoothness indicators are given by

$$\beta_1 = ((\phi_n v_x)_i - (\phi_n v_x)_{i-1})^2, \quad \beta_2 = ((\phi_n v_x)_{i+1} - (\phi_n v_x)_i)^2. \quad (3.15)$$

In regions where $v_x < 0$, the extrapolated numerical flux $(\overline{\phi_n v_x})_{i+\frac{1}{2}}^{(1)}$ and the smoothness indicator β_1 are replaced by a similar extrapolation but from gridpoints $i+1$ and $i+2$. For

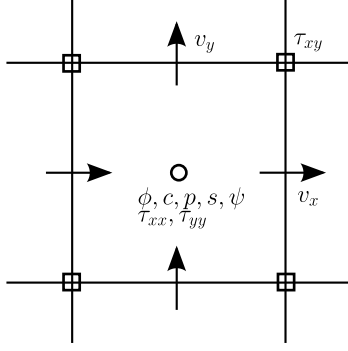


Figure 3.1: Variable locations on the 2-D staggered grid

the advection in higher spatial dimensions, we use the operator addition scheme $L(\phi_n, \mathbf{v}) = L_x(\phi_n, v_x) + L_y(\phi_n, v_y) + L_z(\phi_n, v_z)$.

We assume that the difference between the solvent density and the biomass density is negligible. Then, the density of the solvent and the polymer network are set to be the same; thus ρ^n is in fact a constant. The averaged Reynolds number Re_a is computed by

$$\frac{1}{Re_a} = a \frac{\phi_{max}^n}{Re_n} + \frac{\phi_{max}^n}{Re_{ps}} + \frac{(1 - \phi_{max}^n)}{Re_s}. \quad (3.16)$$

where $\phi_{max}^n = \max\{\phi_{n,i,j}^n \mid 1 \leq i \leq M_x, 1 \leq j \leq M_y\}$. Thus Re_a is a constant at each time step t_n , but varies with time.

For spatial discretization, we use a uniform staggered grid. Its two dimensional version is depicted in Fig.3.1. The computation domain $\Omega = [0, L_x] \times [0, L_y]$ is divided into uniform cells of size $\Delta x = L_x/M_x$, $\Delta y = L_y/M_y$. Values of ϕ_n, c, p are located at cell centers. For example $c_{i,j}^n$ denotes the value of the numerical solution of the nutrient equation (3.7) at time $n\Delta t$ at the point $(x_i, y_j) := ((i - \frac{1}{2})\Delta x, (j - \frac{1}{2})\Delta y)$ for $i = 1, \dots, M_x$ and $j = 1, \dots, M_y$.

The velocity $\mathbf{v} = (v_x, v_y)$ is discretized at the center of cell surfaces as follows,

$$v_{x,i,j} = v_x(x_i + \frac{\Delta x}{2}, y_j) \quad v_{y,i,j} = v_y(x_i, y_j + \frac{\Delta y}{2}) \quad i = 0, \dots, M_x; \quad j = 0, \dots, M_y. \quad (3.17)$$

For the elastic stress, the normal components τ_{xx} and τ_{yy} are discretized at cell centers,

e.g., $\tau_{xx,ij} = \tau_{xx}(x_i, y_j)$. The shear components are discretized at cell corners,

$$\tau_{xy,ij} = \tau_{xy} \left(x_i + \frac{\Delta x}{2}, y_j + \frac{\Delta y}{2} \right). \quad (3.18)$$

At the top and bottom boundaries $y = L_y, 0$ we have a no-flux boundary condition $\mathbf{v} \cdot \mathbf{n}|_{0,L_y} = 0$. The boundary conditions for ϕ_n and c given by (2.14) become

$$\nabla c \cdot \mathbf{n}|_{y=0,L_y} = 0, \quad \nabla \phi_n \cdot \mathbf{n}|_{y=0,L_y} = 0, \quad \nabla \frac{\delta f}{\delta \phi_n} \cdot \mathbf{n}|_{y=0,L_y} = 0. \quad (3.19)$$

The discrete forms of the boundary conditions are handled by introducing ghost cells right outside the boundaries. The conditions (3.19) translate into,

$$\phi_{n,i,1}^n = \phi_{n,i,0}^n, \quad \phi_{n,i,2}^n = \phi_{n,i,-1}^n, \quad \phi_{n,i,M_y+1}^n = \phi_{n,i,M_y}^n, \quad \phi_{n,i,M_y+2}^n = \phi_{n,i,M_y-1}^n, \quad (3.20)$$

$$c_{i,1}^n = c_{i,0}^n, \quad c_{i,M_y+1}^n = c_{i,M_y}^n, \quad \text{for } i = 1, \dots, M_x. \quad (3.21)$$

The spatial discretizations for ϕ_n and c are done using central differences to ensure at least second order accuracy in space.

The numerical method for the elastic stress (2.31) is discretized in first order both temporally and spatially, given by the following three steps.

1. $\tau_{p,ij}^* = \tau_p^n(\mathbf{x}_{ij} - \mathbf{v}_{n,ij}^n \Delta t)$ interpolated linearly.
2. $\tau_{p,ij}^{**} = \left(I + \left(\frac{a+1}{2} (\nabla \mathbf{v}_n)_{ij}^n + \frac{a-1}{2} (\nabla \mathbf{v}_n)_{ij}^{nT} \right) \Delta t \right) \cdot \tau_{p,ij}^* \cdot \left(I + \left(\frac{a+1}{2} (\nabla \mathbf{v}_n)_{ij}^n + \frac{a-1}{2} (\nabla \mathbf{v}_n)_{ij}^{nT} \right)^T \Delta t \right)$.
3. $\tau_{p,ij}^{n+1} = \frac{\tau_{p,ij}^{**} e^{-\frac{\Delta t}{\Lambda_1}} + Bi \phi_n \mathbf{I} \left(1 - e^{-\frac{\Delta t}{\Lambda_1}} \right) - \frac{\alpha Re_n}{\phi_n^{n+1}} \Delta t (\tau_{p,ij}^n - Bi \phi_n^{n+1} \mathbf{I})^2}{1 - \tilde{g}_n^{n+1} \Delta t + \nabla \cdot \mathbf{v}_n \Delta t}$

or alternatively

$$\tau_{p,ij}^{n+1} = \tau_{p,ij}^{**} e^{-\frac{\Delta t}{\Lambda_1} + \frac{1}{2} (\tilde{g}_n^n + \tilde{g}_n^{n+1} - \nabla \cdot \mathbf{v}_n^n - \nabla \cdot \mathbf{v}_n^{n+1}) \Delta t} + Bi \phi_n^{n+1} \mathbf{I} \left(1 - e^{-\frac{\Delta t}{\Lambda_1}} \right) - \frac{\alpha Re_n}{\phi_n^{n+1}} \Delta t (\tau_{p,ij}^n - Bi \phi_n^{n+1} \mathbf{I})^2$$

All second order tensors such as $\tau_{p,ij}^*$, $\tau_{p,ij}^{**}$, $(\nabla \mathbf{v}_n)_{ij}$ have their components discretized at the same locations as $\tau_{p,ij}$. The interpolation in step (1) is done using the four nearest grid

points. For example, given $a, b \in (0, 1)$, the four nearest grid points of $\tau_{p,xy}^n(a\Delta x, b\Delta y)$ are at points $(0, 0)$, $(0, \Delta y)$, $(\Delta x, 0)$ and $(\Delta x, \Delta y)$. We use the interpolation,

$$\begin{aligned} \tau_p^n(a\Delta x, b\Delta y) &= (1-a)(1-b)\tau(0,0) + (1-a)b\tau_p^n(0, \Delta y) \\ &\quad + a(1-b)\tau_p^n(\Delta x, 0) + ab\tau_p^n(\Delta x, \Delta y). \end{aligned} \quad (3.22)$$

Note that τ_p does not need an explicit boundary condition at $y = 0, L_y$ since the backward interpolation along the characteristic line always falls within the domain. The evaluation of the damping term $(\tau_{p,ij}^n - Bi\phi_n^{n+1}\mathbf{I})^2$ requires values from various components of τ_n , which locates at different grid locations. When a value is needed outside its native grid location, we use the average of the values from the nearest neighbors. If the point in question lies on the boundary $y = 0, L_y$, we average only among the nearest neighbors which lie within the domain.

The effective equation of this numerical scheme for τ_p is,

$$\begin{aligned} &\frac{\partial \tau_p}{\partial t} + \nabla \cdot (\mathbf{v}_n \tau_p) + \left(1 - \frac{\Delta t}{\Lambda_1}\right) \left[-\mathbf{W}_n \cdot \tau_p + \tau_p \cdot \mathbf{W}_n - a[\mathbf{D}_n \cdot \tau_p - \tau_p \cdot \mathbf{D}_n^T] + \frac{1}{\Lambda_1}(\tau_p - Bi\phi\mathbf{I}) \right] \\ &\quad - \frac{\Delta t}{\Lambda_1} \mathbf{v}_n \cdot \nabla \tau_p + \Delta t (\nabla \cdot \mathbf{v}_n) \frac{\partial \tau_p}{\partial t} + \frac{\Delta t}{2} \frac{\partial^2 \tau_p}{\partial t^2} \\ &\quad + \Delta t \left((\nabla \mathbf{v}_n)(\mathbf{v}_n \cdot \nabla \tau_p) + (\mathbf{v}_n \cdot \nabla \tau_p)(\nabla \mathbf{v}_n)^T - (\nabla \mathbf{v}_n) \tau_p (\nabla \mathbf{v}_n)^T \right) \\ &= \tilde{g}_n(\tau_p + \Delta t \frac{\partial \tau_p}{\partial t}) - \frac{\alpha Re_n}{\phi_n} (\tau_p - Bi\phi_n \mathbf{I})^2 + \frac{\mathbf{v}_n \Delta x}{2} \nabla^2 \tau_p + O(\Delta x^2, \Delta t^2, \Delta t \Delta x) \end{aligned} \quad (3.23)$$

As can be seen from (2.31), (2.32) and the scheme itself, τ_p should be non-negative definite if $a > 0$ and the initial value $\tau_p(\mathbf{x}, 0)$ is everywhere positive definite. This property can be used to partially check the validity of the code. Note, in contrast, that τ_n is generally not positive definite.

Numerically, the non-negative definiteness of τ_p holds only when all components of τ_p are computed at the same grid lattice. When components of τ_p are not co-located, the smallest eigenvalue λ_{min} might become a little negative in some region. This is easy to occur since $\lambda_{min} \ll \lambda_{max}$, thus a small truncation error during the advection of the big eigen component can substantially affect the small eigen component.

3.2 MOMENTUM EQUATION

During the implementation of the momentum equation, we encountered some unanticipated numerical issues. We discuss them here so that those who wish to reproduce this work can be aware of these subtleties.

Our numerical scheme (3.3)(3.4)(3.5) for the incompressible Navier-Stokes equation (2.1) is based on the Gauge-Uzawa method. A comprehensive overview of that method is given in [23]. It has been extended to cover the case of variable density and viscosity in [58]. For a constant density, the temporally first order numerical scheme is,

$$\begin{cases} \rho^{n+1} \left(\frac{\mathbf{u}^{n+1} - \mathbf{v}^n}{\Delta t} \right) + \rho^n \mathbf{v}^n \cdot \nabla \mathbf{u}^{n+1} \\ \quad + \frac{1}{2} (\nabla \cdot (\rho^n \mathbf{v}^n)) \mathbf{u}^{n+1} + \eta_{pres} \nabla s^n = \mathbf{F}_{CH}^n + \mathbf{F}_{stress}^n \\ \mathbf{u}^{n+1}|_{y=0} = 0, \quad \mathbf{u}^{n+1}|_{y=L_y} = \mathbf{v}_0. \end{cases} \quad (3.24)$$

$$\begin{cases} -\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla \psi^{n+1} \right) = \nabla \cdot \mathbf{u}^{n+1}, \\ \frac{\partial \psi^{n+1}}{\partial n} |_{y=0,H} = 0. \end{cases} \quad (3.25)$$

$$\begin{cases} \mathbf{v}^{n+1} = \mathbf{u}^{n+1} + \frac{1}{\rho^{n+1}} \nabla \psi^{n+1}, \\ s^{n+1} = s^n - \nabla \cdot \mathbf{u}^{n+1}. \end{cases} \quad (3.26)$$

Pressure does not show up explicitly in the scheme, but can be recovered by $p^{n+1} = -\frac{\psi^{n+1}}{\Delta t} + \eta_{pres} s^{n+1}$. The temporally second order numerical scheme is,

$$\begin{cases} \rho^{n+1} \left(\frac{3\mathbf{u}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) + \rho^{n+1} \bar{\mathbf{v}}^{n+1} \cdot \nabla \mathbf{u}^{n+1} \\ \quad + \frac{1}{2} (\nabla \cdot (\rho^{n+1} \bar{\mathbf{v}}^{n+1})) \mathbf{u}^{n+1} + \nabla p^n + \eta_{pres} \nabla s^n = \mathbf{F}_{CH}^{n+1} + \mathbf{F}_{stress}^{n+1} \\ \mathbf{u}^{n+1}|_{y=0} = 0, \quad \mathbf{u}^{n+1}|_{y=L_y} = \mathbf{v}_0. \end{cases} \quad (3.27)$$

$$\begin{cases} -\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla \psi^{n+1} \right) = \nabla \cdot \mathbf{u}^{n+1}, \\ \frac{\partial \psi^{n+1}}{\partial n} |_{y=0,H} = 0. \end{cases} \quad (3.28)$$

$$\begin{cases} \mathbf{v}^{n+1} = \mathbf{u}^{n+1} + \frac{1}{\rho^{n+1}} \nabla \psi^{n+1}, \\ s^{n+1} = s^n - \nabla \cdot \mathbf{u}^{n+1}. \\ p^{n+1} = p^n - \frac{3\psi^{n+1}}{2\Delta t} + \eta_{pres} s^{n+1}. \end{cases} \quad (3.29)$$

The scheme (3.3)(3.4)(3.5) that we use is a hybrid of the temporally first and second order scheme. To minimize truncation errors, we discretize most terms to the second order. However, we use the first order pressure correction scheme (3.24)(3.25)(3.26), for a stability reason that will be discussed in Lemma 3.2.1.

Note that [58] uses $\eta_{pres} = \min \eta = \eta_s$. The term \mathbf{F}_{CH} is the force due to Cahn-Hilliard dynamics, and \mathbf{F}_{stress} is the force due to stress. These forces in our model differ from those in [58]. In our model, they are given by,

$$\mathbf{F}_{CH} = -\nabla \cdot (\Gamma_1 \nabla \phi_n \nabla \phi_n) \quad (3.30)$$

$$\mathbf{F}_{stress} = \nabla \cdot (a\tau_n + \phi_n \tau_n + \phi_s \tau_s) \quad (3.31)$$

$$\tau_n \text{ is discussed in Sec. 3.3} \quad (3.32)$$

$$\tau_{ps} = 2\eta_{ps} \mathbf{D}_n, \quad \mathbf{D}_n = \frac{1}{2} [\nabla \mathbf{v}_n + \nabla \mathbf{v}_n^T], \quad \mathbf{v}_n = \mathbf{v} - \lambda \nabla \frac{\delta f}{\delta \phi_n}, \quad (3.33)$$

$$\tau_s = 2\eta_s \mathbf{D}_s, \quad \mathbf{D}_s = \frac{1}{2} [\nabla \mathbf{v}_s + \nabla \mathbf{v}_s^T], \quad \mathbf{v}_s = \mathbf{v} + \frac{\lambda \phi_n}{\phi_s} \nabla \frac{\delta f}{\delta \phi_n}. \quad (3.34)$$

The scheme given above needs a few modifications to be applicable to our need. Most of these are necessitated by the high viscosity variation in our model $\frac{\eta_n + \eta_{ps}}{\eta_s} \sim 10^4 - 10^5$, and the relatively big timestep $\Delta t \gg \frac{\rho}{\eta} (\Delta x)^2$. This assumption for Δt is required in order for our computation to finish in a reasonable amount of time. To put this timestep size in perspective, recall that the stability condition for forward-time central-space scheme for Navier-Stokes equation is $\Delta t < \frac{1}{2} \frac{\rho}{\eta} (\Delta x)^2$.

One issue we notice is that, in the second order pressure correction scheme, the pressure terms p and s would oscillate over time. This oscillation usually does not cause a noticeable problem. However, once in a while, it can spiral out of hand and cause the simulation to blow up. We track the issue back to this lemma,

Lemma 3.2.1. *If the viscosity variation is very high ($\eta \gg \eta_{pres}$), and timestep $\Delta t \gg \frac{\rho}{\eta}(\Delta x)^2$, then the scheme given by (3.27)(3.28)(3.29) can show an oscillatory behavior.*

Proof. Consider the case of 1-D periodic domain $\Omega = [0, 2\pi]$, with constant density ρ and constant viscosity η such that $\eta \gg \eta_{pres}$. The incompressibility condition yields that $\mathbf{v}^n(x) = 0$. We perform a stability analysis on the remaining variables. For any given wave number $k \in \mathbb{Z}$, let $s^n(x) = S^n e^{ikx}$, $p^n(x) = P^n e^{ikx}$, $\mathbf{u}^n(x) = U^n e^{ikx}$ where $S^n, P^n, U^n \in \mathbb{C}$. Then, (3.27) becomes,

$$\nabla(p^n + \eta_{pres} S^n) = \eta \nabla^2 \mathbf{u} - \frac{3\rho}{2\Delta t} \mathbf{u}^{n+1} \quad (3.35)$$

$$ik(P^n + \eta_{pres} S^n) = \left(-\eta k^2 - \frac{3\rho}{2\Delta t}\right) U^{n+1} \quad (3.36)$$

(3.28) and the first equation in (3.29) yields,

$$\psi^{n+1} = -\frac{\rho}{ik} \mathbf{u}^{n+1} \quad (3.37)$$

The last two equations in (3.29) yield,

$$S^{n+1} = S^n - ikU^{n+1} \quad (3.38)$$

$$P^{n+1} = P^n - \frac{3}{2\Delta t} \left(-\frac{\rho}{ik}\right) U^{n+1} + \eta_{pres} S^{n+1} \quad (3.39)$$

Then substitute in U^{n+1} from (3.36), we get

$$S^{n+1} = S^n - \frac{k^2}{\eta k^2 + \frac{3\rho}{2\Delta t}} (P^n + \eta_{pres} S^n) \quad (3.40)$$

$$P^{n+1} = P^n - \frac{1}{\eta k^2 \frac{2\Delta t}{3\rho} + 1} (P^n + \eta_{pres} S^n) + \eta_{pres} S^{n+1} \quad (3.41)$$

For high wave numbers ($k \approx \frac{1}{\Delta x}$), the timestep assumption $\Delta t \gg \frac{\rho}{\eta}(\Delta x)^2$ yields that $\eta k^2 \gg \frac{3\rho}{2\Delta t}$. Therefore, $\frac{k^2}{\eta k^2 + \frac{3\rho}{2\Delta t}} \approx \frac{1}{\eta}$ and $\frac{1}{\eta k^2 \frac{2\Delta t}{3\rho} + 1} \approx 0$. We are left with,

$$S^{n+1} \approx S^n - \frac{1}{\eta} (P^n + \eta_{pres} S^n) \quad (3.42)$$

$$P^{n+1} \approx P^n + \eta_{pres} S^{n+1} \quad (3.43)$$

Let $Q^n = P^n / \eta_{pres}$ and $\epsilon = \eta_{pres} / \eta$,

$$S^{n+1} \approx S^n - \epsilon(Q^n + S^n) \quad (3.44)$$

$$Q^{n+1} \approx Q^n + S^{n+1} \quad (3.45)$$

Thus,

$$\begin{bmatrix} S^{n+1} \\ Q^{n+1} \end{bmatrix} = \begin{bmatrix} 1 - \epsilon & -\epsilon \\ 1 - \epsilon & 1 - \epsilon \end{bmatrix} \begin{bmatrix} S^n \\ Q^n \end{bmatrix} \quad (3.46)$$

The characteristic polynomial of the matrix is $((1 - \epsilon) - \lambda)^2 + (1 - \epsilon)\epsilon = 0$. The eigenvalues are $\lambda = 1 - \epsilon \pm i\sqrt{\epsilon(1 - \epsilon)} \approx e^{-\frac{\epsilon}{2} \pm i\sqrt{\epsilon}}$. Thus $P^n \approx e^{(-\frac{\epsilon}{2} \pm i\sqrt{\epsilon})n} P^0$. Since $\sqrt{\epsilon} \gg \frac{\epsilon}{2}$, we expect P^n , and thus the pressure, to decay slowly and exhibit an oscillatory behavior. \square

We will encounter this kind of behavior again. So it's worth pointing out that for $0 < \lambda \ll \omega$, the function $f(t) = e^{(-\lambda + i\omega)t}$ represents a damped oscillation in which the decay is much slower than the oscillation. Though analytically stable, its governing equation is hyperbolic. Solving this by the straight forward time marching method often leads to instability.

Since pressure oscillation occurs when η / η_{pres} is small, we are tempted to set η_{pres} to a big value, for example let $\eta_{pres} = \max \eta$. However, this will lead to another instability.

Lemma 3.2.2. *If $\Delta t \gg \frac{\eta_{min}}{\eta_{max}}$ and $\eta_{pres} \gg \frac{1}{\Delta t} \eta_{min}$, then the scheme is unstable.*

Proof. We observe this behavior in our numerical experiment for both first-order and second-order scheme. Here, we give a proof only for the case of first-order scheme.

The timestep precondition implies that there is a point \vec{x} in which $\frac{1}{\Delta t} \eta_{min} \ll \eta(\vec{x}) < \eta_{pres} = \eta_{max}$. In a small region around \vec{x} , we perturb the Gauge variable s^n by a small amount δs^n . The choice $\frac{1}{\Delta t} \eta_{min} \ll \eta(\vec{x})$ allows us to neglect the inertia term in (3.27). The leading terms become $\eta_{pres} \nabla s^n \approx \eta \nabla^2 \mathbf{u}^{n+1}$. Thus, the perturbation δs^n causes a small divergent flow $\delta \mathbf{u}^{n+1}$. This is taken care of by the projection step: $s^{n+1} = s^n - \nabla \cdot \delta \mathbf{u}^{n+1}$.

Thus,

$$\nabla s^{n+1} = \nabla s^n - \nabla(\nabla \cdot \delta \mathbf{u}^{n+1}) = \nabla s^n - \nabla^2 \delta \mathbf{u}^{n+1} - \nabla \times \nabla \times \delta \mathbf{u}^{n+1} \quad (3.47)$$

$$\approx \nabla s^n - \frac{\eta_{pres}}{\eta} \nabla \delta s^n - \nabla \times \nabla \times \delta \mathbf{u}^{n+1} \quad (3.48)$$

$$\approx \nabla s^n - \frac{\eta_{pres}}{\eta} \nabla \delta s^n \quad (3.49)$$

Thus $\|\nabla \delta s\|$ grows by a factor of $\frac{\eta_{pres}}{\eta} > 1$. Therefore, the system is unstable when

$$\frac{1}{\Delta t} \eta_{min} \ll \eta_{pres}. \quad \square$$

Note that the timestep precondition is relevant to biofilm simulations, which usually have $\frac{\eta_{min}}{\eta_{max}} \sim 10^{-4} - 10^{-6}$ while $\Delta t \sim 10^{-3}$. Thus, we must choose η_{pres} that is not too big. Note that [58] uses $\eta_{pres} = \eta_{min}$, which satisfies this stability requirement. We also adopt this value.

Converting the momentum equation into a Helmholtz equation

Solving (3.27) by an iterative method such as BiCG-stab is a slow process. To speed up the computation, we subtract both sides of the equation by $\eta_a \nabla^2 \mathbf{v}$ to get (3.1), which we later formulate as a Helmholtz equation in (3.3). Such equation can be quickly solved using fast Fourier transform, as described in Appendix A. This speed up comes at a cost, as shown in the following analysis.

We consider a simple setting with no Cahn-Hilliard dynamics, EPS growth, nor elastic stress. The only forces in the Navier-Stokes equations are the viscous force and pressure. Consider a 2-D domain where all quantities are constant over x direction but vary in y . That is, $\phi(t, x, y) = \phi(t, y)$. In this setting, any incompressible flow must have the form $\mathbf{v}_x(t, x, y) = \mathbf{v}_x(t, y)$ and $\mathbf{v}_y(x, y) = 0$. We call this a 1-D flow in 2-D domain. It is illustrated in Fig.3.4. In such setting, the advection term is zero $\mathbf{v} \cdot \nabla \mathbf{v} = 0$, and the viscous force $\nabla \cdot (\eta \nabla \mathbf{v})$ has no y component. Thus if we start with zero initial pressure $p^0 = s^0 = 0$, the solution of the momentum equation (3.27) will have no y component,

thus it is divergence-free $\nabla \cdot \mathbf{u}^0(t, y) = 0$. The pressure correction steps (3.28) (3.29) then yield $\psi^1 = p^1 = s^1 = 0$. By induction, we get that $\psi^n = p^n = s^n = 0$ at all time steps. Hence, $\mathbf{v}^n = \mathbf{u}^n$.

For ease of analysis, we now focus on a region where η is constant. This let us simplify the viscous term $\nabla \cdot (\eta \nabla \mathbf{v}) = \eta \nabla^2 \mathbf{v} + (\nabla \eta) \cdot (\nabla \nabla \mathbf{v}) = \eta \nabla^2 \mathbf{v}$. Thus, (3.27) can be rewritten as,

$$\rho \left(\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t} \right) - \eta_a^{n+1} \nabla^2 \mathbf{v}^{n+1} = \eta^n \nabla^2 \mathbf{v}^n - \eta_a^{n+1} \nabla^2 \mathbf{v}^n \quad \text{1st order} \quad (3.50)$$

$$\rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a^{n+1} \nabla^2 \mathbf{v}^{n+1} = \overline{\eta \nabla^2 \mathbf{v}^{n+1}} - \eta_a^{n+1} \overline{\nabla^2 \mathbf{v}^{n+1}} \quad \text{2nd order} \quad (3.51)$$

$$\mathbf{v}^{n+1}|_{y=0} = 0, \quad \mathbf{v}^{n+1}|_{y=L_y} = \mathbf{v}_{shear}. \quad (3.52)$$

We want to use the 2nd order scheme since it has a lower truncation error. However, this scheme turns out to be unstable. To find out the cause, we analyze (3.50), (3.51), and related equations in the absence of shear, $\mathbf{v}_{shear} = 0$. Since \mathbf{v} has no y component, there is no advection in y direction. Thus the EPS concentration is constant over time $\phi(t, y) = \phi(y)$. Therefore, η and η_a are also constant over time. We first analyze the **2nd order scheme**, starting from (3.51) and then Taylor expands \mathbf{v} temporally at step n ,

$$\rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \nabla^2 \mathbf{v}^{n+1} = \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) - \eta_a \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) \quad (3.53)$$

$$\rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \Delta t^2 \nabla^2 \left(\frac{\mathbf{v}^{n+1} - 2\mathbf{v}^n + \mathbf{v}^{n-1}}{\Delta t^2} \right) = \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) \quad (3.54)$$

$$\rho (\mathbf{v}_t + \Delta t \mathbf{v}_{tt}) - \eta_a \Delta t^2 \nabla^2 \mathbf{v}_{tt} = \eta \nabla^2 (\mathbf{v} + \Delta t \mathbf{v}_t + \frac{\Delta t^2}{2} \mathbf{v}_{tt}) + O(\Delta t^2) \quad (3.55)$$

Although one normally drops the Δt^2 in such analysis, we cannot do that here. While

growing biofilms, the values of these parameters are $\rho = 1, \eta_a = 10^8, \eta_s = 10^3, \Delta t = 10^{-3}$. Thus the term $\eta_a \Delta t^2$ is even bigger than the $O(1)$ term! We perform a stability analysis with the ansatz $\mathbf{v} = e^{iky+\lambda t}$,

$$\rho(\lambda + \Delta t \lambda^2) - \eta_a \Delta t^2 (-k^2) \lambda^2 = \eta(-k^2) \left(1 + \Delta t \lambda + \frac{\Delta t^2}{2} \lambda^2\right) \quad (3.56)$$

$$\left(\eta_a \Delta t^2 + \eta \frac{\Delta t^2}{2} + \frac{\rho}{k^2} \Delta t\right) \lambda^2 + \left(\eta \Delta t + \frac{\rho}{k^2}\right) \lambda + \eta = 0 \quad (3.57)$$

In the solution region, for $k = 1$ we have $\lambda = -0.010 \pm 3.16i$. For $k = 100$ we have, $\lambda = -0.005 \pm 3.16i$. Since $|\operatorname{re} \lambda| \ll |\operatorname{im} \lambda|$, this system decays slowly and shows oscillatory behavior. As mentioned earlier, such system is numerically unstable. It often causes the simulation to blow up. Note that such behavior is a numerical artifact from having the timestep Δt too large. In the limit $\Delta t \rightarrow 0$, the system yields a quick decay for all wave number $\lambda = -10^3 k^2$. However, getting into this stable region requires an impractically small timestep.

We next analyze the **1st order scheme**. Start from (3.50) and then Taylor expands \mathbf{v} temporally at step $n + \frac{1}{2}$,

$$\rho \left(\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t}\right) - \eta_a \Delta t \nabla^2 \left(\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t}\right) = \eta \nabla^2 \mathbf{v}^n \quad (3.58)$$

$$\rho \mathbf{v}_t - \eta_a \Delta t \nabla^2 \left(\mathbf{v}_t + \frac{\Delta t^2}{24} \mathbf{v}_{ttt}\right) = \eta \nabla^2 \mathbf{v} - \frac{\Delta t}{2} \eta \nabla^2 \mathbf{v}_t + O(\Delta t^2) \quad (3.59)$$

$$\left(\rho \mathbf{v}_t - \Delta t \eta_a \nabla^2 \mathbf{v}_t\right) - \frac{\eta_a \Delta t^3}{24} \nabla^2 \mathbf{v}_{ttt} = \eta \nabla^2 \mathbf{v} - \frac{\Delta t}{2} \eta \nabla^2 \mathbf{v}_t + O(\Delta t^2) \quad (3.60)$$

We again perform the stability analysis with the ansatz $\mathbf{v} = e^{iky+\lambda t}$. If we ignore the $\eta_a \Delta t^3$ term, we get,

$$\lambda \left(\rho - \eta_a \Delta t (-k^2)\right) = \eta(-k^2) - \frac{\Delta t}{2} \lambda \eta(-k^2) \quad (3.61)$$

$$\lambda \left(\rho + \eta_a \Delta t k^2 - \frac{\Delta t}{2} \eta k^2\right) = -\eta k^2 \quad (3.62)$$

$$\lambda \left(\frac{\rho}{k^2} + \Delta t \left(\eta_a - \frac{\eta}{2}\right)\right) = -\eta \quad (3.63)$$

$$\lambda = -\frac{\eta}{\frac{\rho}{k^2} + \Delta t \left(\eta_a - \frac{\eta}{2}\right)} \approx -10^{-2} \text{ in the solution region} \quad (3.64)$$

Thus, the velocity dissipates as slowly as in the 2nd order scheme. The good news is that λ no longer have the complex part. Thus, here we do not have the oscillatory behavior that destabilized our earlier scheme.

Despite its stability, the very slow decay rate poses a nontrivial problem. Compare to the true decay rate $\lambda = -\frac{\eta k^2}{\rho} = -10^3 k^2$, we are off by a factor of $10^5 k^2$. Its effect is easily noticable when the shear rate \mathbf{v}_{shear} suddenly changes. During a sudden onset of shear, this slow decay causes too much advection inside the EPS, creating unnaturally high strain and hence a very large elastic stress. During a sudden stop of the shear, this slow decay rate causes an unnatural flow reversal in the purely viscous model.

If we keep the $\eta_a \Delta t^3$ term in (3.60), we get,

$$\lambda \left(\rho + \eta_a \Delta t k^2 - \frac{\Delta t}{2} \eta k^2 \right) + \lambda^3 \frac{\eta_a \Delta t^3 k^2}{24} = -\eta k^2 \quad (3.65)$$

$$\lambda \left(\frac{\rho}{k^2} + \Delta t \left(\eta_a - \frac{\eta}{2} \right) \right) + \lambda^3 \frac{\eta_a \Delta t^3}{24} = -\eta \quad (3.66)$$

This yields $\lambda \approx -10^{-2}$, $0.005 \pm 4900i$. The complex roots deserve some comment. They have a positive real part, which normally indicates instability. However, their temporal frequency is much higher than our nyquist rate ($4900 \Delta t \gg 1$). Thus their modes don't directly show up in our simulation. Thus our simulation is still stable.

Since both 1st and 2nd order scheme yield unsatisfactory results, we search for a better scheme. We try a **3rd order scheme** interpolation of the $\eta_a \overline{\nabla^2 \mathbf{v}^{n+1}}$ term. We analyze this scheme by Taylor expanding \mathbf{v} at step $n - \frac{1}{2}$,

$$\rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \nabla^2 \mathbf{v}^{n+1} = \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) - \eta_a \nabla^2 (3\mathbf{v}^n - 3\mathbf{v}^{n-1} + \mathbf{v}^{n-2}) \quad (3.67)$$

$$\rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \Delta t^3 \nabla^2 \left(\frac{\mathbf{v}^{n+1} - 3\mathbf{v}^n + 3\mathbf{v}^{n-1} - \mathbf{v}^{n-2}}{\Delta t^3} \right) = \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) \quad (3.68)$$

$$\rho \left(\mathbf{v}_t + \frac{3}{2} \Delta t \mathbf{v}_{tt} \right) - \eta_a \Delta t^3 \nabla^2 \mathbf{v}_{ttt} = \eta \nabla^2 \left(\mathbf{v} + \frac{3}{2} \Delta t \mathbf{v}_t + \frac{\Delta t^2}{8} \mathbf{v}_{tt} \right) + O(\Delta t^2) \quad (3.69)$$

Stability analysis with the ansatz $\mathbf{v} = e^{iky+\lambda t}$ yields,

$$\rho\left(\lambda + \frac{3}{2}\Delta t\lambda^2\right) - \eta_a\Delta t^3(-k^2)\lambda^3 = \eta(-k^2)\left(1 + \frac{3}{2}\Delta t\lambda + \frac{\Delta t^2}{8}\lambda^2\right) \quad (3.70)$$

$$\left(\eta_a\Delta t^3\right)\lambda^3 + \left(\eta\frac{\Delta t^2}{8} + \frac{3}{2}\frac{\rho}{k^2}\Delta t\right)\lambda^2 + \left(\frac{3}{2}\eta\Delta t + \frac{\rho}{k^2}\right)\lambda + \eta = 0 \quad (3.71)$$

In the solution region, for $k = 1$ we have $\lambda = -21.16, 10.57 \pm 18.99i$. For $k = 100$ we have, $\lambda = -21.31, 10.66 \pm 18.86i$. This is unstable since some λ has a positive real part.

The difficulty we face in 1st and 2nd order scheme has a similar flavor to the behavior of the successive over-relaxation (SOR) method. In SOR, there is an adjustable parameter ω . The system typically converges slowly at $\omega = 1$ and oscillates unstably at $\omega = 2$. In that scheme, the user picks a value $\omega \in [1, 2)$ in order to speed up convergence. This inspires us to try an **interpolation to step** $n + 1 - \epsilon$, for some $\epsilon > 0$. We replace $\eta_a\overline{\nabla^2\mathbf{v}^{n+1}}$ in (3.51) by $\eta_a\overline{\nabla^2\mathbf{v}^{n+1-\epsilon}} := \eta_a\overline{\nabla^2(\mathbf{v}^n + (1 - \epsilon)(\mathbf{v}^n - \mathbf{v}^{n-1}))} + O(\Delta t^2)$. We analyze this scheme by Taylor expanding \mathbf{v} at step n ,

$$\begin{aligned} \rho\left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t}\right) - \eta_a\overline{\nabla^2\mathbf{v}^{n+1}} \\ = \eta\overline{\nabla^2(2\mathbf{v}^n - \mathbf{v}^{n-1})} - \eta_a\overline{\nabla^2((2 - \epsilon)\mathbf{v}^n - (1 - \epsilon)\mathbf{v}^{n-1})} \end{aligned} \quad (3.72)$$

$$\begin{aligned} \rho\left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t}\right) - \eta_a\Delta t^2\overline{\nabla^2\left(\frac{\mathbf{v}^{n+1} - 2\mathbf{v}^n + \mathbf{v}^{n-1}}{\Delta t^2}\right)} \\ = \eta\overline{\nabla^2(2\mathbf{v}^n - \mathbf{v}^{n-1})} + \epsilon\eta_a\overline{\nabla^2(\mathbf{v}^n - \mathbf{v}^{n-1})} \end{aligned} \quad (3.73)$$

$$\begin{aligned} \rho(\mathbf{v}_t + \Delta t\mathbf{v}_{tt}) - \eta_a\Delta t^2\overline{\nabla^2\mathbf{v}_{tt}} \\ = \eta\overline{\nabla^2(\mathbf{v} + \Delta t\mathbf{v}_t + \frac{\Delta t^2}{2}\mathbf{v}_{tt})} + \epsilon\eta_a\overline{\nabla^2(\Delta t\mathbf{v}_t - \frac{\Delta t^2}{2}\mathbf{v}_{tt} + \frac{\Delta t^3}{6}\mathbf{v}_{ttt})} + O(\Delta t^2) \end{aligned} \quad (3.74)$$

As before, we keep the terms $\eta_a\Delta t^3$ and $\eta\Delta t^2$ since they are much bigger than Δt^2 . Plug

ϵ	λ for $\eta_a = 10^8$			ϵ	λ for $\eta_a = 1.5 \times 10^6$		
.1	-5.7e4	-105	-0.10	.2	-2.7e4	-221	-3.4
.01	-6.0e5	-8.9	-1.12	.1	-5.7e4	-99	-7.1
.007	-8.5e5	-5.06	-1.98	.07	-8.3e4	-62	-11.1
.005	-1.2e6	-2.5 ± 1.92i		.05	-1.2e5	-26.0 ± 2.95i	
.003	-2.0e6	-1.5 ± 2.78i		.03	-2.0e5	-15.6 ± 20.84i	
.001	-6.0e6	-.51 ± 3.12i		.01	-6.0e5	-5.36 ± 25.32i	
.0001	-6.0e7	-.055 ± 3.16i		.001	-6.0e6	-0.83 ± 25.81i	

Table 3.1: Decay rate of the $n + 1 - \epsilon$ interpolation scheme

in the ansatz $\mathbf{v} = e^{iky+\lambda t}$, we get,

$$\begin{aligned}
& \rho(\lambda + \Delta t \lambda^2) - \eta_a \Delta t^2 (-k^2) \lambda^2 \\
&= \eta(-k^2)(1 + \Delta t \lambda + \frac{\Delta t^2}{2} \lambda^2) + \epsilon \eta_a (-k^2) (\Delta t \lambda - \frac{\Delta t^2}{2} \lambda^2 + \frac{\Delta t^3}{6} \lambda^3) \quad (3.75) \\
& \epsilon \eta_a \frac{\Delta t^3}{6} \lambda^3 + \left((1 - \frac{\epsilon}{2}) \eta_a \Delta t^2 + \eta \frac{\Delta t^2}{2} + \frac{\rho}{k^2} \Delta t \right) \lambda^2 + \left((\epsilon \eta_a + \eta) \Delta t + \frac{\rho}{k^2} \right) \lambda + \eta = 0 \quad (3.76)
\end{aligned}$$

For the parameters we use, the decay rate λ doesn't depend much on the wave number k . We look for an ϵ that will give us a favorable decay rate. Table.3.1 show the result of a parameter study. For a hard biofilm $\eta_a = 10^8$ the optimal value of ϵ is roughly 0.006. This brings the decay rate to an acceptable value $\lambda = -10^6, -3.02 \pm +0.96i$. This is a factor of 300 improvement over both the 1st and 2nd order scheme. Note that the optimal value of ϵ depends on parameter η_a . For a softer biofilm $\eta_a = 1.5 \times 10^6$, the optimal value of ϵ is roughly $\epsilon = .05$.

The previous method is essentially a 1st order scheme with one free parameter. This inspires us to try a **2nd order sheme with one degree of freedom**. Let $\eta_a \overline{\nabla^2 \mathbf{v}^{n+1}} = \eta_a \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1} + c(\mathbf{v}^n - 2\mathbf{v}^{n-1} + \mathbf{v}^{n-2}))$. This interpolation is 3rd order when $c = 1$ and second order for any c such that $|c| \ll \frac{1}{\Delta t}$. We Taylor expands \mathbf{v} at step n .

$$\begin{aligned}
& \rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \nabla^2 \mathbf{v}^{n+1} \\
&= \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) - \eta_a \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) - c \eta_a \nabla^2 (\mathbf{v}^n - 2\mathbf{v}^{n-1} + \mathbf{v}^{n-2}) \quad (3.77)
\end{aligned}$$

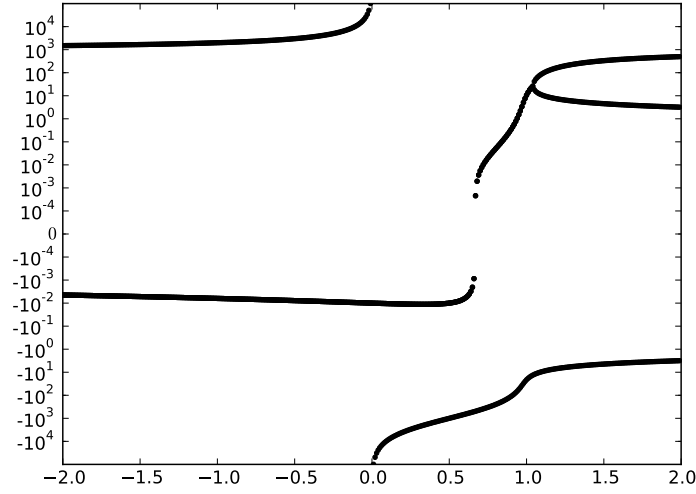


Figure 3.2: Solution of Eq.(3.81). The real part of each λ is plotted as a function of c .

$$\begin{aligned} \rho \left(\frac{3\mathbf{v}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) - \eta_a \Delta t^2 \nabla^2 \left(\frac{\mathbf{v}^{n+1} - 2\mathbf{v}^n + \mathbf{v}^{n-1}}{\Delta t^2} \right) \\ = \eta \nabla^2 (2\mathbf{v}^n - \mathbf{v}^{n-1}) - c\eta_a \Delta t^2 \nabla^2 \left(\frac{\mathbf{v}^n - 2\mathbf{v}^{n-1} + \mathbf{v}^{n-2}}{\Delta t^2} \right) \end{aligned} \quad (3.78)$$

$$\begin{aligned} \rho(\mathbf{v}_t + \Delta t \mathbf{v}_{tt}) - \eta_a \Delta t^2 \nabla^2 \mathbf{v}_{tt} + c\eta_a \Delta t^2 \nabla^2 (\mathbf{v}_{tt} - \Delta t \mathbf{v}_{ttt}) \\ = \eta \nabla^2 (\mathbf{v} + \Delta t \mathbf{v}_t + \frac{\Delta t^2}{2} \mathbf{v}_{tt}) + O(\Delta t^2) \end{aligned} \quad (3.79)$$

With the ansatz $\mathbf{v} = e^{iky + \lambda t}$, we get,

$$\rho(\lambda + \Delta t \lambda^2) - (1 - c)\eta_a \Delta t^2 (-k^2) \lambda^2 - c\eta_a \Delta t^3 (-k^2) \lambda^3 = \eta(-k^2) \left(1 + \Delta t \lambda + \frac{\Delta t^2}{2} \lambda^2 \right) \quad (3.80)$$

$$c\eta_a \Delta t^3 \lambda^3 + \left((1 - c)\eta_a \Delta t^2 + \eta \frac{\Delta t^2}{2} + \frac{\rho}{k^2} \Delta t \right) \lambda^2 + \left(\eta \Delta t + \frac{\rho}{k^2} \right) \lambda + \eta = 0 \quad (3.81)$$

Fig.3.2 shows the real part of λ as a function of c . Notice that, regardless of value of c , there is always a root λ with $\text{re } \lambda > -0.01$. The values at $c = 0$ and $c = 1$ agree with the earlier analysis of the 2nd and 3rd order scheme respectively. Notice that the system bifurcates near $c = 0$. This offers another clue on why the 2nd order scheme blows up during the simulation.

In conclusion, the most stable scheme we found involves interpolating the term $\overline{\nabla^2 \mathbf{v}}$ to step $n+1-\epsilon$. Under a dynamic shear, the leading term of the truncation error is $\epsilon \eta_a \Delta t \nabla^2 \mathbf{v}_t$. This term is first order, and is biggest when v_{shear} is not constant over time. The optimal value ϵ depends on the value η_a , and must be chosen with stability in mind. We have shown that blindly minimizing the truncation error can get us into an unstable system at the time step we choose.

Stability and stiffness

Note that the convergence issue we encountered originates from the fact that $\eta_a \gg \eta$. Thus, it makes sense to try setting η_a to be lower. This would also help reduce the truncation error, which is proportional to η_a . However, when we lowered η_a , we found that the simulations quickly became unstable. The following lemma gives an explanation.

Lemma 3.2.3. *Stability requires that $\eta_a > \frac{1}{2}\eta_{max}$.*

Proof. We perform a stability analysis of the momentum equation (3.3). Consider a region where the viscosity is constant, thus the viscous force can be simplified to $\eta \nabla^2 \mathbf{v}$. Perturb the stoke solution by a small velocity field $\delta \mathbf{u}$. We can choose $\delta \mathbf{u}$ to be divergence-free, thus the pressure force remains the same.

We start by analyzing the purely first order scheme. For the timescale we are interested in, the inertia term is much smaller than the viscous term, $\frac{1}{\Delta t} \ll \eta_a$. Thus the leading terms of the perturbation to the momentum equation become,

$$-\eta_a \nabla^2 \delta \mathbf{u}^{n+1} \approx (\eta - \eta_a) \nabla^2 \delta \mathbf{u}^n \quad (3.82)$$

On each iteration, the perturbation is magnified by $\frac{\eta - \eta_a}{-\eta_a} = 1 - \frac{\eta}{\eta_a}$. For stability inside the EPS network, where $\eta = \eta_{max}$, we need $\left|1 - \frac{\eta_{max}}{\eta_a}\right| < 1$. That is, $\eta_a > \frac{1}{2}\eta_{max}$. This is confirmed in our numerical simulation.

Next we analyze the $n + 1 - \epsilon$ interpolation scheme. With the ansatz $\delta \mathbf{u}^n = c^n \delta \mathbf{u}^0$, the leading terms becomes,

$$-\eta_a \nabla^2 \delta \mathbf{u}^{n+1} \approx (\eta - \eta_a) \nabla^2 \overline{\delta \mathbf{u}}^{n+1-\epsilon} \quad (3.83)$$

$$c^{n+1} \nabla^2 \delta \mathbf{u}^0 \approx \left(1 - \frac{\eta_{max}}{\eta_a}\right) (c^n + (1 - \epsilon)(c^n - c^{n-1})) \nabla^2 \delta \mathbf{u}^0 \quad (3.84)$$

$$c^2 \approx \left(1 - \frac{\eta_{max}}{\eta_a}\right) (2c - 1 - \epsilon c + \epsilon) \quad (3.85)$$

$$c^2 - \left(1 - \frac{\eta_{max}}{\eta_a}\right) (2 - \epsilon)c + \left(1 - \frac{\eta_{max}}{\eta_a}\right) (1 - \epsilon) \approx 0 \quad (3.86)$$

We want $|c| < 1$. For small ϵ , this requirement translates to $\eta_a > \frac{5}{9} \eta_{max}$. This improves the constraint on η_a slightly (about $\frac{1}{9} \eta_{max}$.) \square

Stability of the system is also determined by the magnitude of explicit terms comparing to the magnitude of implicit terms. The largest terms in the momentum equation are pressure, elastic force, viscous force. The pressure term only reacts to other forces through the divergence in \mathbf{u} . In the first order Gauge-Uzawa scheme, the pressure-related term does not cause any instability. On the other hand, both the elastic force and viscous force are large, and handled explicitly. They are the main contributors to the equation's stiffness.

Because of the big difference in viscosity inside our system, the viscous force term $\nabla \cdot (2\eta \mathbf{D})$ requires special care. In the solution, both η and $2\mathbf{D}$ are of order 1. Inside the biofilm, $\eta \sim \eta_{max} \sim 10^5$ and $2\mathbf{D} \sim 10^{-5}$. Thus $\nabla \cdot (2\eta \mathbf{D})$ seems to be of benign magnitudes. However, the discrete version of this operator can have a problem on the biofilm-solution interface.

For example, say our interface width is $m \approx 5$ grids. On the edge of the EPS-solution interface, there are two adjacent cells where one has the volume fraction $\phi_n \approx 0$ and the other has $\phi_n \approx \frac{1}{m} \phi_{max}$. The viscosities in these cells are $\eta \approx \eta_{sol}$ and $\eta \approx \frac{1}{m} \eta_{max}$ respectively. Eq.(3.92) gives their approximate shear rate at $\dot{\gamma} \approx 1$ and $\dot{\gamma} \sim \frac{m}{\eta_{max}}$ respectively. During the discretization process, we occasionally have to average the values of two adjacent cells to get a value at half-cell locations. For the two aforementioned cells, their

average rate-of-strain tensor has the magnitude of order $\frac{1}{2}$. Thus the viscous stress force $\nabla \cdot (2\eta\mathbf{D}^n) \sim \frac{1}{m\Delta x}\eta_{max}$ can be of order 10^6 .

Since this force is handled explicitly, we have a very stiff equation. Fortunately, the momentum equation has the implicit term $\eta_a \nabla^2 \mathbf{u}^{n+1}$ that we introduced during the Helmholtz-ification process. If η_a is big enough, this term will dominate, and allow the momentum equation to accept a relatively large force without blowing up. In the purely-viscous model, we have,

$$\eta_a = \eta_{ps}\phi_{max}^n + \eta_s(1 - \phi_{max}^n). \quad (3.87)$$

The elastic force is also large and explicit, thus adding to the stiffness of the momentum equation. In order to keep the equation stable, we increase η_a further,

$$\eta_a = a\eta_n\phi_{max}^n + \eta_{ps}\phi_{max}^n + \eta_s(1 - \phi_{max}^n). \quad (3.88)$$

One way to alleviate the equation's stiffness is to handle the viscous stress implicitly. This can be done by augmenting an **iterative scheme** to our solver. Let $\mathbf{u}^{n+1,0} = \bar{\mathbf{v}}^{n+1-\epsilon}$, then for $i = 0, 1, 2, \dots$, we solve for $\mathbf{u}^{n+1,i+1}$ in,

$$\begin{aligned} \rho \left(\frac{3\mathbf{u}^{n+1,i+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) + \rho \bar{\mathbf{v}}^{n+1} \cdot \nabla \mathbf{u}^{n+1,i} + \eta_{pres} (\nabla S^n) - \eta_a \nabla^2 \mathbf{u}^{n+1,i+1} \\ = \bar{\mathbf{F}}_{CH}^{n+1} + \bar{\mathbf{F}}_{elastic}^{n+1} + \nabla \cdot (\eta \mathbf{D}^{n+1,i}) - \eta_a \nabla^2 \mathbf{u}^{n+1,i}. \end{aligned} \quad (3.89)$$

Once $\mathbf{u}^{n+1,i+1}$ converges, say, to \mathbf{u}^{n+1} , we have

$$\begin{aligned} \rho \left(\frac{3\mathbf{u}^{n+1} - 4\mathbf{v}^n + \mathbf{v}^{n-1}}{2\Delta t} \right) + \rho \bar{\mathbf{v}}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \eta_{pres} (\nabla S^n) \\ = \bar{\mathbf{F}}_{CH}^{n+1} + \bar{\mathbf{F}}_{elastic}^{n+1} + \nabla \cdot (\eta \mathbf{D}^{n+1}). \end{aligned} \quad (3.90)$$

Hence the viscous force is taken care of implicitly.

In our experience, this implicit scheme seems to work fine in purely viscous simulations. It needs several iterations when \mathbf{v}_{shear} changes value, and needs only one iteration otherwise. The drawback is that we no longer have the $\eta_a \nabla^2 \mathbf{v}^{n+1}$ term that helps absorb

the equation's stiffness. Such term is very useful for the viscoelastic simulation, where $\bar{\mathbf{F}}_{elastic}^{n+1}$ can be very big. Thus, we opt not to use this implicit scheme in the viscoelastic model.

Boundary condition and ghost cell values

One pitfall in implementing the Gauge-Uzawa scheme is to assume that the flow field \mathbf{v}^n has a Dirichlet boundary condition. It doesn't! Only \mathbf{u}^n satisfies the a zero dirichlet boundary at the wall $y = 0$. The projection step yields $\mathbf{v}^n = \mathbf{u}^n + \frac{1}{\rho^n} \nabla \psi^n$. Recall that ψ has the zero Neumann boundary condition. Thus $\mathbf{v}_{y=0}^n = \frac{1}{\rho^n} \nabla \psi^n$, which is zero in y direction, but nonzero in x and z directions. It is a mistake to compute the ghost cell value of \mathbf{v} using the assumption $\mathbf{v}_{y=0} = 0$. The correct value is $\mathbf{v}_{y=0}^n = \frac{1}{\rho^n} \nabla \psi_{y=0}^n$.

Such mistake can easily go unnoticed since $\nabla \psi_{y=0}^n$ is usually small. However, our numerical scheme for the momentum equation has the added term $\eta_a \nabla^2 \mathbf{v}$. Chapter 3.2 shows that this term makes any error in $\nabla^2 \mathbf{v}$ dissipates slowly, especially so in the first order scheme. This allows small errors to accumulate across thousands of steps into something noticeably, as shown in fig.3.3.

Such mistake can also be rendered harmless if one solves the momentum equation fully explicitly. In that case, the error due to this mistake would dissipate in only one timestep. Prior works on this biofilm model [76] [75] use viscous model on a regular grid, which doesn't need the ghost cell value for \mathbf{v} , thus most likely did not run into this issue.

1-D analytical solution

One can readily find the analytical solution to the 1-D purely-viscous flow in a 2-D domain. Though simple, the result that we obtain gives us some intuition about the flow in the system. In this setting, all quantities are constant over x direction but vary in y . That is, $\phi(t, x, y) = \phi(t, y)$. The flow is $\mathbf{v}_x(t, x, y) = \mathbf{v}_x(t, y)$ and $\mathbf{v}_y(x, y) = 0$. Thus, $\mathbf{v} \cdot \nabla \mathbf{v} = 0$. Without Cahn Hilliard dynamics, the dynamic equilibrium (i.e. the stoke solution) has all

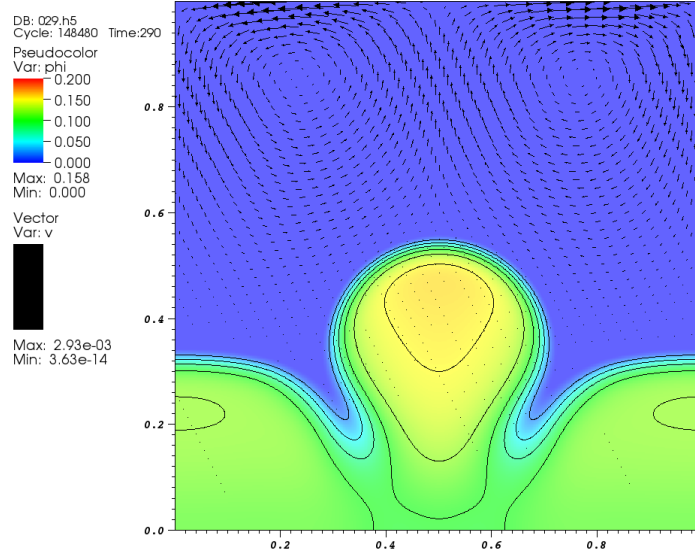


Figure 3.3: The correct boundary condition for the Gauge-Uzawa scheme is $\mathbf{u}_{y=1} = 0$. The incorrect assumption that $\mathbf{v}_{y=1} = 0$ can introduce a boundary error of order 10^{-7} on each time step. Using the first order scheme (3.50), such error accumulates over 10^5 steps into the noticeable spurious flow of magnitude 3×10^{-3} .

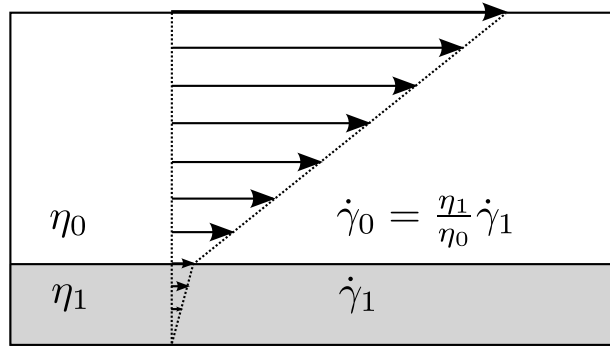


Figure 3.4: Shear profile for 1-D flow in a 2-D domain.

terms in the momentum equation (3.3) equal zero. The velocity profile is determined by

$$\nabla \cdot (2\eta\mathbf{D}) = 0,$$

$$\nabla \cdot \left(\eta(y) \begin{bmatrix} 0 & \dot{\gamma}(y) \\ \dot{\gamma}(y) & 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (3.91)$$

where $\dot{\gamma} = \frac{\partial}{\partial y}v_x(y)$ is the shear rate. This yields $\eta(y)\dot{\gamma}(y) = \text{const.}$

Nontrivial EPS profile has a more complicated flow. However, up to an order of mag-

nitude, it is still true that

$$\dot{\gamma} \propto 1/\eta. \quad (3.92)$$

This can be used to partially check the correctness of the numerical simulation.

3.3 ELASTIC STRESS EQUATION

The elastic stress in our model originates from the EPS, which exists only in the biomass. Thus $\tau_n = \phi_n \tilde{\tau}_n$, where $\tilde{\tau}_n$ is the elastic stress per biomass volume fraction. In the region with no biomass, $\tilde{\tau}_n$ is a ghost quantity which we compute but do not use.

Analytically, it does not matter whether we choose to keep track of either the quantity τ_n or $\tilde{\tau}_n$. Numerically, however, it is better to keep track of τ_n . The reason is, when the system is being sheared, the magnitude of $\tilde{\tau}_n$ is very high in regions where ϕ_n is very low, and vice versa. Thus there is a big loss in numerical accuracy of the product $\phi_n \tilde{\tau}_n$ when ϕ_n and $\tilde{\tau}_n$ are advected separately.

Due to advection and the Cahn-Hilliard dynamics, the value of ϕ_n outside the biofilm is usually not exactly zero. As will be discussed in Sec.3.4, it is common to see a faint level of biomass $\phi_n \approx 10^{-5}$ in the solution region. Under shear, the EPS in these faint biomass experiences a big strain rate. The damping term in the Giesekus model keeps the strain in such regions from getting too large.

As remarked in Sec.2.3, the Giesekus constitutive equation for τ_n has a singularity at $\phi_n = 0$. We circumvent this by introducing a numerical parameter ϵ_ϕ in Sec.2.3, which generates a small amount of arbitrariness to our model. One alternative is to use the **Phan-Thien-Tanner** model, which is similar to Giesekus model (2.10), but the damping term $\tau_n \cdot \tau_n$ is replaced by $tr(\tau_n)\tau_n$. Instead of (2.22), we have,

$$\begin{aligned} \frac{\partial \tau_n}{\partial t} + \mathbf{v}_n \cdot \nabla \tau_n - \mathbf{W}_n \cdot \tau_n + \tau_n \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_n + \tau_n \cdot \mathbf{D}_n) \\ + \frac{\alpha}{\phi_n \eta_m} tr(\tau_n) \tau_n + \frac{\tau_n}{\Lambda_1} = \tilde{g}_n \tau_n + \frac{2\phi_n \eta_m}{\Lambda_1} \mathbf{D}. \end{aligned} \quad (3.93)$$

The advantage of the $tr(\tau_n)\tau_n$ form is that we can easily handle the singularity at ϕ_n . After the substitution $\tau_p = \tau_n + Bi\phi_n\mathbf{I}$ with $Bi = \frac{\eta_n}{a\Lambda_1}$, instead of Eq. (2.30), we have

$$\begin{aligned} \frac{\partial \tau_p}{\partial t} + \mathbf{v}_n \cdot (\nabla \tau_p) - \mathbf{W}_n \cdot \tau_p + \tau_p \cdot \mathbf{W}_n - a(\mathbf{D}_n \cdot \tau_p + \tau_p \cdot \mathbf{D}_n) \\ + \left(\frac{\alpha}{\phi_n \eta_n} tr(\tau_p - Bi\phi_n\mathbf{I}) + \frac{1}{\Lambda_1} \right) (\tau_p - Bi\phi_n\mathbf{I}) = \tilde{g}_n \tau_p. \end{aligned} \quad (3.94)$$

This can be solved by the same numerical scheme we presented in Sec.3.1, but with no explicit damping term, and with the effective relaxation time $\tilde{\Lambda}_1(\vec{x}, t)$ defined as the half the harmonic mean of Λ_1 and $\frac{\phi_n \eta_n}{\alpha tr(\tau_p - Bi\phi_n\mathbf{I})}$. The benefit of this method is that, when ϕ_n is small, the damping term now shows up as a decaying factor, which is easy to handle and can be computed faithfully. Contrast this to the damping term in the Giesekus model which requires a numerical approximation in order to avoid the singularity at $\phi_n = 0$. One drawback is that there is a study [22] that shows the Phan-Thien-Tanner model to be less stable than the Giesekus model.

In our numerical simulations, the Phan-Thien-Tanner model yields more or less the same results as the Giesekus model. The two models differ only in the damping term. The key contribution of this term to the biofilm model is to damp the stress outside the EPS to zero. It has only a small affect on the stress inside the main biomass, whose strain is relatively small. Thus, both models yield a similar elastic stress inside the biomass. In this work, we decide to focus on the Giesekus model.

3.4 BIOMASS VOLUME FRACTION EQUATION

In this section, we discuss some basic properties of the biomass volume fraction equation (ϕ_n), and issues we encounter while implementing it. Recall the mixing free energy density,

$$f = \frac{\Gamma_1}{2} |\nabla \phi_n|^2 + \Gamma_2 \left[\frac{\phi_n}{N} \ln(\phi_n) + (1 - \phi_n) \ln(1 - \phi_n) + \chi \phi_n (1 - \phi_n) \right] \quad (3.95)$$

$$=: \frac{\Gamma_1}{2} |\nabla \phi_n|^2 + \Gamma_2 \tilde{f}(\phi_n). \quad (3.96)$$

The Γ_1 term maintains finite interface width to some extent. The Γ_2 term encourages the biomass to mix with the solution when \tilde{f} is concave up, and to nucleate when \tilde{f} is concave down. Thus we are interested in the second derivative of \tilde{f} ,

$$\tilde{f}(\phi_n) = \frac{\phi_n}{N} \ln(\phi_n) + (1 - \phi_n) \ln(1 - \phi_n) + \chi\phi_n(1 - \phi_n), \quad (3.97)$$

$$\tilde{f}'(\phi_n) = \frac{1}{N} + \frac{\ln \phi_n}{N} - \ln(1 - \phi_n) - 1 + \chi(1 - 2\phi_n), \quad (3.98)$$

$$\tilde{f}''(\phi_n) = \frac{1}{N\phi_n} + \frac{1}{1 - \phi_n} - 2\chi. \quad (3.99)$$

We then find the inflection points,

$$0 = \tilde{f}''(\phi_n) = \frac{1}{N\phi_n} + \frac{1}{1 - \phi_n} - 2\chi \quad (3.100)$$

$$0 = \frac{1 - \phi_n}{N} + \phi_n - 2\chi\phi_n(1 - \phi_n) \quad (3.101)$$

$$0 = 2\chi\phi_n^2 - \left(2\chi - 1 + \frac{1}{N}\right)\phi_n + \frac{1}{N} \quad (3.102)$$

$$0 = \phi_n^2 - \left(1 - \frac{1}{2\chi}\left(1 - \frac{1}{N}\right)\right)\phi_n + \frac{1}{N2\chi} \quad (3.103)$$

$$\phi_n = \frac{1}{2} \left(1 - \frac{1}{2\chi}\left(1 - \frac{1}{N}\right) \pm \sqrt{\left(1 - \frac{1}{2\chi}\left(1 - \frac{1}{N}\right)\right)^2 - \frac{4}{N2\chi}} \right) \quad (3.104)$$

Fig.3.5 plots the derivatives of the bulk free energy for $N = 1000$, $\chi = 0.55$. It shows that the biomass nucleates when $\phi_n \in (0.0113, 0.0805)$, and mixes with the solution otherwise. Fig.3.6 plots the inflection points ϕ_1, ϕ_2 of \tilde{f} as a function of the parameters N and χ . The biomass nucleates when $\phi_n \in (\phi_1, \phi_2)$.

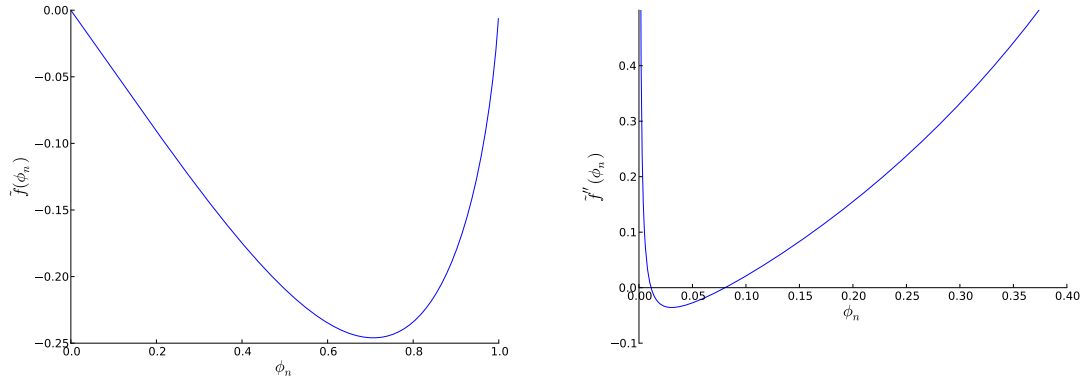
As the biomass migrates, the two quantities of interest are the excessive network velocity $-\lambda \nabla \frac{\delta f}{\delta \phi_n}$ and the corresponding flux $-\lambda \phi_n \nabla \frac{\delta f}{\delta \phi_n}$, where,

$$\nabla \frac{\delta f}{\delta \phi_n} = -\Gamma_1 \nabla (\nabla^2 \phi_n) + \Gamma_2 \nabla \left(\frac{1}{N} \ln(\phi_n + \epsilon) - \ln(1 - \phi_n) - 2\chi\phi_n \right) \quad (3.105)$$

$$= -\Gamma_1 \nabla (\nabla^2 \phi_n) + \Gamma_2 \left(\frac{1}{N} \frac{1}{\phi_n + \epsilon} + \frac{1}{1 - \phi_n} - 2\chi \right) \nabla \phi_n. \quad (3.106)$$

$$\phi_n \nabla \frac{\delta f}{\delta \phi_n} = \phi_n \nabla \left(-\Gamma_1 \nabla^2 \phi_n + \Gamma_2 \left(\frac{1}{N} \ln(\phi_n + \epsilon) - \ln(1 - \phi_n) - 2\chi\phi_n \right) \right) \quad (3.107)$$

$$= -\Gamma_1 \phi_n \nabla (\nabla^2 \phi_n) + \Gamma_2 \left(\frac{\phi_n}{N(\phi_n + \epsilon)} + \frac{\phi_n}{1 - \phi_n} - 2\phi_n \chi \right) \nabla \phi_n. \quad (3.108)$$



$$(a) \tilde{f}(\phi_n) = \frac{\phi_n}{N} \ln(\phi_n) + (1 - \phi_n) \ln(1 - \phi_n) + \chi \phi_n(1 - \phi_n) \quad (b) \tilde{f}''(\phi_n) = \frac{1}{N} \frac{1}{\phi_n} + \frac{1}{1 - \phi_n} - 2\chi$$

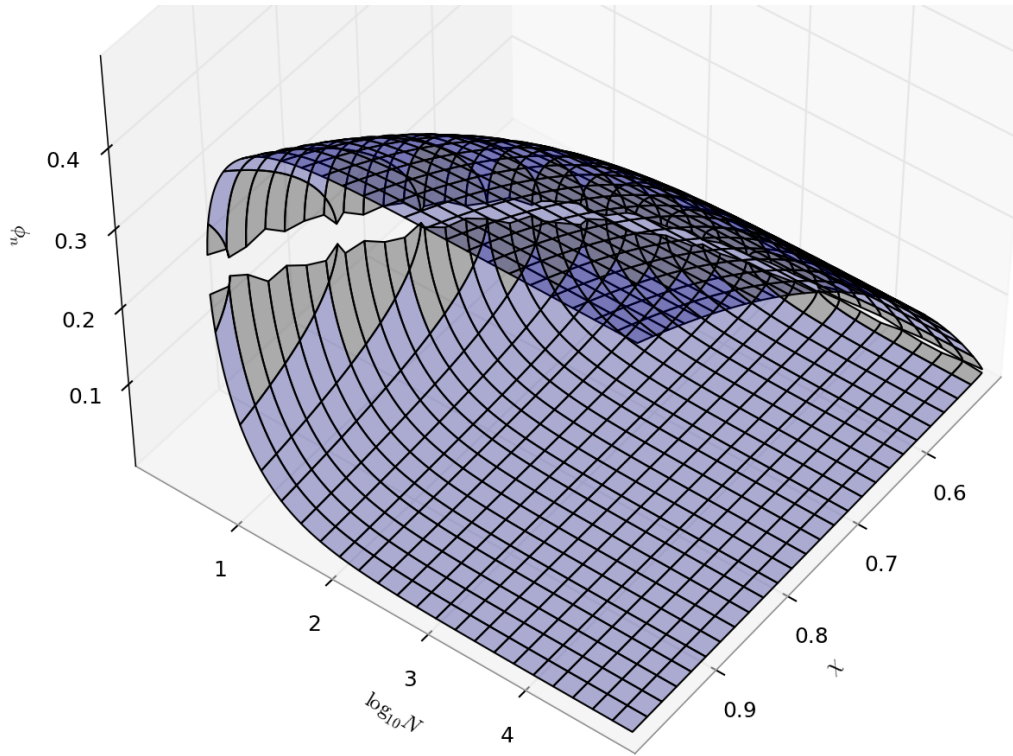
Figure 3.5: The bulk free energy \tilde{f} and its second derivative, plotted with $N = 1000$, $\chi = 0.55$. The biomass nucleates when $\phi_n \in (0.0113, 0.0805)$.

The ϵ is required in (3.105), (3.106), (3.107) to regularize the singularity at $\phi_n = 0$. It is not required in (3.108). The choice of ϵ impacts the network flux only in regions where the biomass is faint, $\phi_n \ll \frac{1}{N}$. The biomass flux inside the main biofilm lobe is virtually independent of ϵ .

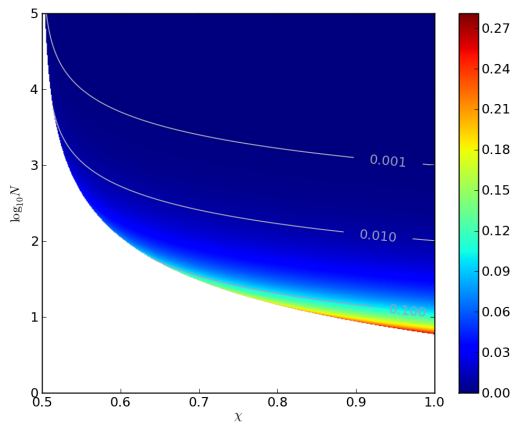
On the staggered grid, evaluating the form (3.108) involves computing the average of two adjacent cells $(\phi_n)_{i+\frac{1}{2}}$, while (3.107) does not need this. On the other hand, (3.108) has the advantage of being readily linearizable, so it can be used in the BiCG-stab solver. The numerical results presented here use the form (3.108), which yields the biomass volume fraction equation (3.6).

By definition, ϕ_n should be nonnegative. Analytically, starting with a nonnegative initial $\phi_n(\mathbf{x}, 0)$, our model maintains the nonnegativity of $\phi_n(\mathbf{x}, t)$ over the whole domain at all time. For numerical computation under a finite grid size and finite time step, there are two main causes of ϕ_n becoming slightly negative.

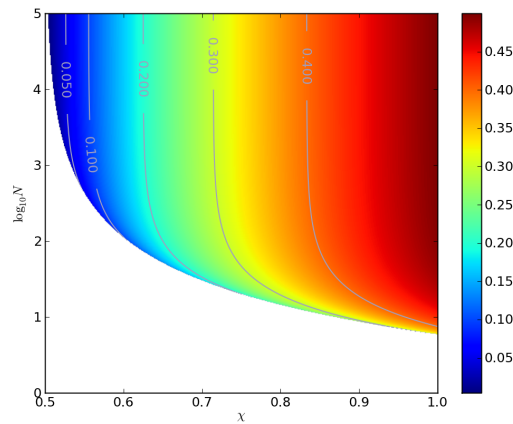
One cause is the advection of the biomass. Even when we start with $\phi_n = 0$ outside the biofilm, any spatially second order finite difference advection schemes will yield ϕ_n that is slightly negative near the interface ([64] p. 73). Since the bulk free energy term requires



(a) In the region bounded by the two surfaces, the bulk free energy \tilde{f} is concave down, thus the biomass nucleates.



(b) First inflection point of $\tilde{f}(\phi_n)$



(c) Second inflection point of $\tilde{f}(\phi_n)$

Figure 3.6: Inflection points ϕ_1, ϕ_2 of the bulk free energy $\tilde{f}(\phi_n)$. The biomass nucleates when $\phi_n \in (\phi_1, \phi_2)$.

that ϕ_n is nonnegative, we enforce this by clipping its values in our implementation: $\phi_n := \max(0, \phi_n)$. However, the clipping introduces a truncation error which effectively increases the total biomass volume slightly. Under a straightforward finite-difference discretization, clipping increases the total biomass volume $\int_{\Omega} \phi(\mathbf{x}, t) d\mathbf{x}$ during a shearing simulation by 0.1 - 1%. We mitigate this issue by advecting ϕ_n with WENO as outlined in Sec.3.1. This reduces the truncation error due to clipping to a negligible level.

Another cause of ϕ_n being negative is the mixing due to the Cahn-Hilliard dynamics. The numerical computation of this process can yield negative ϕ_n near the interface. Through numerical investigations, we found that reducing the time step size does not mitigate the error due to clipping, but refining the spatial grid size solves the problem.

In summary, we use the biomass flux of the form (3.108), which yields the biomass volume fraction equation (3.6). The clipping of ϕ_n results in a slight increase of the biomass, but the problem subsides as the grid size gets finer, and ϕ_n is advected by the WENO scheme.

3.5 GPU IMPLEMENTATION

The numerical scheme is implemented to run on a graphics processing unit (GPU), offering almost a hundred time speed up over a single-thread CPU-only program. Numerical computations are carried out on a GPU, while input/output and bookkeeping are done on the CPU. The GPU code is written in CUDA C, which is a general purpose GPU programming language based on standard C created by the GPU maker Nvidia. The CPU code that performs input/output and parameters handling is written in C++.

Nvidia provides several guides and manuals about GPU programming. Good starting points are [43] [44], which introduce many terms and concepts that we will use in this section. A **thread** is a basic software unit that carries out a series of instructions. These instructions are physically executed on a CPU **core** or a GPU **thread processor** (also called a GPU core). Each thread resides in one core. On the other hand, each core might work on

several threads concurrently. A **multithreaded** software uses several threads to complete its computation. Both CPUs and GPUs run on a clock that ticks on the order of 10^9 cycles per second.

A GPU gains its computational efficiency/speed through massive parallelism. A typical GPU contains 200 - 512 thread processors. In contrast, a typical computational server contains 4 - 8 CPUs, each of which contains 4 - 8 cores. Thus each server typically contains 16 - 64 CPU cores. A simulation can be computed across multiple servers, but network communication often becomes the bottle neck for the computation.

A GPU also gains its speed through its high memory bandwidth, which is typically 100 GB/sec. Multi-threaded CPU codes keep their data in the CPU main memory, which typically has the bandwidth of 10 GB/sec. If the calculation is spanned across multiple servers, the bottle neck is posed by the network speed, which has a typical bandwidth of 0.1 - 1 GB/sec. For straightforward calculations like the finite difference method, memory bandwidth can easily become the bottleneck for the computation. Thus, it is important to keep most data inside the GPU, and minimize the read/write to the CPU main memory.

In both the CPU and GPU, when several computational threads share the same computational core, the core has to divide its time among the threads. This is done through the process called **context switch**, in which the core saves all computational resources belonging to one thread, and then switches to working on another thread. Context switch usually occurs hundreds of times each second. One fundamental difference between CPU and GPU computing is that CPU threads are “heavy” while GPU threads are “light”. A CPU context switch takes thousands of computational cycles, while a GPU context switch incurs almost no computational cost. Therefore, CPU-based multithreaded numerical computations normally create only one thread per CPU core. On the other hand, to fully utilize GPU computational resource, the user is *required* to create several threads per core.

Additionally, CPU threads take thousands of cycles to create, while GPU threads are created at almost no computational cost. Once created, each CPU thread is used for a long

time, often for the entire simulation duration. In contrast, GPU threads are ephemeral. They are continually spawned and terminated. Each lives for only a fraction of a second.

Hardware-wise, GPU cores are grouped into **streaming multiprocessors (SM)**. Software-wise, GPU threads are grouped into **blocks**. Threads that belong to the same block are computed on cores that reside in the same SM. As a consequence, threads in the same block are executed in a **lockstep** fashion, i.e. they execute the same instruction at the same time, and can communicate very fast within the block. Due to speed and resource constraints, a block should contain 32 - 1024 threads. The optimal number of threads per block depends on the exact hardware, the memory access pattern, computational intensity, and the amount of register memory that each thread uses. GPUs use thread parallelism to hide the **memory latency**, i.e. the duration that a thread processor needs to wait for the data to arrive from the memory. Therefore, in order to utilize the full computational speed, each core should be working on several threads concurrently. This is achieved both by having many threads in each block, and by running several blocks concurrently on each SM. Because of this, a high-end GPU needs at least tens of thousands of threads in order to run at the full speed.

Take for example a computation of a $200 \times 200 \times 200$ data points by using 448 CPU cores or 448 GPU cores. On CPUs, this computation could be done on 14 servers, each with 32 CPU cores. One would create one computational thread per CPU core, totaling 448 threads. Each thread would be responsible for roughly 18,000 data points. Each thread would live for the entire length of simulation, passing data to and from other threads as needed. Threads on the same server communicate via the CPU main memory at 10 GB/sec bandwidth and $1/10 \mu\text{s}$ latency. Threads on different servers communicate at 1 GB/sec bandwidth and $10 \mu\text{s}$ latency.

On GPUs, this computation can be done on a single GPU card. The card consists of 14 SM, each containing 32 cores, for the total of 448 cores. One might create 8,000,000 threads, grouped into 62,500 blocks, each of size 128. Each thread is responsible for only

one data point. The blocks are queued to be processed by the SMs. Each SM processes several blocks concurrently. As an SM finishes computing one block, all threads in that block terminate. The SM starts working on another block from the queue. This repeats until all blocks have been processed. Threads are short-lived. They all run the same code called **kernel**. A program would consist of dozens of kernels, each created to accomplish a short specific goal. One launches a kernel with 8,000,000 threads to compute a Laplacian. They would finish this job in a fraction of a second, then terminate. Then another kernel of 8,000,000 threads are launched to take a finite difference. This too would finish in a blink. Then another kernel with 8,000,000 threads is launched to perform the next specific task, and so on. Threads in the same block can communicate via the SM's shared memory at virtually no bandwidth limit, i.e. as fast as they can compute results, and at the latency of about 10-20 clock cycles ($1/50 - 1/25 \mu s$). Threads in different blocks can communicate via the GPU's global memory at the bandwidth of 100 GB/sec and latency of 500 cycles ($1 \mu s$).

Data layout is also an important issue. The memory is divided into consecutive blocks called **cache lines**. Each cache line is 128 bytes on the GPUs we currently use. Under certain conditions, fetching the whole 128-byte of data from the same cache line can take the same amount of time as fetching just one byte. This is called a **coalesced** memory access. The condition in which this happens is listed in the GPU programming manual, and varies from GPU to GPU. In general, we want threads in the same block to access data from the same few cache lines.

Managing data dependency is an important aspect of parallel programming. In GPUs, threads in the same block are executed in lockstep. However, threads in different blocks are executed in non specified order. They might be executed concurrently or at a different time. Thus, there is no communication between blocks. If one block needs a data computed by another block, we must wait for the kernel to terminate, then launch new kernel.

The biofilm code consists of about 30 kernels that take turn running on a single GPU.

All variables are kept in the GPU. They are transferred to the CPU memory only when we want to save the data into a file. A kernel can be as short as two lines, such as the one used for adding two arrays together. On the other hand, the kernel for computing terms in the momentum equation is about 150 lines.

The boundary conditions are handled by the method of ghost cells. There are several kernels whose sole job is to set the ghost cells to the appropriate values based on the imposed boundary conditions. This allows other kernels to treat boundary nodes and internal nodes in the same way, thus simplifying the code structure.

Starting from the domain of size $N_x \times N_y \times N_z$, we pad each size by one data point to hold the ghost cell values. This $(N_x + 2) \times (N_y + 2) \times (N_z + 2)$ data is further padded into $(N_x + 2) \times (N_y + 2) \times (N_z^{padded})$ in order to align with the cache line. This allows a chunk of data to be fetched in a coalesced manner mentioned earlier. The value $N_z^{padded} = 16 \times \lceil (N_z + 2)/16 \rceil$ seems to work fine for the two types of GPUs we often use (Tesla C1060 and C2070).

The grid size, along with other grid-related parameters, are stored in the class `GridParamsBase`. Since Tesla GPUs can perform multiplication quicker than division, we also store the value $idx = 1/\Delta x$. Any division by Δx can then be replaced by a multiplication by idx . Oft-used quantities such as $1/(\Delta x)^2$ are also stored so that they don't need to be computed repeatedly. Similarly, values of nondimensionalized parameters are stored in their own class. These parameters are kept in the GPU's constant memory, denoted by the keyword `__constant__`. The **constant memory** is a small but very fast part of the GPU memory. It can be set only by the CPU, not by the GPU. It can hold only 64 KB, and only 8 KB is cached by each SM at any time.

```

1 class GridParamsBase {
2 public:
3     int Nx , Ny , Nz;
4     int Nx1, Ny1, Nz1;
5     int Nxp, Nyp, Nzp;
6     bool is_3d;
7

```

```

8   myreal Lx, Ly, Lz;
9   myreal dx, dy, dz;
10  myreal idx , idy , idz ; // 1/dx
11  myreal idx2, idy2, idz2; // 1/dx/dx
12
13  int dbx, dby, dbz;      // dimBlock ... in C ordering
14  int BLOCK_SIZE_3D;
15
16  int Nxpp, Nypp, Nzpp;
17  int sx, sy, sz;      // stride , sz == 1
18  size_t Ndata;
19  size_t NxpNypNzp;
20  int dimGrid_y;
21  ...
22  }
23
24  __constant__ GridParamsBase GP;

```

The computation domain $(N_x + 2) \times (N_y + 2) \times (N_z^{padded})$ is split into blocks of size $\text{dimBlock} = (\text{dbx}, \text{dby}, \text{dbz})$. There are several considerations for choosing the block size. For the optimal usage of SM resources, we want $\text{dbx} * \text{dby} * \text{dbz}$ to be roughly 64 - 256. Cache works in blocks of 128 bytes, which amounts to sixteen floating point doubles, each taking up 8 bytes. Therefore, to maximize coalesced memory access, we want dbz to be a factor of 16. Some optimization can be done if the ghost points and the points that use them belong to the same block. Thus we want dbx and dby to be of size two or more. For our biofilm code, we set $\text{dimBlock} = (2, 2, 16)$.

Once we fix the block size, we can calculate the number of blocks needed to cover the entire computation domain. This is called the grid dimension, $\text{dimGrid} = (\lceil \frac{N_x + 2}{\text{dbx}} \rceil, \lceil \frac{N_y + 2}{\text{dby}} \rceil, \lceil \frac{N_z + 2}{\text{dbz}} \rceil)$. Each computation kernel is then given launch parameters through a special CUDA syntax

```

1  kernel_name <<<dimGrid, dimBlock>>> (arg1, arg2, arg3, ...);

```

This command launches roughly $(N_x + 2) \times (N_y + 2) \times (N_z^{padded})$ threads, each running the same kernel code. Each thread is automatically given the variables blockIdx and threadIdx , which together can be used to figure out which data point the thread is responsible for.

```

1  int i = blockIdx.z * blockDim.z + threadIdx.z;
2  int j = blockIdx.y * blockDim.y + threadIdx.y;
3  int k = blockIdx.x * blockDim.x + threadIdx.x;

```

The indices i , j , k and x , y , z above are paired up in the opposite way of what one would normally expect. This is because the CUDA kernel launch assumes that threads access data in the way that x is the fastest varying coordinate, as is customary in Fortran language. However, the biofilm code and the HDF5 library, which we use for saving files, store data in the way that z is the fastest varying coordinate, as is customary in C/C++ languages. Thus the coordinates must be reversed at some point in the program.

Unfortunately, GPU devices of capability 1.3, which include Tesla C1060, can handle only two dimensions in `dimGrid`. We get around this by folding the x and y dimensions of `dimGrid` together, $dimGrid = (\lceil \frac{N_x+2}{dbx} \rceil \times \lceil \frac{N_y+2}{dby} \rceil, \lceil \frac{N_z+2}{dbz} \rceil)$. The kernel then needs to unfold this.

```

1  int blockIdx_z = blockIdx.y / GP.dimGrid_y;
2  int blockIdx_y = blockIdx.y % GP.dimGrid_y;
3  int i = blockIdx_z * blockDim.z + threadIdx.z;
4  int j = blockIdx_y * blockDim.y + threadIdx.y;
5  int k = blockIdx.x * blockDim.x + threadIdx.x;
6  int idx = i*GP.sx + j*GP.sy + k;
7  if (i>=GP.Nxp || j>=GP.Nyp || k>=GP.Nzp) return;

```

The snippet above also includes the calculation of the index `idx`, which the thread uses to fetch and store data. It is possible that the padded data size is not a multiple of the block size. For example, we might have $N_z + 2 < dimGrid.z * dimBlock.z$. In this case, there will be some threads that are launched to take care of the grid locations outside the computational domain, such as $k = N_z + 3, N_z + 4, \dots$. We do not need these threads, thus we stop them as shown in the last line of the above snippet.

An example of a complete kernel is given in the next snippet. This kernel is a part of the Helmholtz solver described in Appendix A. This kernel's sole duty is to divide the eigenvalues from the eigenmodes.

```

1  __global__

```

```

2 void divide_eigenval(cufftComplex *data, myreal coef, myreal lamb,
3                     myreal scale, int dsx, int dsy) {
4     let_ijk(i,j,k); // Compute the value of i,j,k,idx as done above.
5     if (!(i<GP.Nx && j<=GP.Ny && k<GP.Nz/2+1)) return;
6
7     if (i==0 && j==0 && k==0) {
8         if (abs(lamb)<1e-30) {
9             data[0].x = 0.; data[0].y = 0.; return;
10        }
11    }
12    myreal eigen = scale * (lamb
13        - coef * (+ 4. * pow2(sin(PI*i/GP.Nx      )) * GP.idx2
14                + 4. * pow2(sin(PI*j/(2*GP.Ny))) * GP.idy2
15                + 4. * pow2(sin(PI*k/GP.Nz      )) * GP.idz2));
16    data[i*dsx + j *dsy + k].x /= eigen;
17    data[i*dsx + j *dsy + k].y /= eigen;
18
19    if (1<=j && j<GP.Ny) {
20        data[i*dsx + (2*GP.Ny-j)*dsy + k].x /= eigen;
21        data[i*dsx + (2*GP.Ny-j)*dsy + k].y /= eigen;
22    }
23 }

```

This Helmholtz solver is implemented on top of the CUFFT package. It could have been implemented more efficiently through discrete cosine transform (DST). However, no DST package is available on CUDA at the time of writing. Since CUFFT is highly optimized by Nvidia, our own in-house implementation of DST will likely be much slower. We thus stay with CUFFT.

Instead of discretizing each equation out by hand, each equation is coded as a composition of basic operators, such as gradient, laplacian, and divergence. This makes it easier to catch bugs. Any programming error in these basic operators would show up in multiple equations, hence more likely to catch attention and get fixed. As an example, the following snippet shows code from three separate kernels. Together, they handle the EPS volume fraction equation (3.6), formulated as $Ax = b$ and solved using the BiCG-stab method. The first part computes the right hand side quantity b . The next two parts together compute Ax based on the input x .

```

1 RHS = ST.idt_2 * (4.*ST.phi[idx] - ST.phi_old[idx])
2     - advect(ST.velo, ST.phi, idx, i, j, k);

```

```

3 |
4 | // Second derivative of bulk mixing free energy
5 | __device__ myreal f2(const myreal phi, const myreal eps) {
6 |     return NP.NP_inv/(phi+eps) + 1./(1-phi) - 2.*NP.Kai_CH;
7 | }
8 | myreal pb = ST.phi_bar[idx];
9 | temp1[idx] = lam1 * laplacian(x,idx);
10 | temp2[idx] = lam2 * pb * f2(pb,eps);
11 |
12 | Ax = ST.idt3_2 * x[idx]
13 |     + div_scalar_grad(ST.phi_bar, temp1, idx)
14 |     - div_scalar_grad(temp2, x, idx)
15 |     - x[idx] * g_n_tilde(ST.phi_bar[idx], ST.cc_bar[idx]);

```

As an example, we show the implementation of a basic operator that was used in the earlier snippet,

```

1 | inline __device__
2 | myreal div_scalar_grad(const myreal *a, const myreal *b) {
3 |     // return div (a grad b), where a and b are both cell centered.
4 |     return 0.5 * (
5 |         + GP.idx2 * ((a[ GP.sx ] + a[0]) * (b[GP.sx] - b[0])
6 |                     -(a[-GP.sx] + a[0]) * (b[0] - b[-GP.sx]))
7 |         + GP.idy2 * ((a[ GP.sy ] + a[0]) * (b[GP.sy] - b[0])
8 |                     -(a[-GP.sy] + a[0]) * (b[0] - b[-GP.sy]))
9 |         + GP.idz2 * ((a[ GP.sz ] + a[0]) * (b[GP.sz] - b[0])
10 |                    -(a[-GP.sz] + a[0]) * (b[0] - b[-GP.sz]))
11 |     );
12 | }
13 |
14 | inline __device__
15 | myreal div_scalar_grad(const myreal *a, const myreal *b, int idx) {
16 |     return div_scalar_grad(a+idx, b+idx);
17 | }

```

In addition to the EPS volume fraction, the nutrient equation is also solved by BiCG-stab. We use the implementation provided by the CUSP library [19]. This library provides several ways to represent a sparse matrix. If one discretizes these equations into the form $Ax = b$ for a domain of size N^3 , the matrix A will have $7N^3$ to $25N^3$ non-zero elements. Each element requires 4 bytes for the index and 8 bytes for the double floating point value. As an example, for a 256^3 grid, a sparse array A with $25N^3$ non-zero elements requires $12 \times 25 \times 256^3$ bytes, which is just over 5 GB of memory. This is too big, considering that

our largest GPU has just under 6 GB of memory, and we still need some memory space for other variables. Instead of writing the coefficients of A out explicitly, we write a function that carries out the operation of multiplying by A . This requires no additional memory space, aside from those already used to hold $\phi_n, \mathbf{v}, c, \tau$ and other variables that form the coefficients of the equation. As an added advantage, this type of implementation tends to naturally access memory in a coalesced manner.

Instead of allocating and freeing the GPU memory manually, which can be error prone, we use Thrust library. Memory used by temporary variables are automatically freed at the end of its naming scope. Thrust library also allows us to quickly write a GPU kernel that fits the map-reduce paradigm [12] such as max, sum, norm, and clipping. The parallel communications that are needed to efficiently accomplish such action are automatically handled by the library.

Data are written out in HDF5 file format, which can be loaded into Python or Matlab for further analysis. The file can also be read by VisIt [36] or Paraview [31] for visualization.

3.6 MESH REFINEMENT

To check for the code's correctness and accuracy, we perform mesh refinement tests under two very different characteristic timescales: growing a biofilm with $t_0 = 1000$ sec, and shearing a biofilm with $t_0 = 1$ sec. The code is tested in both 2-D and 3-D geometries.

The truncation errors are proportional to the smoothness of functions in the system. If their derivatives are very large, a mesh refinement might not show a clear convergence order at the grid sizes that one runs the simulations. In order to avoid this problem, while conducting mesh refinement, we set up our initial biofilm profile to be smoother than what we normally use by increasing the biofilm-solution interface width. These initial profiles are shown in Fig.3.7, 3.9. We also reduce the viscosity ratio $\frac{\eta_n + \eta_{ps}}{\eta_s}$ from 10^4 to 10^3 . Since the volume fraction ϕ_n and nutrient concentration c are solved by an iterative method, it is also crucial to set the convergence criteria to be small enough so that the residual error will

not contaminate the refinement tests. Here we set the BiCG-stab relative tolerance for ϕ_n and c to be 10^{-12} .

The limited amount of GPU memory places an upper limit on the grid size that we can use. In 3-D geometries, we can simulate on a grid size up to 240^3 for the viscoelastic model or 256^3 for the viscous model. In 2-D geometries, we can simulate on a grid size up to just over 1024^2 for both models. We perform mesh refinements in both 2-D and 3-D geometries. Apparently, the 2-D mesh refinements allows us to refine the geometry spatially into a finer grid size than that is possible in 3-D. On the other hand, 3-D mesh refinements test the code base more thoroughly.

For 2-D spatial refinement, we fix $\Delta t = 10^{-4}$ for growing a biofilm and $\Delta t = 10^{-5}$ for shearing a biofilm, then run the simulations on n^2 grids where $n = 32, 64, \dots, 1024$. The dimensionless domain size is 1×1 . Thus $\Delta x = \Delta y = 1/n$. We use the finest grid as the reference solution. Appendix B discusses the details of the convergence rate calculation. The convergence order reported here is calculated from (B.8). We grow a bump of biofilm until $t = 2$. The initial and final biofilm profiles are shown in Fig.3.7 for 2-D and 3.8 for 3-D. Table 3.2 depicts the spatial refinement for all the governing equations. A clear 2nd order convergence rate is established for physical variables $\phi_n, c, \tau_p, \mathbf{v}$ in both the L_2 and L_∞ norm. Note that the numerical scheme for the advection of τ_p is spatially first order. However, the velocity field is so weak that this first order truncation error does not show up in the refinement result even at the finest grid size $\Delta x = \frac{1}{1024}$. For the refinement test in time, we fix a spatial grid size $\Delta x = \frac{1}{1024}$ and then vary Δt . Table 3.3 clearly demonstrates the first order convergence.

For 3-D spatial refinements, the finest grid that we can simulate is roughly 240^3 . One possible setup is to run the simulations over the grid sizes n^3 for $n = 30, 60, 120, 240$. The problem with this scheme is that most of the grids are very coarse, thus the convergence results can be contaminated by the higher order terms in the errors. To alleviate this problem, we run simulation on n^3 grids where $n = 48, 96, 144, 192, 240$. This let us run more simu-

lations at the finer grids. Note that these numbers are all factors for 48, thus their results can easily be sampled to the 48^3 grids for comparison. As in the 2-D case, the time step is fixed at $\Delta t = 10^{-4}$ or 10^{-5} . The domain size is $1 \times 1 \times 1$. Thus $\Delta x = \Delta y = \Delta z = 1/n$. Like in the case of 2-D mesh refinement tests, the 2nd order spatial and first order temporal order are established through these numerical experiments.

We then conduct a numerical experiment in which a piece of biofilms is sheared under a constant velocity up to $t = 2$. The initial and final biofilm profiles are shown in Fig.3.9 and 3.10. The refinement results are summarized in Table.3.6 and 3.7 for the 2-D case, and in Table.3.8 and 3.9 for the 3-D case. Temporally, all key values converges first-order. Spatially, the stress τ_p converges first-order; its computed convergence rate is almost one, and probably will converge to one at finer grid sizes. The EPS volume fraction ϕ_n and velocity field v converge at the rate somewhere between the second order, as expected by our discretization of the momentum equation, and the third order, as expected by the WENO advection scheme that we use for the biomass volume fraction. The conclusion applies to both the 2-norm and the infinity norm. The reduction of convergence rate for some of the physical variable is the consequence of enhanced coupling among the elastic stress and the other physical quantities. Since the scheme we adopt for the elastic stress τ_p is first order along the "streamline", it is first order in both space and time. This reduced accuracy propagates into the governing system so as to lower the order of other physical variables. But, overall, the scheme shows what we designed it for, at least first order in both space and time.

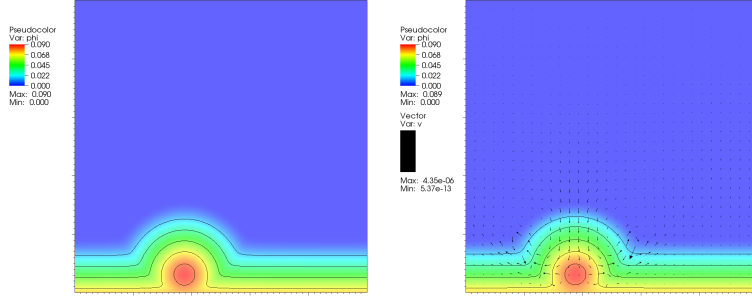


Figure 3.7: 2-D mesh refinement of growing a biofilm. Profiles at $t = 0$ and $t = 2$.

Δx	ϕ_n			c		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/32	1.29e-4			2.37e-5		
1/64	4.40e-5	2.92	1.53	6.01e-6	3.94	1.98
1/128	1.29e-5	3.42	1.75	1.48e-6	4.06	2.00
1/256	3.11e-6	4.14	1.98	3.51e-7	4.22	2.01
1/512	6.24e-7	4.98	1.99	6.94e-8	5.06	2.02
1/1024	reference					
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/32	8.50e-4			8.93e-5		
1/64	4.42e-4	1.92	0.88	2.27e-5	3.94	1.97
1/128	1.20e-4	3.68	1.86	5.62e-6	4.04	2.00
1/256	3.04e-5	3.94	1.90	1.33e-6	4.21	2.00
1/512	6.08e-6	5.00	2.00	2.65e-7	5.04	2.01
1/1024	reference					

Δx	τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/32	3.87e+1			5.28e-7		
1/64	1.33e+1	2.92	1.53	1.59e-7	3.32	1.72
1/128	3.84e+0	3.46	1.77	4.15e-8	3.84	1.92
1/256	9.15e-1	4.19	2.00	9.90e-9	4.19	1.99
1/512	1.82e-1	5.03	2.01	1.98e-9	5.01	2.00
1/1024	reference					
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/32	4.06e+2			4.99e-6		
1/64	2.13e+2	1.90	0.87	1.59e-6	3.13	1.64
1/128	5.90e+1	3.61	1.83	4.03e-7	3.95	1.96
1/256	1.49e+1	3.96	1.91	9.74e-8	4.14	1.98
1/512	3.00e+0	4.97	1.99	1.96e-8	4.97	1.99
1/1024	reference					

Table 3.2: 2-D spatial refinement result for growing a biofilm with $\Delta t = 10^{-4}$

Δt	ϕ_n			c		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/200	1.02e-6			2.01e-6		
1/400	4.99e-7	2.04	1.00	9.85e-7	2.04	1.00
1/800	2.39e-7	2.09	1.00	4.72e-7	2.09	1.00
1/1600	1.09e-7	2.19	1.00	2.16e-7	2.19	1.00
1/3200	4.44e-8	2.46	0.99	8.74e-8	2.47	1.00
1/10000	reference					
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/200	5.20e-6			3.41e-6		
1/400	2.55e-6	2.04	1.00	1.67e-6	2.04	1.00
1/800	1.22e-6	2.08	1.00	8.02e-7	2.08	1.00
1/1600	5.59e-7	2.19	1.00	3.66e-7	2.19	1.00
1/3200	2.27e-7	2.47	1.00	1.49e-7	2.47	1.00
1/10000	reference					

Δt	τ_p			v		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/200	8.28e+0			8.03e-8		
1/400	4.05e+0	2.04	1.00	4.07e-8	1.97	0.95
1/800	1.94e+0	2.09	1.00	1.96e-8	2.08	0.99
1/1600	8.86e-1	2.19	1.00	8.91e-9	2.20	1.01
1/3200	3.59e-1	2.47	1.00	3.59e-9	2.48	1.01
1/10000	reference					
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/200	4.62e+2			1.77e-6		
1/400	2.20e+2	2.10	1.04	8.10e-7	2.19	1.11
1/800	1.06e+2	2.07	0.99	3.62e-7	2.24	1.12
1/1600	4.89e+1	2.18	0.99	1.59e-7	2.28	1.07
1/3200	1.98e+1	2.46	0.99	6.29e-8	2.52	1.04
1/10000	reference					

Table 3.3: 2-D temporal refinement result for growing a biofilm with $\Delta x = 1/1024$

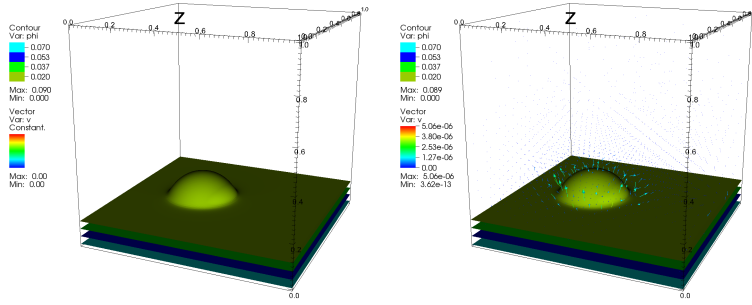


Figure 3.8: 3-D mesh refinement of growing a biofilm. Profiles at $t = 0$ and $t = 2$.

Δx	ϕ_n			c		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/48	3.78e-5			7.84e-6		
1/96	8.57e-6	4.41	1.94	1.70e-6	4.60	2.01
1/144	3.03e-6	2.83	1.86	5.77e-7	2.96	2.00
1/192	9.68e-7	3.13	1.94	1.82e-7	3.16	2.00
1/240	reference					
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/48	5.02e-4			4.64e-5		
1/96	1.09e-4	4.61	2.01	1.01e-5	4.59	2.01
1/144	3.69e-5	2.95	2.00	3.41e-6	2.96	2.00
1/192	1.17e-5	3.16	2.00	1.08e-6	3.16	2.01
1/240	reference					

Δx	τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/48	1.27e+1			1.92e-7		
1/96	3.32e+0	3.82	1.69	4.90e-8	3.91	1.73
1/144	1.21e+0	2.74	1.73	1.71e-8	2.86	1.89
1/192	3.92e-1	3.09	1.88	5.49e-9	3.12	1.93
1/240	reference					
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/48	2.38e+2			3.90e-6		
1/96	5.53e+1	4.30	1.90	9.14e-7	4.27	1.88
1/144	1.96e+1	2.82	1.84	3.14e-7	2.92	1.96
1/192	6.28e+0	3.13	1.94	9.96e-8	3.15	1.98
1/240	reference					

Table 3.4: 3-D spatial refinement result for growing a biofilm with $\Delta t = 10^{-4}$

Δt	ϕ_n			c		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/200	9.90e-7			1.62e-6		
1/400	4.85e-7	2.04	1.00	7.96e-7	2.04	1.00
1/800	2.33e-7	2.09	1.00	3.82e-7	2.08	0.99
1/1600	1.06e-7	2.19	1.00	1.76e-7	2.17	0.99
1/3200	4.31e-8	2.47	1.00	7.22e-8	2.44	0.97
1/10000	reference					
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/200	6.56e-6			3.16e-6		
1/400	3.25e-6	2.02	0.98	1.55e-6	2.03	0.99
1/800	1.56e-6	2.08	0.99	7.47e-7	2.08	0.99
1/1600	7.15e-7	2.18	1.00	3.43e-7	2.18	0.99
1/3200	2.90e-7	2.47	1.00	1.40e-7	2.44	0.98
1/10000	reference					

Δt	τ_p			v		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/200	8.13e-1			7.08e-8		
1/400	3.99e-1	2.04	1.00	3.27e-8	2.16	1.09
1/800	1.91e-1	2.09	1.00	1.49e-8	2.20	1.09
1/1600	8.72e-2	2.19	1.00	6.61e-9	2.25	1.05
1/3200	3.53e-2	2.47	1.00	2.64e-9	2.50	1.03
1/10000	reference					
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/200	7.65e+0			1.89e-6		
1/400	3.76e+0	2.03	0.99	8.65e-7	2.18	1.10
1/800	1.81e+0	2.08	1.00	3.84e-7	2.25	1.13
1/1600	8.26e-1	2.19	1.00	1.67e-7	2.29	1.08
1/3200	3.34e-1	2.47	1.00	6.61e-8	2.53	1.05
1/10000	reference					

Table 3.5: 3-D temporal refinement result for growing a biofilm with $\Delta x = 1/240$

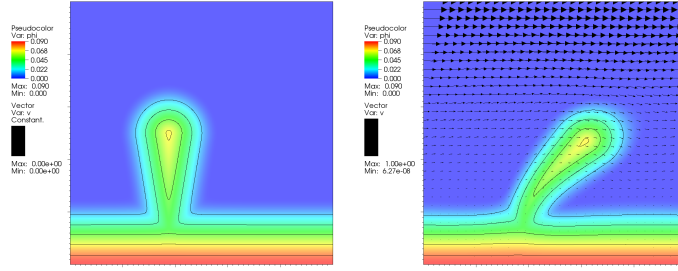


Figure 3.9: 2-D mesh refinement of shearing a biofilm. Profiles at $t = 0$ and $t = 2$.

Δx	ϕ_n			τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/32	8.29e-4			4.98e-4			5.25e-3		
1/64	3.54e-4	2.34	1.20	2.78e-4	1.79	0.76	1.63e-3	3.22	1.68
1/128	1.15e-4	3.07	1.58	1.42e-4	1.96	0.84	3.65e-4	4.46	2.14
1/256	2.30e-5	5.02	2.28	6.55e-5	2.16	0.85	6.00e-5	6.08	2.57
1/512	2.31e-6	9.95	3.16	2.32e-5	2.82	0.87	9.62e-6	6.24	2.39
1/1024	reference								
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/32	7.55e-3			5.79e-3			2.67e-2		
1/64	4.72e-3	1.60	0.56	4.08e-3	1.42	0.31	8.69e-3	3.07	1.61
1/128	2.09e-3	2.26	1.09	2.47e-3	1.65	0.51	2.12e-3	4.10	2.02
1/256	5.79e-4	3.60	1.76	1.29e-3	1.92	0.61	3.83e-4	5.54	2.43
1/512	7.55e-5	7.66	2.74	5.01e-4	2.57	0.65	6.44e-5	5.95	2.31
1/1024	reference								

Table 3.6: 2-D spatial refinement result for shearing a biofilm with $\Delta t = 10^{-5}$

Δt	ϕ_n			τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/1250	4.11e-5			1.41e-5			1.66e-4		
1/2500	2.04e-5	2.02	0.99	7.14e-6	1.97	0.96	8.15e-5	2.03	1.01
1/5000	9.95e-6	2.05	1.00	3.52e-6	2.03	0.98	3.96e-5	2.06	1.00
1/10000	4.71e-6	2.11	1.00	1.68e-6	2.10	0.99	1.88e-5	2.11	1.00
1/20000	2.09e-6	2.25	1.00	7.48e-7	2.24	0.99	8.33e-6	2.25	1.00
1/100000	reference								
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/1250	1.74e-4			1.89e-4			6.33e-4		
1/2500	8.59e-5	2.03	1.00	9.93e-5	1.90	0.90	3.09e-4	2.05	1.02
1/5000	4.17e-5	2.06	1.00	4.99e-5	1.99	0.95	1.49e-4	2.07	1.01
1/10000	1.97e-5	2.12	1.00	2.40e-5	2.08	0.97	7.05e-5	2.12	1.01
1/20000	8.75e-6	2.25	1.00	1.08e-5	2.23	0.99	3.13e-5	2.25	1.00
1/100000	reference								

Table 3.7: 2-D temporal refinement result for growing a biofilm with $\Delta x = 1/1024$

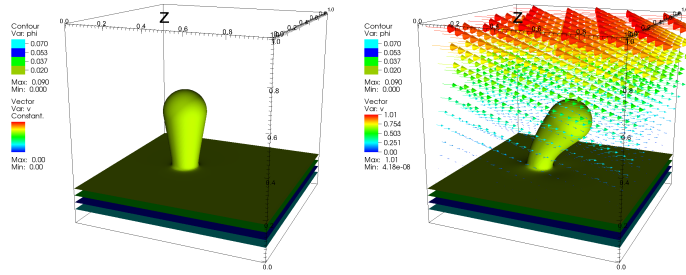


Figure 3.10: 3-D mesh refinement of shearing a biofilm. Profiles at $t = 0$ and $t = 2$.

Δx	ϕ_n			τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/48	1.90e-4			1.26e-4			1.53e-3		
1/96	4.79e-5	3.96	1.75	5.52e-5	2.29	0.67	3.53e-4	4.32	1.90
1/144	1.49e-5	3.21	2.28	2.53e-5	2.18	0.87	1.13e-4	3.13	2.19
1/192	4.15e-6	3.59	2.69	9.65e-6	2.62	0.89	3.36e-5	3.36	2.33
1/240	reference								
Δx	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/48	5.62e-3			3.51e-3			1.18e-2		
1/96	2.10e-3	2.68	1.01	1.78e-3	1.97	0.31	3.07e-3	3.83	1.69
1/144	8.26e-4	2.54	1.46	8.83e-4	2.02	0.55	9.75e-4	3.14	2.21
1/192	2.63e-4	3.14	1.97	3.38e-4	2.61	0.87	2.73e-4	3.57	2.66
1/240	reference								

Table 3.8: 3-D spatial refinement result for shearing a biofilm with $\Delta t = 10^{-5}$

Δt	ϕ_n			τ_p			\mathbf{v}		
	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order	$\ err\ _2$	ratio	order
1/250	6.07e-5			1.76e-5			4.46e-4		
1/500	2.98e-5	2.04	1.02	9.36e-6	1.88	0.91	2.11e-4	2.11	1.08
1/1000	1.47e-5	2.03	1.02	4.91e-6	1.91	0.92	1.02e-4	2.07	1.04
1/2000	7.18e-6	2.04	1.02	2.50e-6	1.96	0.96	4.98e-5	2.05	1.02
1/4000	3.48e-6	2.06	1.01	1.24e-6	2.01	0.98	2.42e-5	2.05	1.01
1/8000	1.66e-6	2.10	1.01	5.98e-7	2.07	0.99	1.16e-5	2.09	1.00
1/16000	7.54e-7	2.20	1.01	2.74e-7	2.18	0.99	5.28e-6	2.19	1.00
1/100000	reference								
Δt	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order	$\ err\ _\infty$	ratio	order
1/250	7.38e-4			3.82e-4			6.41e-3		
1/500	3.57e-4	2.07	1.04	2.00e-4	1.91	0.93	2.70e-3	2.38	1.25
1/1000	1.73e-4	2.06	1.04	1.16e-4	1.73	0.77	1.17e-3	2.31	1.20
1/2000	8.40e-5	2.06	1.03	6.16e-5	1.88	0.89	5.37e-4	2.17	1.11
1/4000	4.05e-5	2.08	1.03	3.13e-5	1.97	0.94	2.55e-4	2.10	1.05
1/8000	1.92e-5	2.11	1.02	1.52e-5	2.05	0.97	1.21e-4	2.11	1.02
1/16000	8.70e-6	2.20	1.01	7.01e-6	2.17	0.99	5.49e-5	2.20	1.01
1/100000	reference								

Table 3.9: 3-D temporal refinement result for growing a biofilm with $\Delta x = 1/240$

3.7 VISUALIZING A STRESS FIELD

Since stress is a two dimensional tensor, visualizing a stress field takes a little more work than visualizing a scalar or vector field. Given a symmetric stress τ , and a surface normal to a unit vector \mathbf{n} , the traction vector on that plane $\tau^{(\mathbf{n})} := \tau \cdot \mathbf{n}$ can be decomposed into the normal component and the shear component, $\tau^{(\mathbf{n})} = \tau_{normal}^{(\mathbf{n})} + \tau_{shear}^{(\mathbf{n})}$ where $\tau_{normal}^{(\mathbf{n})}$ is the traction vector in \mathbf{n} direction and $\tau_{shear}^{(\mathbf{n})}$ is the traction vector perpendicular to \mathbf{n} .

For a viscous flow in the x direction with the velocity gradient in the y direction, the shear stress is given by $\tau_{xy} = \tau_{shear}^{(y)}$. Without an imposed shear, however, there might not be a principal flow direction. In this situation, there is no reason why the x - y basis would be more special than any other orthonormal basis. Thus, τ_{xy} does not have a special meaning like earlier.

One can take the eigenvalue decomposition $\tau = Q\Lambda Q^{-1}$ where $\Lambda = diag(\sigma_1, \sigma_2, \sigma_3)$ with $\sigma_1 \geq \sigma_2 \geq \sigma_3$ and $Q = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3]$. Then, Q is called the principal frame, σ_i the principal stresses, and \mathbf{n}_i the corresponding principal directions. These directions have the following properties [7] [2],

- The maximum normal stress, $\max_{\mathbf{n}} \tau_{normal}^{(\mathbf{n})}$, is given by σ_1 . It occurs in the direction $\mathbf{n} = \mathbf{n}_1$.
- The minimum normal stress is $\min_{\mathbf{n}} \tau_{normal}^{(\mathbf{n})} = \sigma_3$. It occurs in the direction \mathbf{n}_3 .
- The maximum shear stress is $\tau_{max} := \max_{\mathbf{n}} \tau_{shear}^{(\mathbf{n})} = \frac{1}{2}(\sigma_1 - \sigma_3)$. It occurs on the plane that bisects \mathbf{n}_1 and \mathbf{n}_3 .

At each point in space, a symmetric stress τ can be represented by an ellipse in the 2-D or an ellipsoid in the 3-D, with its axes aligned with the principal directions and axis lengths proportional to the absolute values of the principal stresses $|\sigma_i|$. This yields an intuitive visualization of the stress. However, when some principal stresses are negative,

the plot can be misleading since a positive and a negative value of the principal stress can yield the same ellipsoid.

In an incompressible flow, the isotropic part of the stress does not affect the hydrodynamics since it is balanced by the pressure. Therefore, each individual principal stress σ_i is not as important as their differences $\sigma_i - \sigma_j$. Therefore, we visualize the hydrodynamically relevant part of the stress by plotting the max shear stress $\frac{1}{2}(\sigma_1 - \sigma_3)$ and the major principal direction \mathbf{n}_1 .

CHAPTER 4

NUMERICAL SIMULATION AND DISCUSSIONS

We study the dynamics of the biofilm-solvent interaction in both 2-D and 3-D geometries. Periodical boundary conditions are imposed on all physical unknowns except in the y-direction, where physical boundary conditions are imposed. We run the simulations under two different time scales: one is the biomass growth time scale and the other the flow-induced time scale. Biofilms grow under the biomass growth characteristic time $t_0 = 1000$ sec. In this regime, the corresponding dimensionless parameter values are,

$$\begin{aligned} \frac{1}{Re_s} = 1002, \quad \frac{1}{Re_n} + \frac{1}{Re_{ps}} = 1 \times 10^7, \quad \Lambda_1 = 1.10, \quad \Lambda = 1 \times 10^{-9}, \\ \Gamma_1 = 33.467, \quad \Gamma_2 = 1.29 \times 10^6, \quad D_s = 2.3, \\ A = 4.85 \times 10^3, \quad \mu = 0.40, \quad K_c = 0.0425, \quad K_1 = 0.0425. \end{aligned}$$

Biofilms are sheared under a much shorter flow induced characteristic time $t_0 = 1$ sec. The corresponding dimensionless parameter values are,

$$\begin{aligned} \frac{1}{Re_s} = 1.002, \quad \frac{1}{Re_n} + \frac{1}{Re_{ps}} = 1 \times 10^4, \quad \Lambda_1 = 1100, \quad \Lambda = 1 \times 10^{-6}, \\ \Gamma_1 = 3.347 \times 10^{-5}, \quad \Gamma_2 = 1.29, \quad D_s = 2.3 \times 10^{-3}, \\ A = 4.85, \quad \mu = 4.00 \times 10^{-4}, \quad K_c = 0.0425, \quad K_1 = 0.0425. \end{aligned}$$

Table 4.1 lists the range of the dimensional parameter values used in our simulations.

In these situations, in order to investigate how the EPS elasticity can impact the biofilm growth process and its hydrodynamics under an imposed shear, we conduct a comparative study to contrast the model prediction of the viscoelastic model versus the corresponding viscous model while the total biomass viscosity is kept the same. In the purely viscous

Symbol	Parameter	value	Unit
T	Temperature	303	Kelvin
γ_1	Distortional energy coefficient	8×10^6	m^{-1}
γ_2	Mixing free energy coefficient	3×10^{17}	m^{-3}
χ	Flory-Huggins parameter	0.55	
λ	Mobility parameter	1×10^{-9}	$\text{kg}^{-1}\text{m}^3\text{s}$
N	Generalized polymerization parameter	1×10^3	
D_s	Substrate diffusion coefficient	2.3×10^{-9}	m^2s^{-1}
A	Max. Consumption rate	4×10^{-2}	$\text{kg m}^{-3}\text{s}^{-1}$
μ	Max. Production rate	4×10^{-4}	s^{-1}
k_c	Monod constant for g_n	3.5×10^{-4}	kg m^{-3}
k_1	Monod constant for g_c	3.5×10^{-4}	kg m^{-3}
λ_1	Elastic relaxation time of EPS	1100	s
a	Slip coefficient	0.95	
α	Damping coefficient in Giesekus model	0.02	
η_n	Dynamic viscosity of EPS	$\left. \begin{array}{l} \eta_n + \eta_{ps} = 10. \\ \end{array} \right\}$	$\text{kg m}^{-1}\text{s}^{-1}$
η_{ps}	Dynamic viscosity of bacteria		
η_s	Dynamic viscosity of solvent	1.002×10^{-3}	$\text{kg m}^{-1}\text{s}^{-1}$
ρ_n	Network density	1×10^3	kg m^{-3}
ρ_s	Solvent density	1×10^3	kg m^{-3}
c_0	Characteristic substrate concentration	8.24×10^{-3}	kg m^{-3}
h	Characteristic length scale	1×10^{-3}	m
t_0	Characteristic time scale	1 or 1,000	s
L_x, L_y, L_z	size of computation domain	$1 - 3 \times 10^{-3}$	m
M_x, M_y, M_z	Number of sub-intervals in each direction	16 - 1024	

Table 4.1: Parameter values used in the simulations

case, we fix a value for the Reynolds number Re_{ps} and let $1/Re_n = 0$. In the viscoelastic case, we let $1/Re_n + 1/Re_{ps} = 1/Re_{ps}^{\text{purely-viscous}}$. In the results presented below, we pick $1/Re_n = 1/Re_{ps} = 1/(2Re_{ps}^{\text{purely-viscous}})$. The elastic stress constitutive equation utilizes either the network velocity \mathbf{v}_n or the average velocity \mathbf{v} for the transport and deformation. For brevity, we call the former “viscoelastic-N” and the latter “viscoelastic-A” model, respectively.

2-D biofilm growth dynamics

We simulate the growth of a small bud of biofilm under the characteristic time scale $t_0 = 1000 \text{ sec}$ until $t_{end} = 300$, which corresponds to roughly 3.5 days in reality. The numerical simulations are carried out on a 512×512 grid. The initial biofilm profile is shown in Fig.4.1. It grows into the final results shown in Fig.4.2. All three models (viscous, viscoelastic A and N) yield almost identical biofilm shapes and volume fraction distributions. In fine details, the biofilm profile given by the viscous and viscoelastic A model are identical which differs slightly from that of the viscoelastic N model. The three models yields very similar nutrient profiles, and the locations and rates of growth and nutrient consumption. Most growth and consumption occur near the biofilm-solution interface, where nutrient gets consumed quickly, thus cannot penetrate far into the biofilm. As additional biomass is produced during growth, it gets redistributed by the Modified Cahn-Hilliard dynamics to lower the global sum of the free energy. Fig.4.3 shows the flux of biomass migrating away from the high-growth region near the interface, expanding outward toward the solution. At the same time, the solution penetrates back into the biofilm causing the biofilm to swell up and cover a larger region. This accelerates the biofilm growth by increasing the surface area and bringing it closer to the nutrient source.

The biomass movement causes the EPS to elongate and deform evidenced in the viscoelastic models, thus generating elastic stresses. In a growing biofilm, the average velocity is quite small shown in Fig.4.3. Therefore, the biomass velocity $\mathbf{v}_n = \mathbf{v} - \lambda \nabla \frac{\delta f}{\delta \phi_n}$ is dominated by the excessive component $-\lambda \nabla \frac{\delta f}{\delta \phi_n}$, which is instantiated by molecular mixing between the small solvent molecule and the large EPS molecule and bacterial cell and carried out in the direction of $\pm \nabla \phi$. Thus, the biomass migrates mostly along the volume fraction gradient, compressing and elongating the EPS in this direction. Consequently, the viscoelastic-N model shows a strong non-isotropic normal stress in τ_n in the direction of the

volume fraction gradient, as illustrated in Fig.4.4. In contrast, the EPS in the viscoelastic-A model is deformed by the average velocity \mathbf{v} and its gradient, which is very weak in this setting. Hence, the elastic stress τ_n is essentially isotropic, and originates mostly from the biomass growth in the viscoelastic A model. Such isotropic stress is balanced out by the pressure in the incompressible system. Thus, despite its potentially high principal elastic stress, the viscoelastic-A model shows a relatively weak flow field and little viscous stress comparing to the viscoelastic-N model.

Both viscoelastic models, which contain a large elastic principal component, show a much higher pressure than the purely viscous model. This is because pressure exists essentially as a reaction to other forces in the system. Therefore, the pressure gradient generally has roughly the same strength as the strongest remaining force in the system. In viscoelastic models, the pressure is of the same strength as the elastic stress. In viscous-only model the pressure gradient is of the same strength as the force due to the Modified Cahn-Hilliard dynamics, which is weaker than the force due to the elastic stress by 1-2 order of magnitude.

To better emulate the spatial heterogeneity of the real biofilm, we compute the growth of scattered bits of biomass on a 512×1536 grid using the viscoelastic-N model. This produces a biofilm that grow into a finger formation, as shown in Fig.4.5. As in the previous case, biomass growth and nutrient consumption occur mostly near the biofilm-solution interface. It is worth pointing out that the nutrient concentration is low not only deep inside the biofilm, but also in the channels/cavities between biofilm lumps, as illustrated in Fig.4.6. The nutrient consumption deep in the channels is a consequence of the nutrient diffusion and consumption by nearby biomass. This situation will be mitigated either by a reduced consumption rate or by a reduced nutrient diffusion rate, which in return will retard the biomass growth. The tallest bud shows the highest growth rate and nutrient consumption rate, yet it does not become much taller than other buds due to the limited availability of nutrient.

The biomass flux is again strongest on the biofilm-solution interface, as illustrated in Fig.4.7. Note that the flux on the top interface is stronger than that on the side interface. This is likely due to the higher growth rate on the top interface and more readily available nutrient, which necessitates more biomass movement in order to settle into a free-energy equilibrium. The region of high elastic stress coincides with the region of high biomass flux, since the network velocity and flux are the primary causes of the elastic stress. The viscous stress is an order of magnitude weaker than the elastic stress in the simulations. As Fig.4.7 shows, the net force due to elastic stress is stronger than the net force due to viscous stress, both of which are more than an order of magnitude stronger than the interfacial force due to fluid mixing. Both the elastic force and the interfacial force switch directions across the biofilm interface. The elastic force tries to squeeze the interface thinner, while the interfacial force tries to pull the interface wider. The viscous stress force is unidirectional and does not show such a trend.

3-D biofilm growth dynamics

In 3-D geometries, we simulate biofilm growth on a $2 \times 1 \times 2$ domain with $256 \times 128 \times 256$ grid, while other parameters are kept the same as in the 2-D settings. The 3-D results are qualitatively the same as the 2-D results and of course with more heterogeneous details in the additional direction. All three models produce similar final biofilm profiles. A small bud of biofilm grows into a ball of mushroom shape as shown in Fig.4.9. Scattered bits of biomass grow in to one large connected colony of biofilm intertwined with water channels, as shown in Fig.4.10. Some small bits merge together and grow into a large lobe, while most medium-size bits retain their own identity and do not completely merge with their neighbors. They maintain an interface separating them from their neighbors by water channels. The heterogeneity in biofilm structure is a noticeable feature of biofilms.

2-D biofilms in shear flows

We take the biofilms grown in the previous section as the initial data, then apply a shear velocity of 1 mm/s at the top boundary, which corresponds to the nondimensional value of $v_{shear} = 1$ with the characteristic time scale $t_0 = 1s$. We apply shear from time $t = 0$ until $t = 200$. The shear velocity ramps up linearly from $t = 0$ to $t = 10$, stays constant until $t = 190$, then ramps down linearly until stop at $t = 200$. We keep the simulation running until $t = 800$ to observe further biomass movements. In this setting, the average velocity \mathbf{v} and the network velocity \mathbf{v}_n are almost identical. Thus, the viscoelastic-N and viscoelastic-A models yield the same result. Therefore, we only compare the results of the viscous model and the viscoelastic-N model.

We first shear the biofilm grown from scattered bits. The results are presented in Fig.4.11. Compared to the viscous model, the biofilm in the viscoelastic model distorts more under the shear flow. This is because the elastic stress initially offers no resistance at all to the shear. Only after the biofilm has been strained by some significant amount would it start to show a substantial elastic stress to counter the force of the shearing fluid.

The pressure concentrates mostly on the base corners of the biofilm. The upstream corner has a negative pressure, showing that it is being pulled. The downstream corner has a positive pressure, showing that it is being compressed. As Fig.4.11 shows, the tallest biofilm lobe experiences the highest elastic stress. Interestingly, the middle lobe experiences significant stress on its upstream interface, even though that interface is not directly in contact with the strong shear flow. While shearing, the faint streaming biomass has its principal directions for both the elastic stress and viscous stress at roughly 45° , which is the principal direction of the rate of strain tensor \mathbf{D} .

Just after the shear stops ($t = 200$), the viscous model predicts little remaining viscous stress, most of which is at the base of the biofilm. This stress quickly subsides, and the

biofilm's hydrodynamic stays mostly static in the absence of shear. The viscous model predicts that the biofilm essentially stops moving after the imposed shear ceases. In contrast, the viscoelastic model predicts that the biofilm gradually recoils back partially toward the original position. The elastic stress tries to contract the fluid along its principal stress direction. The incompressibility condition, in turn, causes the fluid to expand in the direction perpendicular to the contraction. This induces a flow throughout the biofilm, which in turn induces the viscous stress. Fig.4.12 illustrates this mechanism. It can be seen that the location of high viscous stress coincides with the location of high elastic stress, and that the viscous stress tends to have its principal direction perpendicular to that of the elastic stress.

The net elastic force, net viscous force, and net MCH force are strongest near the biofilm-solution interface, as illustrated in Fig.4.11. This is surprising at first, since the stress is distributed throughout the biofilm, so intuition tells us that their corresponding forces should act throughout the biofilm body. After a moment of reflection, one realizes that these forces are indeed strong inside the biomass. However, they are canceled out by an equally strong force in the opposite direction, leaving the net forces to be small inside the biomass. Near the biofilm interface, these opposing forces have unequal strength, since the elastic stress and bacterial viscous stress exist only in the biomass, thus we observe a high net force there.

We also shear the mushroom-shaped biofilm that we grew earlier. The result, presented in Fig.4.14, shows an interesting scenario where the tip of a biofilm bud stretches out due to the shear, and sticks to a different part of the biofilm. Consequently, the biofilm cannot recoil back as much as in the previous case. The pressure is low inside the mushroom bud, especially at its upstream neck which is being pulled by the shear. The bud is pushed against its downstream shoulder, creating a high pressure at the place they fuse. The tip of the bud stretches out due to the shear, creating high viscous and elastic stresses.

3-D biofilms in shear flows

For comparison, we also simulate sheared mushroom-shaped biofilms in 3-D. The result is similar to the 2-D setting for the most part, but the downstream side of the mushroom develops a tip in the viscous model, and a streamer nose in the viscoelastic model. This nose extends out from the main body and streams along with the flow, then partially retracts back once the imposed shear ceases. This happens because the 3-D geometry allows the fluid to flow around the sides of the mushroom and form a confluence on its downstream side. This confluence shears the biomass into a tip or a nose. The elastic stress offers less initial resistance to shear, thus the viscoelastic model predicts a biofilm that more readily elongates into a nose than that predicted by a viscous model. Once the nose develops, it continues to be sheared on all sides, thus maintaining its length against the elastic recoil. We did not observe this kind of tip and streamer nose during the 2-D shear. In a 2-D geometry, the fluid can strongly shear such a nose only on the top side. The bottom side would be a cavity with a relatively weak flow field. In addition, 2-D simulation can be regarded as a cross-section of the 3-D cylindrical geometry, which perhaps carry quite different hydrodynamic response to the truly 3-D closed shape carried out in the 3-D simulation.

The amount of biomass recoil in the viscoelastic model varies by cases. Fig.4.16 shows the results of shearing the 3-D biofilm colony that we have earlier grown from scattered bits. As expected, the viscous model predicts a biofilm that deforms less under the shear than that in the viscoelastic model. Surprisingly, the viscoelastic model predicts a biofilm that recoils back by only an extremely small amount. It appears that each biomass bud becomes stuck to the side of its neighboring buds in a similar fashion to the 2-D mushroom-shaped biofilm in Fig.4.14, thus creating a resistance to the recoiling force. We try shearing another colony of biofilm, with shorter and less number of buds. This way, each bud is less likely to be stuck to the side of its neighbor. As illustrated in Fig.4.17, the viscoelastic model predicts that this new colony of biofilm will recoil back partially as expected.

In order to observe how the elasticity affects detachment process, we shear an artificial

top-heavy biofilm with a thin neck, as illustrated in Fig.4.18. The shear rate is 10 times that of the previous cases. Detachment occurs sooner in viscoelastic models than in the purely-viscous model. This agrees with the earlier observation that elastic stress does not offer as much resistance to an applied shear as the viscous stress. For the viscoelastic case, we run numerical simulations for both the Giesekus model and the Phan-Thien-Tanner model of the elastic stress. They produce almost identical results. This phenomenon has to be understood in the context that the overall viscosity of the material is held constant in the simulation. The viscoelastic material is shear thinning and therefore the viscosity tends to be small under shear.

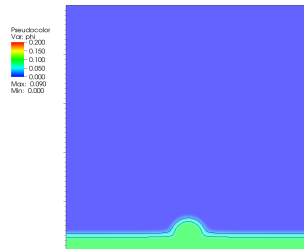


Figure 4.1: Initial biomass volume fraction ϕ_n profile.

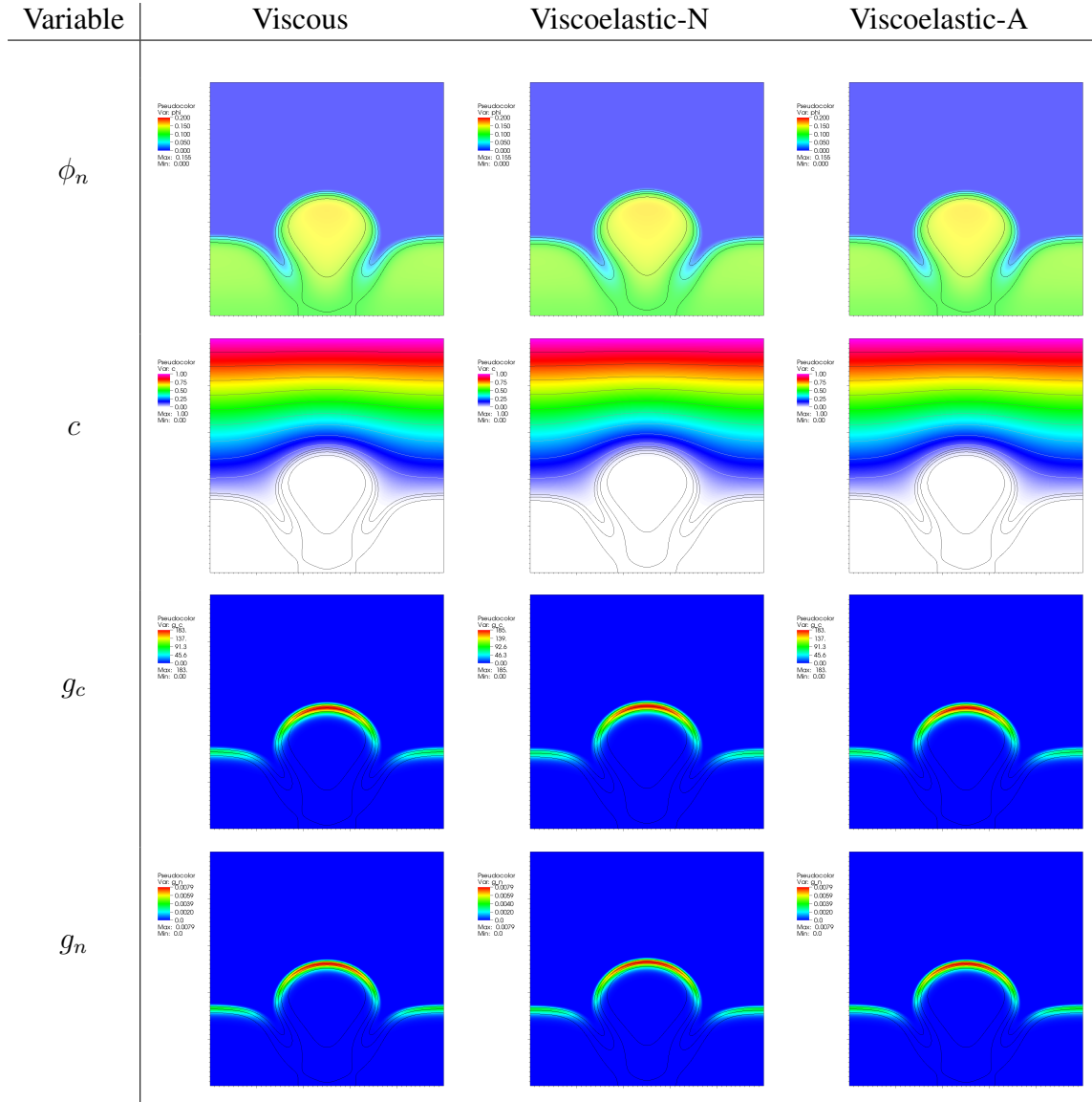


Figure 4.2: Biomass volume fraction ϕ_n , nutrient concentration c , nutrient consumption rate g_c , and biomass production rate g_n in the simulation of a growing bud of biofilm at $t = 300$. Results from three models (viscous, viscoelastic A & N) are contrasted. In these simulations, the three models give qualitatively and quantitatively the same results for all growth-related quantities.

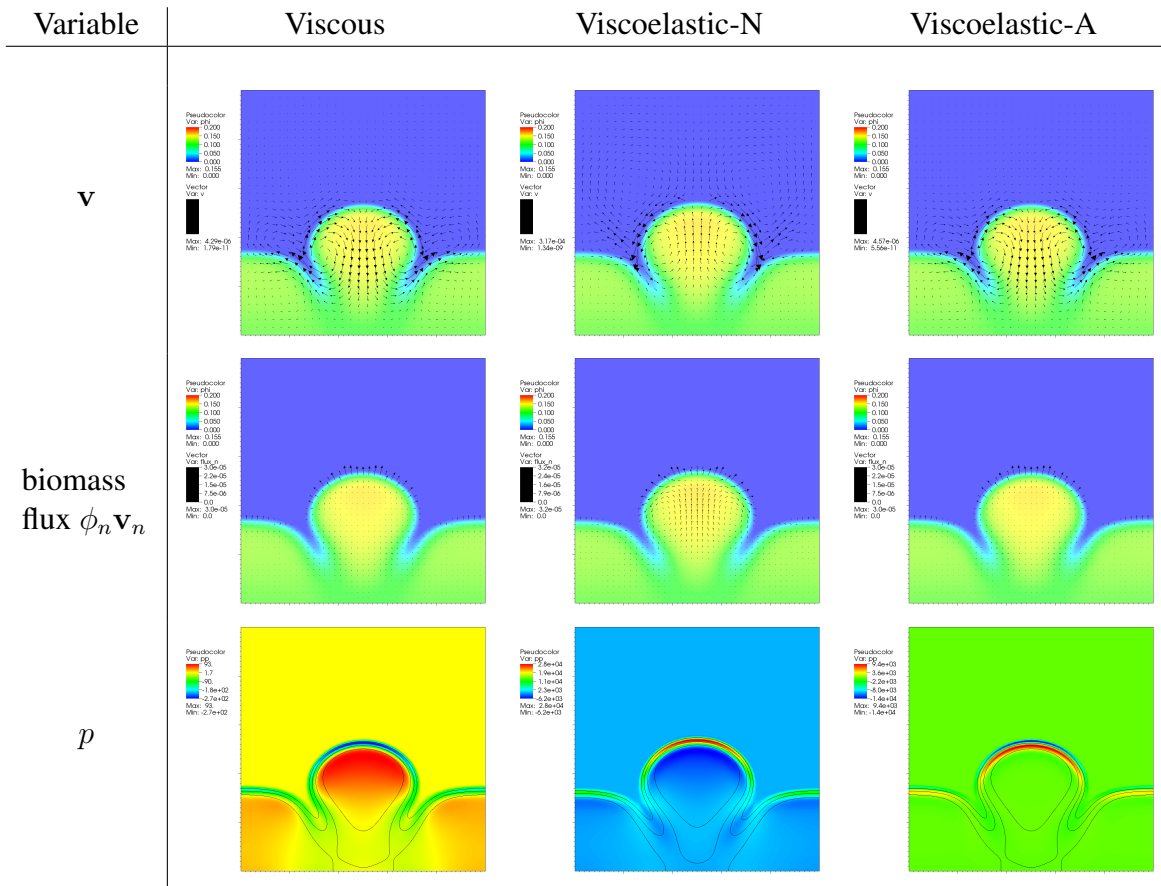


Figure 4.3: Velocity \mathbf{v} , biomass flux $\phi_n \mathbf{v}_n$, and pressure p in the simulation of a growing bud of biofilm. Roll cells form within the biofilm colony above the neck region. Both viscous and viscoelastic-A models predict quantitatively the same average velocity; whereas the viscoelastic-N model yields a slightly different average velocity. The pressure deviations in both viscoelastic models are much higher than in the viscous model.

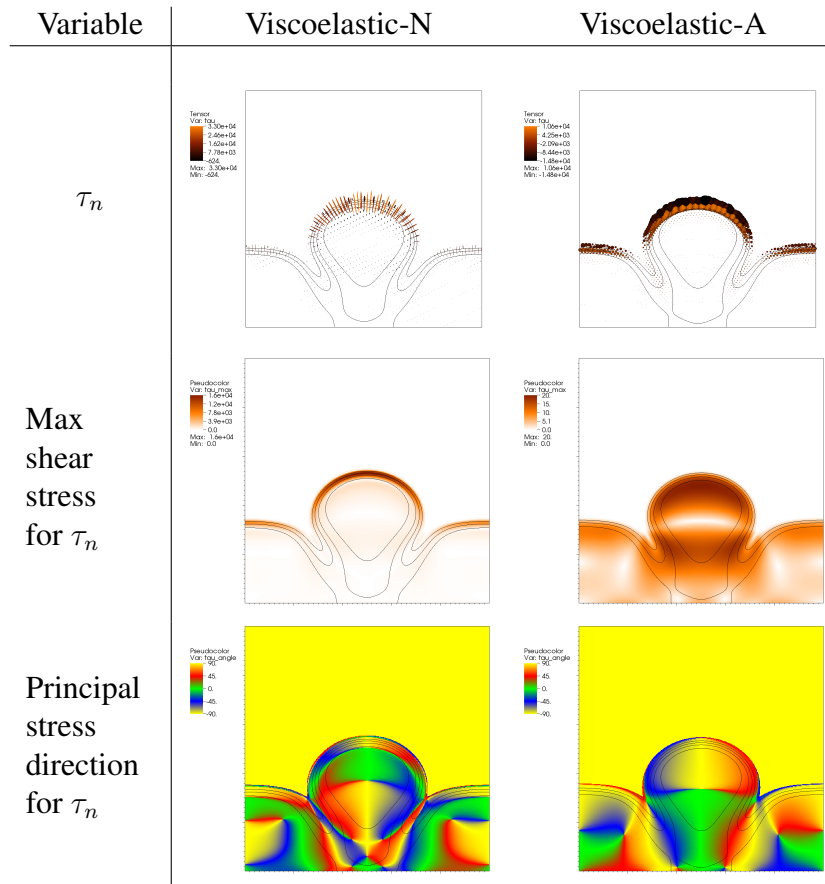


Figure 4.4: Elastic stress distributions. Sec.3.7 explains the terms and the meaning of each plot. The major principal direction is shown by the counterclockwise angle from the x axis. This value is meaningful only in the regions where the max shear stress is non-zero. Both viscoelastic models predict that the elastic stress concentrate mostly on the biofilm interface. The viscoelastic-N model predicts a stress that is in the direction of volume fraction gradient, while the viscoelastic-A model predicts an almost isotropic stress, which is balanced out by the pressure.

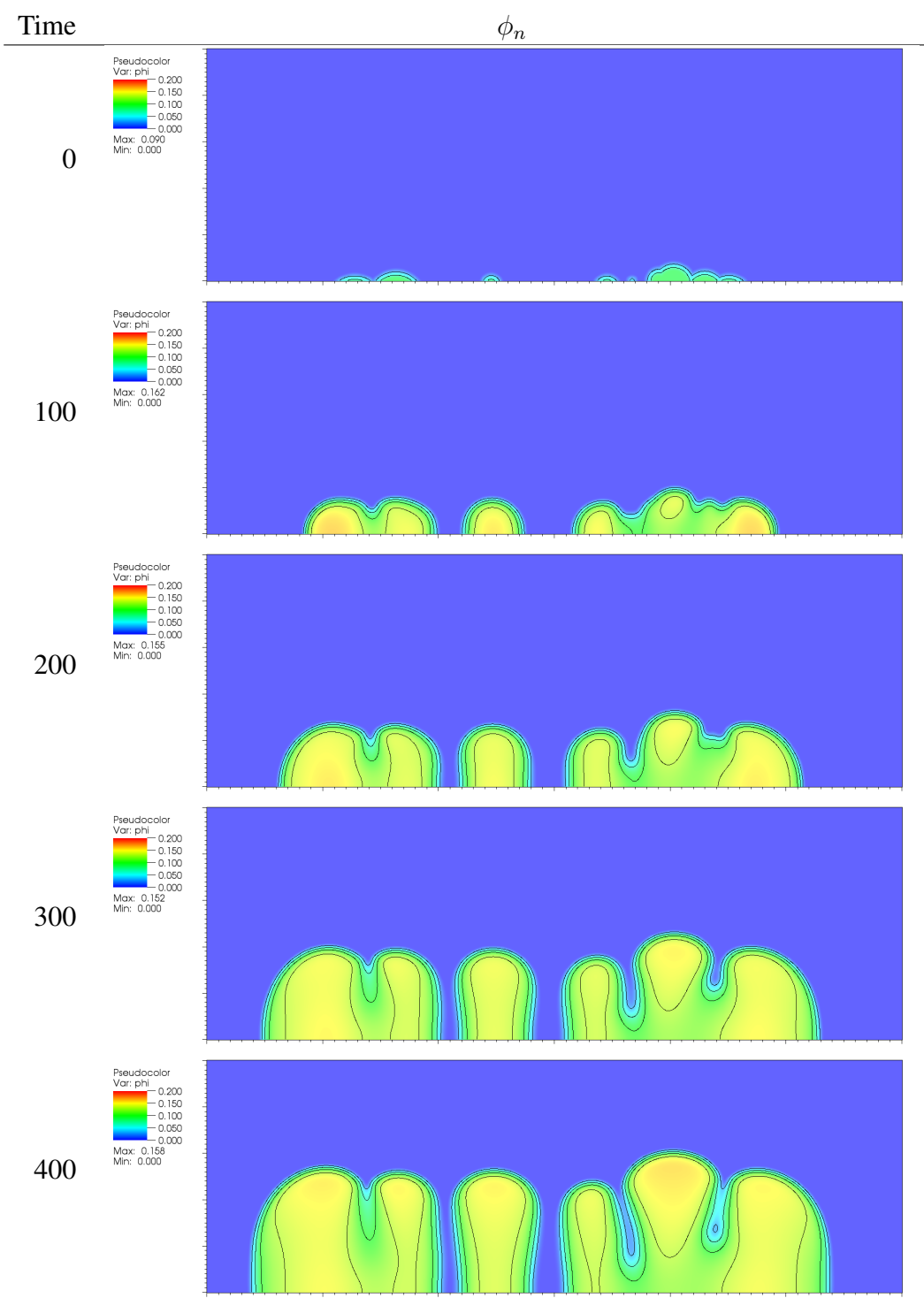


Figure 4.5: Snapshots of the simulation of a growing randomly scattered bits of biofilm on the 512×1536 grid. The biofilm grows into a finger formation. As each colony grows toward the top where the nutrient is fed, smaller colonies merge into bigger ones.

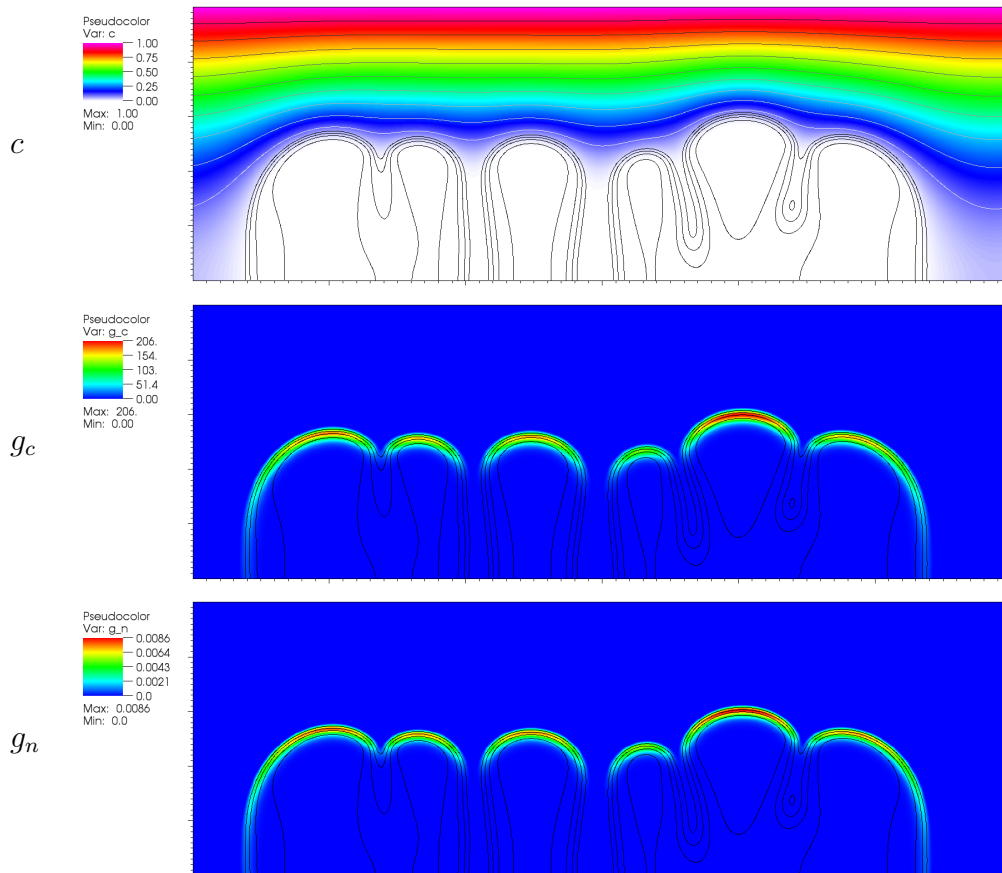


Figure 4.6: The left and right colonies grow faster than those in the middle since it can access nutrient through larger interfacial areas on the left/right of the biofilm interface in addition to the top interface. The tallest bud shows the highest growth rate and nutrient consumption rate, yet does not become much taller than other buds.

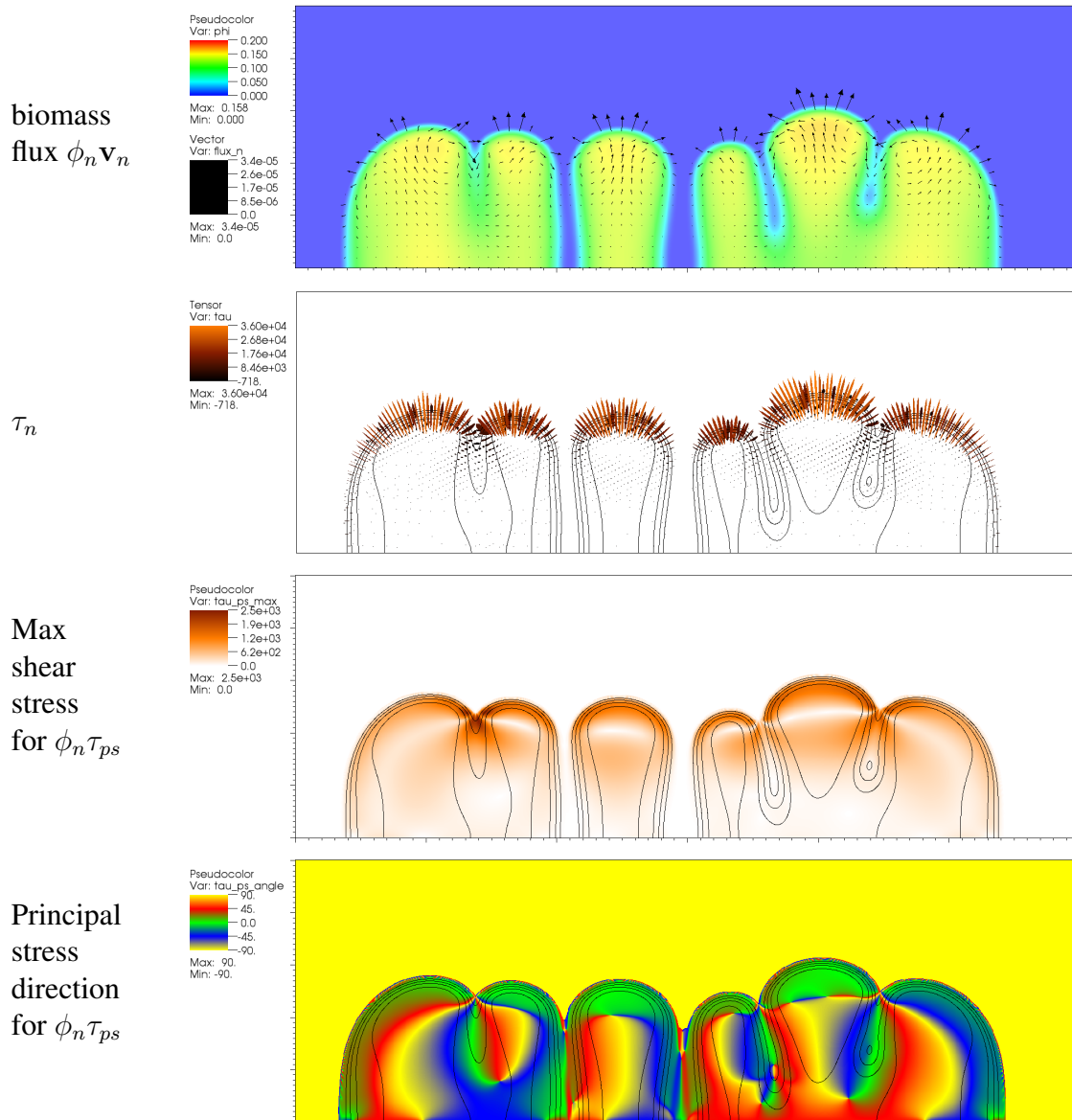


Figure 4.7: The biomass flux and elastic stress follow the same pattern as in the case of growing a bud of biofilm. Note that the elastic stress is higher on the top interface than on the sides because the top has higher biomass flux. Bacterial viscous stress $\phi_n \tau_{ps}$ are shown by its max shear stress and principal direction (explain in Fig.4.4). We avoid using the ellipsoid plot for viscous stresses since it can be misleading.

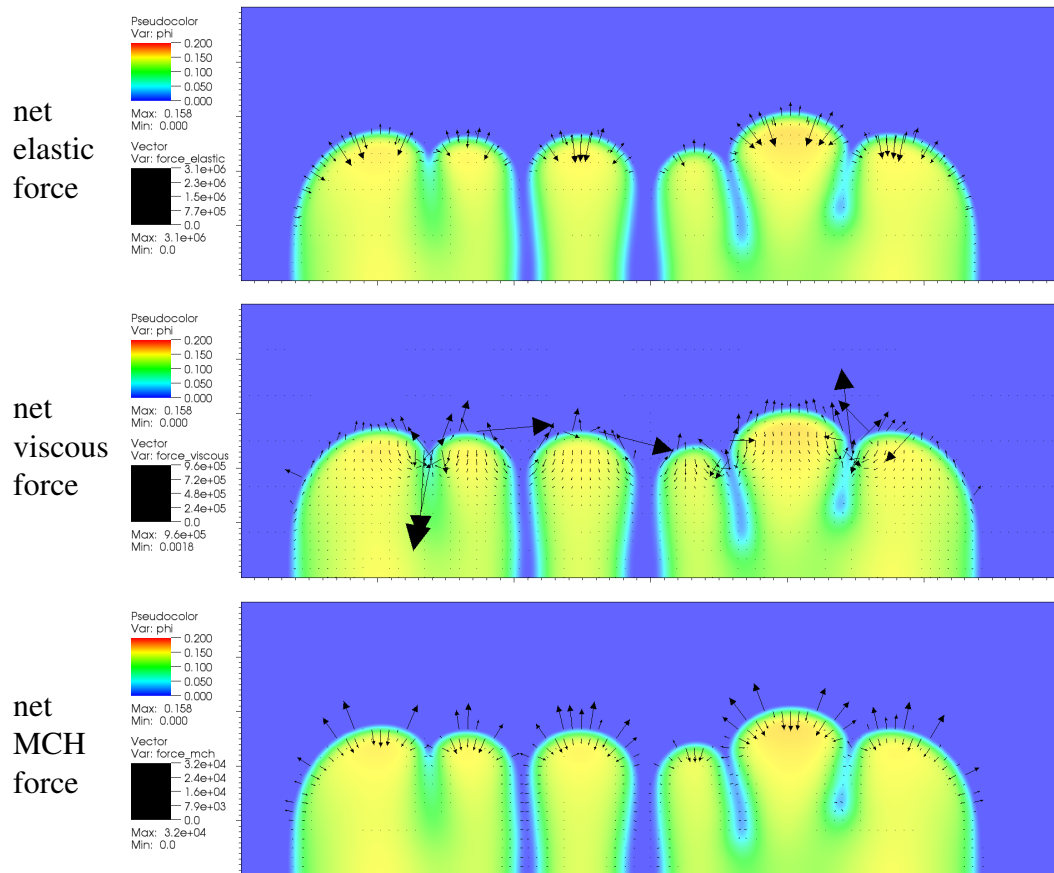


Figure 4.8: Net forces in a growing colony of biofilm. They are of the same order of magnitudes as in the case of growing a bud of biofilm. Both the net elastic force and the net interfacial force switch directions at the biofilm interface. The net elastic force tries to squeeze the interface thinner, while the net MCH force tries to pull the interface wider. The net viscous force does not show such a trend.

Time Viscous Giesekus

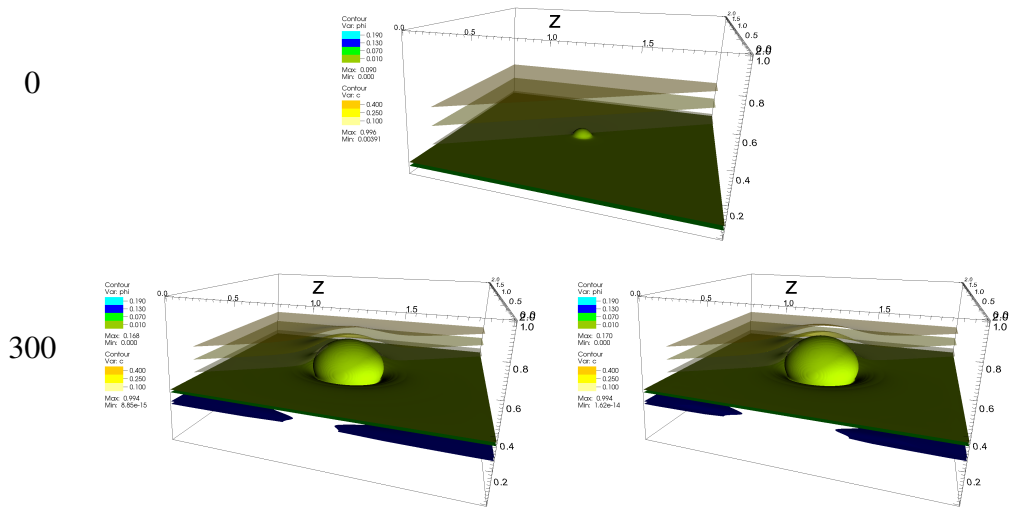


Figure 4.9: Growing a bud of biofilms in 3-D. The viscous and viscoelastic models yield similar results, with the biomass growing into a mushroom-shaped bud. The translucent layers show nutrient contour.

Time Viscous Giesekus

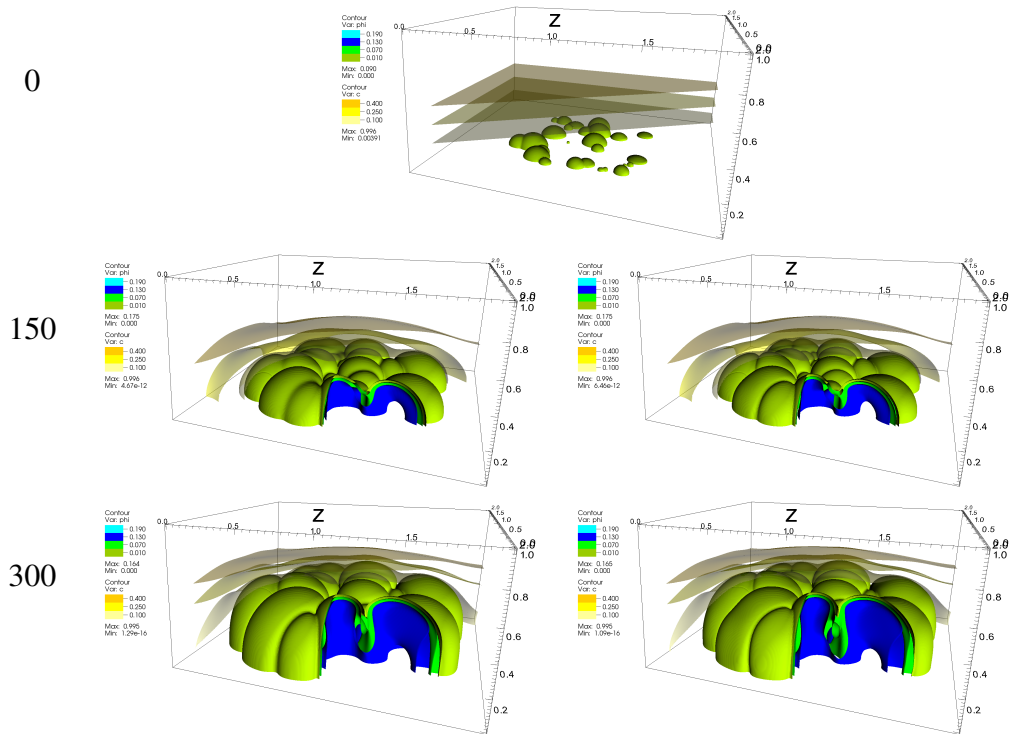


Figure 4.10: Snapshots of growing scattered bits of biofilms in 3-D

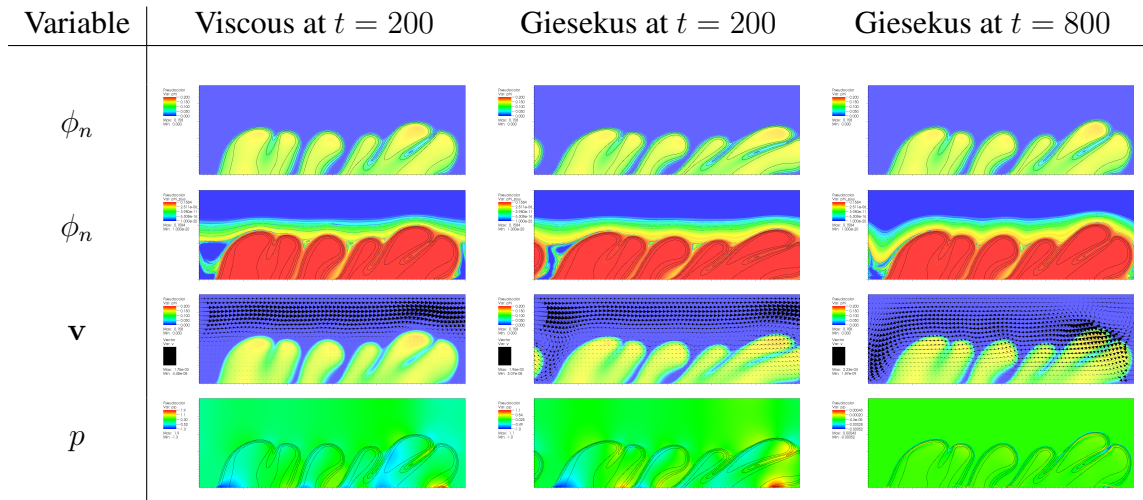


Figure 4.11: Biomass volume fraction ϕ_n , average velocity \mathbf{v} , and pressure p . The second row show log plots of ϕ_n , in which a streaming layer of faint biomass can be seen ($\phi_n \approx 10^{-6}$). The pressure concentrates mostly on the corners of biofilm bases.

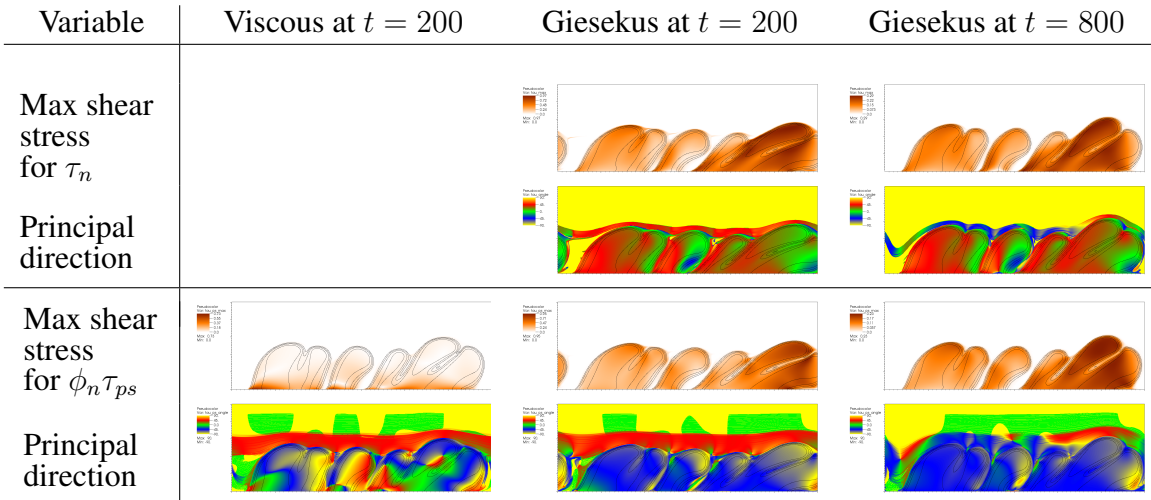


Figure 4.12: Elastic stress τ_n and bacterial viscous stress $\phi_n \tau_{ps}$. Please see Fig.4.4 for an explanation of the terms. The tallest bud has the highest elastic stress. In the principal direction plots, the stream line above the main biofilm profile is caused by the stream of faint biomass shown in Fig.4.11. In the presence of an imposed shear, the stress in this streaming biomass has the principal direction at 45° , which is the principal direction of the rate of strain tensor \mathbf{D} . In the viscous model, the viscous stress and the velocity diminish soon after the shear ceases. In viscoelastic models, during the biomass recoil, the region with high viscous stress coincides with the region with high elastic stress, while their principal directions are perpendicular to each other.

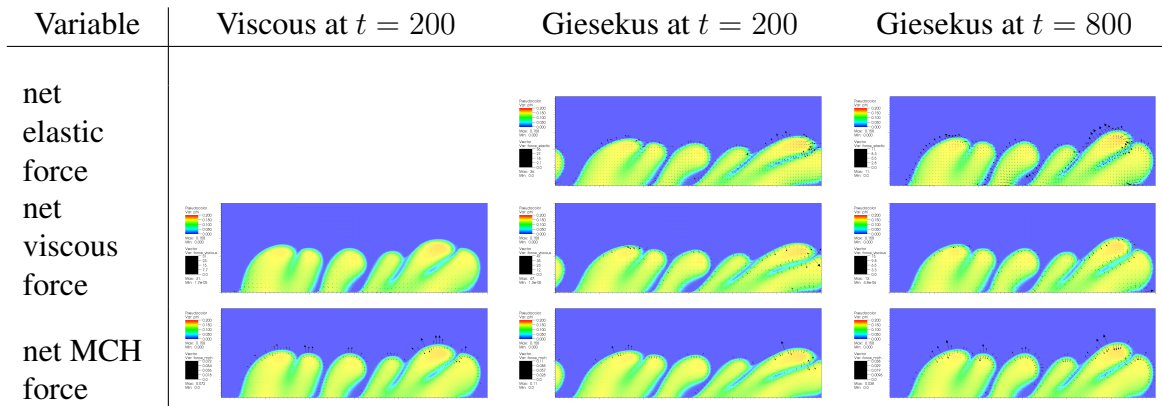


Figure 4.13: Forces in the momentum equation. The net elastic force and net viscous force are of comparable magnitudes. The net interfacial force is weaker by 2-3 orders of magnitude.

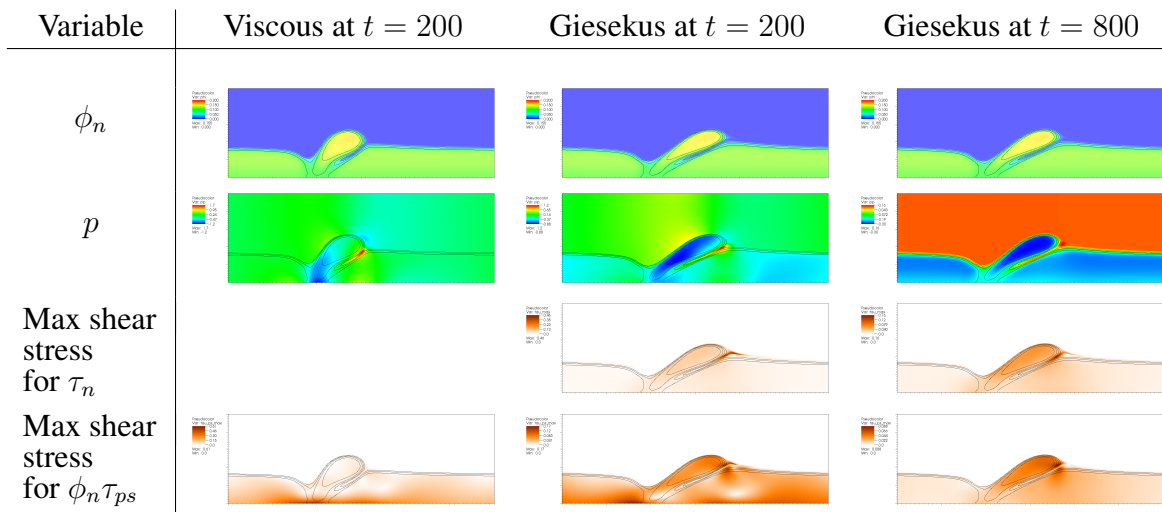


Figure 4.14: Shearing the biofilm grown from a small bud. The tip of the bud stretches and then rubs against the shoulder on the right. This creates a region of high stress, and thus also generate high pressure. The bud sticks to the shoulder and does not recoil back even in the viscoelastic model.

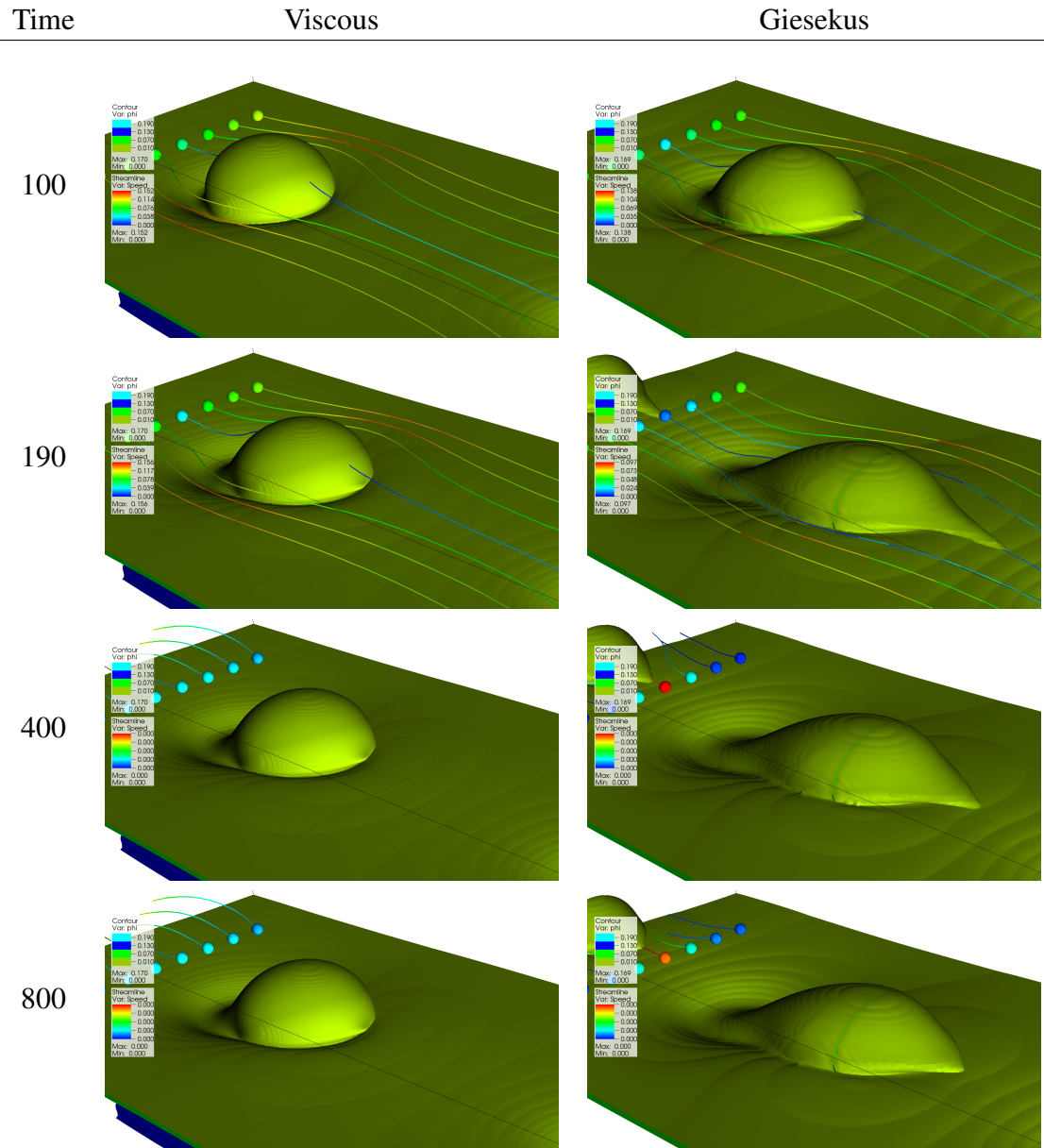


Figure 4.15: Shearing of a 3-D biofilm grown from a bud. The viscoelastic model predicts that a nose of the biomass extends out and streams along with the flow. This nose partially retracts back when we stop imposing the shear.

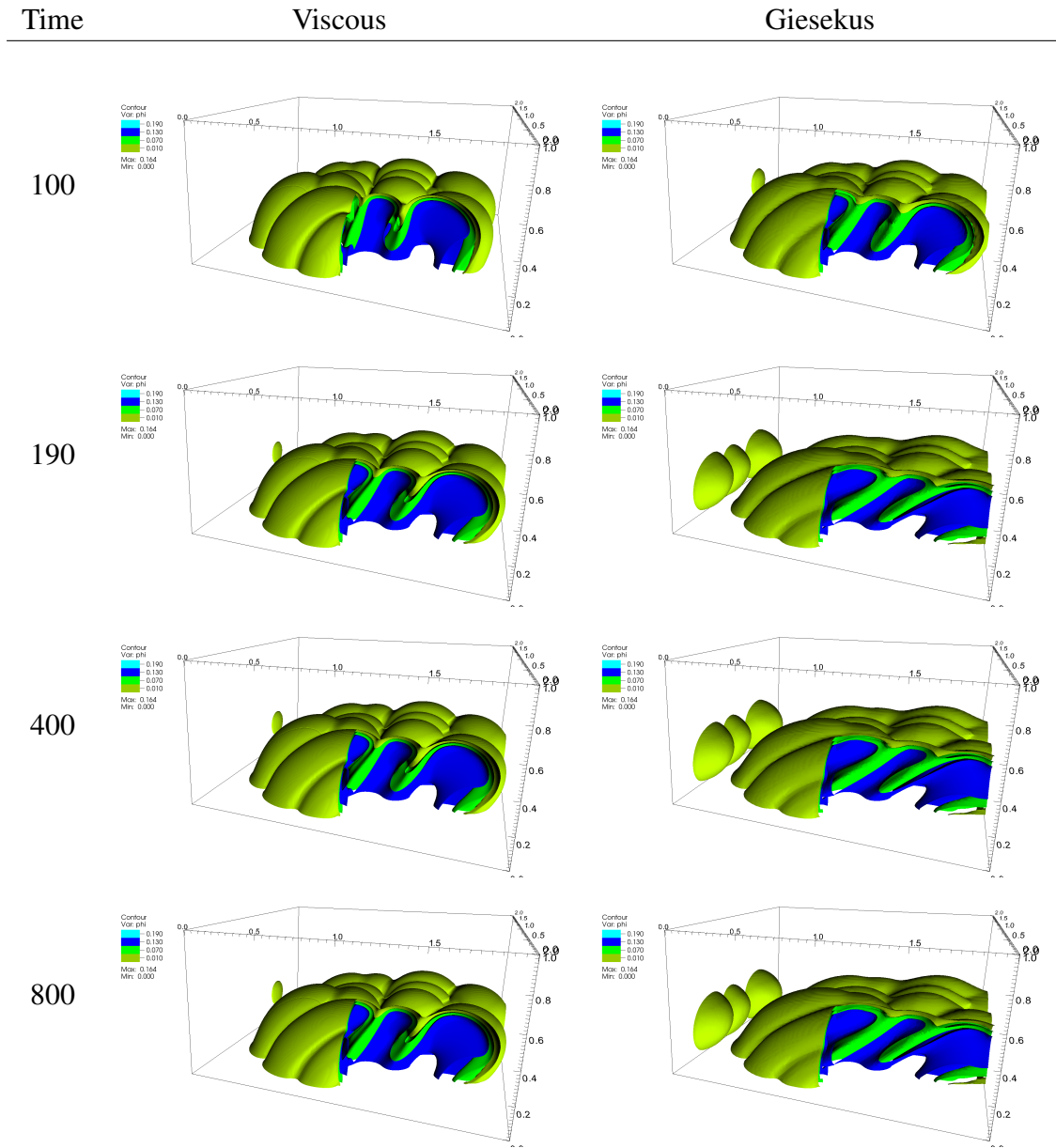


Figure 4.16: Shearing of a 3-D biofilm grown from scattered bits. The viscoelastic model predicts a biofilm that sways more under the flow than that in the viscous model. Interestingly, the viscoelastic biofilm does not recoil back because each lobe becomes stuck to the next lobe.

Time

Giesekus

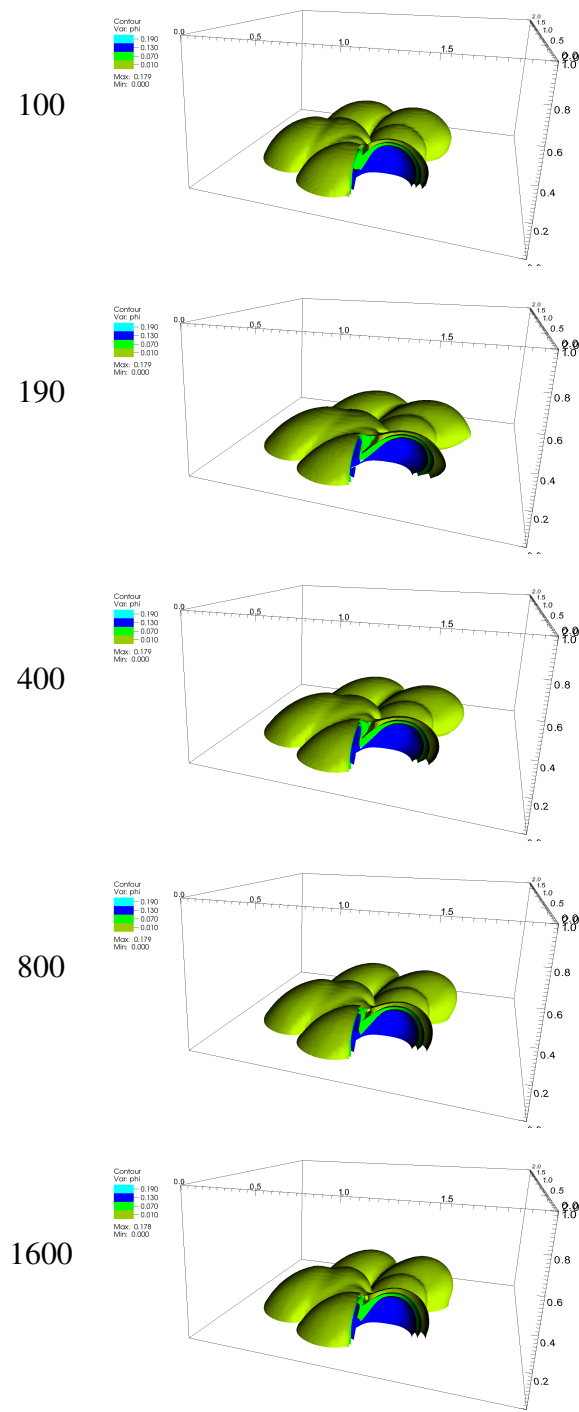


Figure 4.17: Shearing of a 3-D colony of biofilm with fewer buds. The colony contains fewer lobes, each of which is shorter than those in Fig.4.16, thus each lobe does not become stuck to its neighbor. The biofilm recoils back more than that in Fig.4.16.

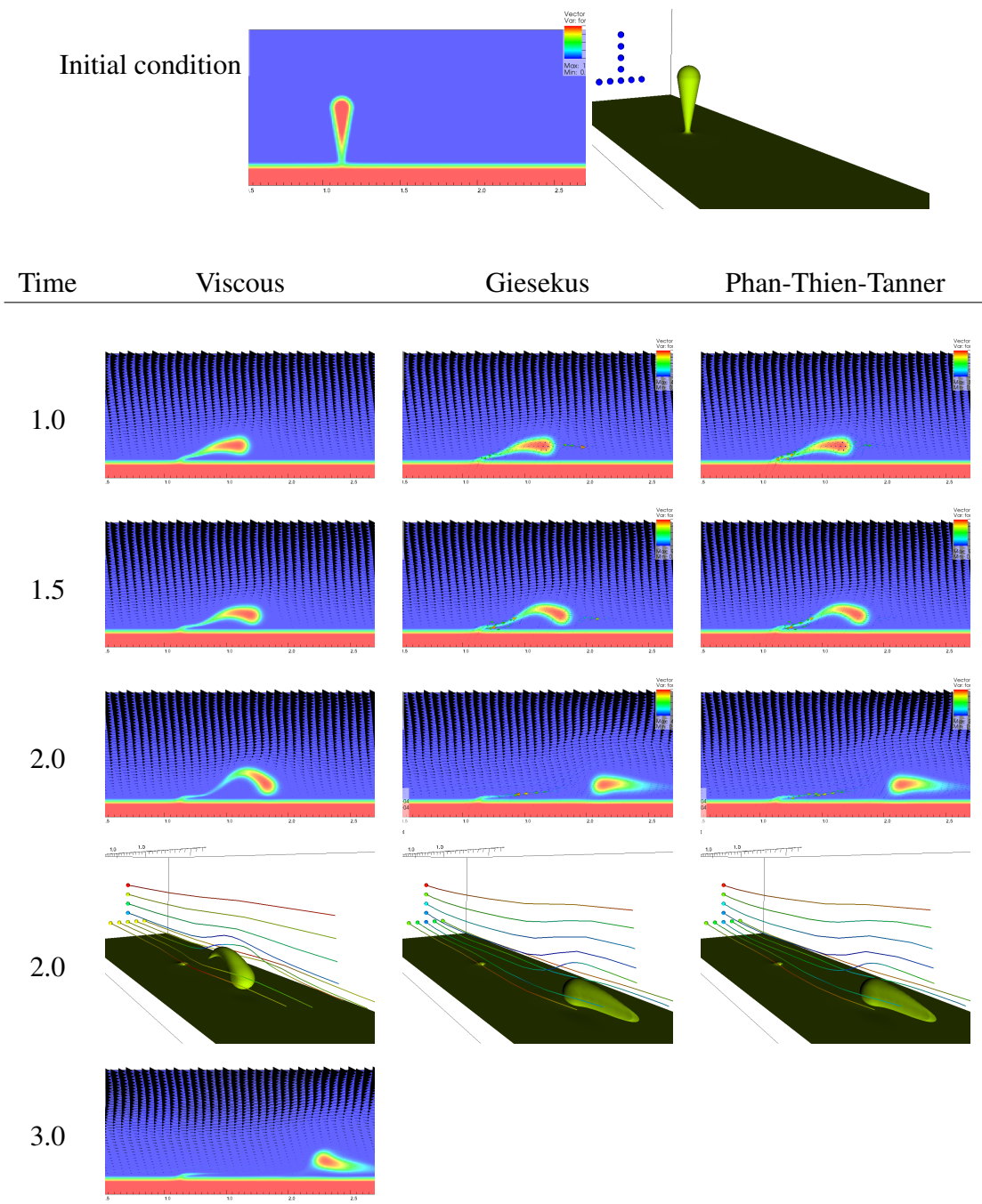


Figure 4.18: Shearing an biofilm lump with a thin neck. The biofilm colony detaches at high shear rates.

CHAPTER 5

CONCLUSION

We have extended the biofilm model from [75] [76] to account for viscoelasticity. The phase field method allows us to use one set of constitutive equations for the whole domain, easing the numerical implementation in resolving the biofilm-solution interface by allowing the biofilm-solution interface to be handled in the same way as the rest of the domain rather than imposing an interface tracking method. The model explores the biomass growth by nutrient consumption and solvent molecule penetration via fluid mixing. The biofilm and surrounding fluid flow interaction, transport through advection, dissipation, and reaction together with nutrient transport are studied in a channel flow geometry. EPS elasticity and its implication to the biofilm dynamics are investigated on vastly different timescales, spanning from seconds to days via 2-D and 3-D numerical solvers, developed for this purpose specifically.

We presented a numerical scheme for this model based on the finite difference method on the staggered grid. The Navier-Stokes equation is solved by the Gauge-Uzawa method, modified to take advantage of the fast Fourier transform and the interface coupling. The stress constitutive equation is converted into a difference form that can be quickly solved by a back interpolation followed by an explicit updating rule. The remaining equations are discretized by the implicit method and solved iteratively by the BiCG-stab method. The drastic change in viscosity from the biomass to the ambient solution fluid poses severe numerical difficulties in solving the Navier-Stokes equation and the viscoelastic constitutive equation accurately at a moderate time step size. We identified these issues and presented methods for mitigating or overcoming them. As a result, we have to settle with an overall

lower order scheme for the mixture fluid. A potential solution for this problem is to abandon the strategy of using FFT to solve the Helmholtz equation in the semidiscrete system. Instead, we can use an iterative solver to solve the linear system of variable coefficient with a choice of a good pre-conditioner.

We analyzed the discrete model's numerical properties and implemented the scheme on a hybrid CPU/GPU system. In order to efficiently utilize GPU's resources, we designed our code and data structure so that memory are accessed in a coalesced manner, and avoid the bottleneck of the CPU-GPU data transfer. We use this code to explore the process of biofilms growth and its dynamics under shear, in both two and three space dimensions.

Results from the viscoelastic model are compared to those from the viscous model. Both models predict similar results for growing biofilms, since this timescale is longer than the elastic relaxation time and the viscous effect dominates. Biofilms are sheared under a much shorter flow-induced time scale. The model predicts that the elastic stress offers less initial resistance to deformation than the viscous stress. This allows a part of the viscoelastic biofilm to elongate and stream along the shearing direction in 3-D simulations. Once we stop the shear, the viscoelastic model shows that the biofilm partially recoils back to its original shape, while the viscous-only model predicts that the biofilm simply stops moving.

We have used numerical computation to investigate and compare the predictions of a viscous and viscoelastic biofilm models. This methodology allows researchers to query and visualize variables in a model, which include the important physical quantities such as biomass volume fraction, nutrient concentration, and the fluid fields of the biofilm components, thus gaining a more intimate understanding of the model. We hope that this work provide a foundation on which more details and other constitutive equations can be added in, and then used to investigate other biofilm-related phenomena of interest such as cell migration and quorum sensing [33].

BIBLIOGRAPHY

- [1] N. Aravas and C.S. Laspidou, *On the calculation of the elastic modulus of a biofilm streamer*, *Biotechnology and Bioengineering* **101** (2008), no. 1, 196–200.
- [2] Miguel J. Bagajewicz and Han-Chin Wu, *Continuum mechanics and plasticity*, CRC Press, 2004.
- [3] Howard A. Barnes, John Fletcher Hutton, and Kenneth Walters, *An introduction to rheology*, Elsevier, 1989.
- [4] J.W. Bigger, *Treatment of staphylococcal infections with penicillin by intermittent sterilisation*, *The Lancet* (1944), 497–500.
- [5] R. Byron Bird, Robert C. Armstrong, and Ole Hassager, *Dynamics of polymeric liquids*, 2 ed., vol. 1, Wiley-Interscience, 1987.
- [6] J. Cadafalch, C. D. Pérez-Segarra, R. Cònsul, and A. Oliva, *Verification of finite volume computations on steady-state fluid flow and heat transfer*, *Journal of Fluids Engineering* **124** (2002), no. 1, 11.
- [7] Rabindranath Chatterjee, *Mathematical theory of continuum mechanics*, Alpha Science International, 1999.
- [8] Richard M. Christensen, *Theory of viscoelasticity*, 2 ed., Courier Dover Publications, 2003.
- [9] N. G. Cogan and James P. Keener, *The role of the biofilm matrix in structural development*, *Mathematical Medicine and Biology* **21** (2004), no. 2, 147–166.
- [10] _____, *Channel formation in gels*, *SIAM Journal on Applied Mathematics* **65** (2005), no. 6, 1839.
- [11] Alfred B. Cunningham, John E. Lennox, and Rockford J. Ross, *Biofilms: The hypertextbook*, http://biofilmbook.hypertextbookshop.com/public_version/, 2001–2011.

- [12] Jeffrey Dean and Sanjay Ghemawat, *Mapreduce : Simplified data processing on large clusters*, Communications of the ACM **51** (2008), no. 1, 107.
- [13] J. Dockery and I. Klapper, *Finger formation in biofilm layers*, SIAM J. Appl. Math **62** (2001), 853–869.
- [14] Masao Doi, *Introduction to polymer physics*, Oxford University Press, 1996.
- [15] Masao Doi and Samuel F. Edwards, *The theory of polymer dynamics*, Oxford University Press, 1988.
- [16] Hermann Eberl (ed.), *Mathematical modeling of biofilms*, IWA Publishing, 2006, Volume 18 of Scientific and Technical Report No. 18 Series.
- [17] Hermann J. Eberl, David F. Parker, and Mark C.M. van Loosdrecht, *A new deterministic spatio-temporal continuum model for biofilm development*, Journal of Theoretical Medicine **3** (2001), no. 3, 161.
- [18] James Glazier et al., *CompuCell3D*, <http://www.compuCell3d.org>.
- [19] Nathan Bell et al., *CUSP – a library for sparse linear algebra and graph computations on cuda*, <http://code.google.com/p/cusp-library/>.
- [20] Hans-Curt Flemming and Jost Wingender, *The biofilm matrix*, Nature Reviews Microbiology (2010), 623–633.
- [21] Sabine Ulrike Gerbersdorf, Thomas Jancke, Bernhard Westrich, and David M. Paterson, *Microbial stabilization of riverine sediments by extracellular polymeric substances*, Geobiology **6** (2008), no. 1, 57–69.
- [22] A Grillet, *Stability analysis of constitutive equations for polymer melts in viscometric flows*, Journal of Non-Newtonian Fluid Mechanics **103** (2002), no. 2-3, 221–250.
- [23] J Guermond, P Minev, and J Shen, *An overview of projection methods for incompressible flows*, Computer Methods in Applied Mechanics and Engineering **195** (2006), no. 44-47, 6011–6045.
- [24] Luanne Hall-Stoodley and Paul Stoodley, *Biofilm formation and dispersal and the transmission of human pathogens*, Trends in Microbiology **13** (2005), no. 1, 7–10.

- [25] Chang Dae Han, *Rheology and processing of polymeric materials*, vol. 1, Oxford University Press, 2007.
- [26] Danial N. Hohne, John G. Younger, and Michael J. Solomon, *Flexible microfluidic device for mechanical property characterization of soft viscoelastic solids such as bacterial biofilms*, *Langmuir* **25** (2009), no. 13, 7743–7751.
- [27] S. M. Hunt, E. M. Werner, B. Huang, M. A. Hamilton, and P. S. Stewart, *Hypothesis for the role of nutrient starvation in biofilm detachment*, *Applied and Environmental Microbiology* **70** (2004), no. 12, 7418–7425.
- [28] Stephen M. Hunt, Martin A. Hamilton, John T. Sears, Gary Harkin, and Jason Reno, *A computer investigation of chemically mediated detachment in bacterial biofilms*, *Microbiology* **149** (2003), no. 5, 1155–1163.
- [29] Irmgard Jager-Zurn and Meinhard Grubert, *Podostemaceae depend on sticky biofilms with respect to attachment to rocks in waterfalls*, *International Journal of Plant Sciences* **161** (2000), no. 4, 599–607.
- [30] *Journal of Fluids Engineering*, *Editorial policy statement on the control of numerical accuracy*.
- [31] Kitware, Sandia National Laboratories, and Los Alamos National Laboratory, *ParaView – an open-source, multi-platform data analysis and visualization application*, <http://www.paraview.org>.
- [32] I. Klapper, C. J. Rupp, R. Cargo, B. Purvedorj, and P. Stoodley, *Viscoelastic fluid description of bacterial biofilm material properties*, *Biotechnology and Bioengineering* **80** (2002), no. 3, 289–296.
- [33] Isaac Klapper and Jack Dockery, *Mathematical description of microbial biofilms*, *SIAM Review* **52** (2010), no. 2, 221–265.
- [34] V Körstgens, H-C Flemming, J Wingender, and W Borchard, *Influence of calcium ions on the mechanical properties of a model biofilm of mucoid pseudomonas aeruginosa*, *Water Science & Technology* **43** (2001), no. 6, 49–57.
- [35] Jan-Ulrich Krefta, Cristian Picioreanu, Julian W. T. Wimpenny, and Mark C. M. van Loosdrecht, *Individual-based modeling of biofilms*, *Microbiology* **147** (2001), 2897–2912.

- [36] Lawrence Livermore National Laboratory, *VisIt – a free interactive parallel visualization and graphical analysis tool*, <https://wci.llnl.gov/codes/visit/>.
- [37] K. Lewis, *Riddle of biofilm resistance*, *Antimicrobial Agents and Chemotherapy* **45** (2001), no. 4, 999–1007.
- [38] Brandon Lindley, Qi Wang, and Tianyu Zhang, *A multicomponent model for biofilm-drug interaction*, *Discrete and Continuous Dynamical Systems - Series B* **15** (2011), no. 2, 417–456.
- [39] M. Matsushita, F. Hiramatsu, N. Kobayashi, T. Ozawa, Y. Yamazaki, and T. Matsuyama, *Colony formation in bacteria: experiments and modeling*, *Biofilms* **1** (1999), no. 4, 305–317.
- [40] Don Monroe, *Looking for chinks in the armor of bacterial biofilms*, *PLoS Biology* **5** (2007), no. 11, e307.
- [41] E. Morgenroth, H.J. Eberl, M.C.M. van Loosdrecht, D.R. Noguera, G.E. Pizarro, C. Picioreanu, B.E. Rittmann, A.O. Schwarz, and O. Wanner, *Comparing biofilm models for a single species biofilm system*, *Water Science & Technology* **49** (2004), no. 11-12, 145–154.
- [42] Steven J. Norris, David L. Cox, and George M. Weinstock, *Biology of treponema pallidum: Correlation of functional activities with genome sequence data*, *Journal of Molecular Microbiology and Biotechnology* **3** (2001), no. 1, 37–62.
- [43] NVIDIA corporation, *CUDA C best practices guide*.
- [44] ———, *CUDA C programming guide*.
- [45] George O’Toole, Heidi B. Kaplan, and Roberto Kolter, *Biofilm formation as microbial development*, *Annual Review of Microbiology* **54** (2000), no. 1, 49–79.
- [46] C. Picioreanu, J.-U. Kreft, and M. C. M. van Loosdrecht, *Particle-based multidimensional multispecies biofilm model*, *Applied and Environmental Microbiology* **70** (2004), no. 5, 3024–3040.
- [47] C. Picioreanu, J.S. Vrouwenvelder, and M.C.M. van Loosdrecht, *Three-dimensional modeling of biofouling and fluid dynamics in feed spacer channels of membrane devices*, *Journal of Membrane Science* **345** (2009), no. 1-2, 340–354.

- [48] C. Picioranu, J. B. Xavier, and M. C. M. van Loosdrecht, *Advances in mathematical modeling of biofilm structure*, *Biofilms* **1** (2004), no. 4, 337–349.
- [49] Cristian Picioranu, Mark C.M. van Loosdrecht, and Joseph J. Heijnen, *Mathematical modeling of biofilm structure with a hybrid differential-discrete cellular automaton approach*, *Biotechnology and Bioengineering* **58** (1998), no. 1, 101–116.
- [50] ———, *A new combined differential-discrete cellular automaton approach for biofilm modeling: Application for growth in gel beads*, *Biotechnology and Bioengineering* **57** (1998), no. 6, 718–731.
- [51] G.E. Pizarro, C. Garcia, R. Moreno, and M. E. Sepulveda, *Two-dimensional cellular automaton model for mixed-culture biofilm*, *Water Science & Technology* **49** (2004), no. 11-12, 193–198.
- [52] Gonzalo Pizarro, David Griffeath, and Daniel R. Noguera, *Quantitative cellular automaton model for biofilms*, *Journal of Environmental Engineering* **127** (2001), no. 9, 782.
- [53] Nikodem J. Popławski, Abbas Shirinifard, Maciej Swat, and James A. Glazier, *Simulation of single-species bacterial-biofilm growth using the Glazier-Graner-Hogeweg model and the CompuCell3D modeling environment*, *Math Biosciences Eng.* **5** (2008), no. 2, 355–388.
- [54] Bruce E. Rittmann and Perry L. McCarty, *Model of steady-state-biofilm kinetics*, *Biotechnology and Bioengineering* **22** (1980), no. 11, 2343–2357.
- [55] Patrick J. Roache, *Verification and validation in computational science and engineering*, Hermosa Pub, 1998.
- [56] C. J. Rupp, C. A. Fux, and P. Stoodley, *Viscoelasticity of staphylococcus aureus biofilms in response to fluid shear allows resistance to detachment and facilitates rolling migration*, *Applied and Environmental Microbiology* **71** (2005), no. 4, 2175–2178.
- [57] T. Shaw, M. Winston, C. J. Rupp, I. Klapper, and P. Stoodley, *Commonality of elastic relaxation times in biofilms*, *Physical Review Letters* **93** (2004), no. 9, 098102.
- [58] Jie Shen and Xiaofeng Yang, *A phase-field model and its numerical approximation for two-phase incompressible flows with different densities and viscosities*, *SIAM J. on Scientific Computing* **32** (2010), 1159–1179.

- [59] Daniel O. Sordelli, M. Cristina Cerquetti, and Anne Morris Hooke, *Replication rate of pseudomonas aeruginosa in the murine lung*, *Infection and Immunity* **50** (1985), no. 2, 388–391.
- [60] Fred Stern, Robert V. Wilson, Hugh W. Coleman, and Eric G. Paterson, *Comprehensive approach to verification and validation of CFD simulation—part 1: Methodology and procedures*, *Journal of Fluids Engineering* **123** (2001), no. 4, 793.
- [61] P Stoodley, R Cargo, C J Rupp, S Wilson, and I Klapper, *Biofilm material properties as related to shear-induced deformation and detachment phenomena*, *Journal of Industrial Microbiology and Biotechnology* **29** (2002), no. 6, 361–367.
- [62] P. Stoodley, K. Sauer, D. G. Davies, and J. W. Costerton, *Biofilms as complex differentiated communities*, *Annual Review of Microbiology* **56** (2002), no. 1, 187–209.
- [63] Paul Stoodley, Zbigniew Lewandowski, John D. Boyle, and Hilary M. Lappin-Scott, *Structural deformation of bacterial biofilms caused by short-term fluctuations in fluid shear: An in situ investigation of biofilm rheology*, *Biotechnology and Bioengineering* **65** (1999), no. 1, 83–92.
- [64] John C. Strikwerda, *Finite difference schemes and partial differential equations*, 2 ed., SIAM, 2004.
- [65] Susan Tolman and Paul Meakin, *Off-lattice and hypercubic-lattice models for diffusion-limited aggregation in dimensionalities 2–8*, *Phys. Rev. A* **40** (1989), 428–437.
- [66] Gerard J. Tortora, Berdell R. Funke, and Christine L. Case, *Microbiology: An introduction*, 10 ed., Benjamin Cummings, 2009.
- [67] L. Vachova, V. Stovicek, O. Hlavacek, O. Chernyavskiy, L. Stepanek, L. Kubinova, and Z. Palkova, *Flo11p, drug efflux pumps, and the extracellular matrix cooperate to form biofilm yeast colonies*, *The Journal of Cell Biology* **194** (2011), no. 5, 679–687.
- [68] Qi Wang and Tianyu Zhang, *Review of mathematical models for biofilms*, *Solid State Communications* **150** (2010), no. 21-22, 1009–1022.
- [69] O. Wanner and W. Gujer, *A multispecies biofilm model*, *Biotechnology and Bioengineering* **28** (1986), no. 3, 314–328.
- [70] O. Wanner and E. Morgenroth, *Biofilm modeling with aquasim*, *Water Science & Technology* **49** (2004), no. 11-12, 137–144.

- [71] Mark Wheelis, *Principles of modern microbiology*, Jones & Bartlett, 2007.
- [72] T. Witten and L. Sander, *Diffusion-limited aggregation, a kinetic critical phenomenon*, Physical Review Letters **47** (1981), no. 19, 1400–1403.
- [73] L. Yang, J. A. J. Haagensen, L. Jelsbak, H. K. Johansen, C. Sternberg, N. Hoiby, and S. Molin, *In situ growth rates and biofilm development of pseudomonas aeruginosa populations in chronic lung infections*, Journal of Bacteriology **190** (2008), no. 8, 2767–2776.
- [74] Liang Yang, *Pseudomonas aeruginosa quorum-sensing – a factor in biofilm development, and an antipathogenic drug target*, Ph.D. thesis, Technical University of Denmark, Lyngby, Denmark, July 2009.
- [75] Tianyu Zhang, N. G. Cogan, and Qi Wang, *Phase field models for biofilms. I. theory and one-dimensional simulations*, SIAM Journal on Applied Mathematics **69** (2008), no. 3, 641.
- [76] _____, *Phase-field models for biofilms II. 2-D numerical simulations of biofilm-flow interaction*, Communications in Computational Physics **4** (2008), 72–101.

APPENDIX A

SOLVING THE HELMHOLTZ EQUATION BY DFT

We use the discrete Fourier transform (DFT) to solve the discretized form of the Helmholtz equation,

$$\nabla^2 u - \lambda u = g. \quad (\text{A.1})$$

Our numerical scheme for the biofilm simulation only uses the case where λ is zero or a small positive number. Note that the sign of λ is opposite of that in the standard Helmholtz equation that is used to solve a hyperbolic system. Our value λ comes from the Navier-Stokes equation, which is parabolic. Regardless, we implement the solver for the general case where λ can be almost any constant value. We start by listing some notations and facts,

- Let $C^N = \{u := (u_i)_{i \in \mathbb{Z}} \mid u_i \in \mathbb{C} \text{ and } u_i = u_{i+N} \text{ for all } i \in \mathbb{Z}\}$ denote the set of N -periodic complex-valued data.
- For any operator $T : C^N \rightarrow C^N$, we use the shorthand $T_n(u) := (T(u))_n$.
- DFT is an operator $C^N \rightarrow C^N$ defined by $\text{DFT}_k u := (\text{DFT } u)_k := \sum_{n=0}^{N-1} u_n e^{i2\pi nk/N}$.
- For $u, v \in C^N$, their circular convolutions is defined by $(u * v)_n := \sum_{m=0}^{N-1} u_m v_{n-m}$.
- The circular convolution theorem states that, $\text{DFT}_k(u * v) = (\text{DFT}_k u)(\text{DFT}_k v)$.

Periodic boundary

First we solve (A.1) for the case of periodic boundary. All other boundary conditions will be converted eventually to a periodic boundary problem. Let $T : C^N \rightarrow C^N$ be the discrete laplacian operator $T_n(u) = \frac{1}{h^2}(u_{n-1} - 2u_n + u_{n+1})$, and let $t \in C^N$ be $\frac{1}{h^2}(-2, 1, 0, 0, \dots, 0, 1)$. Thus, $Tu = t * u$. With $u, g \in C^N$, start at the discretized Helmholtz equation, we get,

$$Tu - \lambda u = g \quad (\text{A.2})$$

$$t * u - \lambda u_n = g_n \quad (\text{A.3})$$

$$(\text{DFT } t)(\text{DFT } u) - \lambda \text{DFT } u = \text{DFT } g \quad (\text{A.4})$$

$$(\text{DFT}_k t - \lambda)(\text{DFT}_k u) = \text{DFT}_k g \quad (\text{A.5})$$

$$(\text{DFT}_k u) = (\text{DFT}_k g) / (\text{DFT}_k t - \lambda) \quad (\text{A.6})$$

$$u = \text{DFT}^{-1} ((\text{DFT } g) / (\text{DFT } t - \lambda)). \quad (\text{A.7})$$

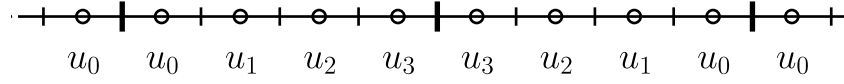
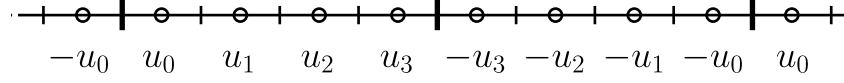
Note that,

$$\text{DFT}_k t = \sum_{n=0}^{N-1} t_n e^{i2\pi nk/N} \quad (\text{A.8})$$

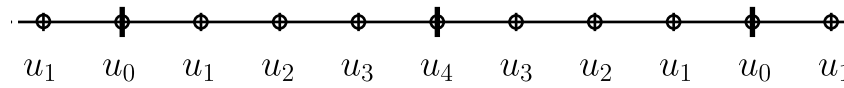
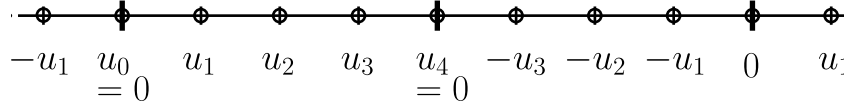
$$= \frac{1}{h^2} (-2 + e^{i2\pi k/N} + e^{-i2\pi k/N}) \quad (\text{A.9})$$

$$= \frac{1}{h^2} (-2 + 2 \cos(2\pi k/N)) \leq 0. \quad (\text{A.10})$$

Computing $\text{DFT}_k u$ in (A.6) requires a division by $\text{DFT}_k t - \lambda$, thus has a singularity when $\lambda = \text{DFT}_k t$. The biofilm simulation only uses the case where $\lambda \geq 0$, thus $\text{DFT}_k u$ is well-defined except for the case $\lambda = 0$ and $k = 0$. Instead of computing $\text{DFT}_0 u$ by (A.6), if we arbitrarily set $\text{DFT}_0 u = C$, the final answer u is off by constant $\bar{u}_n = u_n + C/N$. The only time the biofilm code encounters $\lambda = 0$ is when it solves the pressure Poisson equation (3.4), whose solution is defined only up to a shift by a constant. Therefore we arbitrarily set $\text{DFT}_0 u = 0$.



Staggered grid -- odd/even paddings



Regular grid -- odd/even paddings

Figure A.1: Odd/even extensions for grids with 4 data points

For a low wave number k , the term $1 - \cos(2\pi k/N)$ in (A.10) should not be computed directly since we would lose some numerical accuracy. We compute this term via the half angle formula $1 - \cos(x) = 2 \sin^2(x/2)$.

Homogeneous Dirichlet boundary

We now show how to solve (A.1) with the homogeneous Dirichlet boundary conditions. Given an N -point staggered grid with variables u_0, \dots, u_{N-1} , as shown in Fig.A.1, the boundary is located where $u_{-\frac{1}{2}}$ and $u_{N-\frac{1}{2}}$ would be. The homogeneous Dirichlet boundary condition can be satisfied to the second order at the boundary by using ghost nodes u_{-1} and u_N and solve,

$$\begin{cases} Tu_n - \lambda u_n = g_n & \text{for } n = 0, \dots, N-1 \\ \frac{1}{2}(u_{-1} + u_0) = 0 \\ \frac{1}{2}(u_{N-1} + u_N) = 0 \end{cases} \quad (\text{A.11})$$

This system has $N + 2$ variables and $N + 2$ equations. One can check that this is equivalent to solving the following system of $2N$ -periodic variables $u \in C^{2N}$,

$$\begin{cases} Tu_n - \lambda u_n = g_n & \text{for } n = 0, \dots, N - 1 \\ u_{2N-1-n} = -u_n & \text{for } n = 0, \dots, N - 1 \end{cases} \quad (\text{A.12})$$

This, in turn, is equivalent to setting $g_{2N-1-n} := -g_n$ for $n = 0, \dots, N - 1$ and then solve,

$$Tu_n - \lambda u_n = g_n, \quad \text{for } n = 0, \dots, 2N - 1. \quad (\text{A.13})$$

Thus, we can solve (A.11) by using odd extension and then treat the problem as a periodic boundary condition of period $2N$. Specifically, for any $u \in \mathbb{C}^N$, let $\bar{u} \in C^{2N}$ be its odd extension as shown in Fig.A.1. To solve the Helmholtz equation $Tu - \lambda u = g$, we start with the given $g \in \mathbb{C}^N$, then follow these steps,

1. Extend g into $\bar{g} \in C^{2N}$.
2. Solve $T^{2N}\bar{u} - \lambda\bar{u} = \bar{g}$ with the periodic boundary method.
3. Extract u from \bar{u} .

For the regular grid with N points, we need to solve this system of $N + 1$ equations,

$$\begin{cases} Tu_n - \lambda u_n = g_n & \text{for } n = 1, \dots, N - 1 \\ u_0 = 0 \\ u_N = 0 \end{cases} \quad (\text{A.14})$$

It is tempting to treat u as an N -periodic data. However, we won't be able to apply our algorithm from the periodic boundary case because we do not know the value of g_0 . Thus, we again resort to oddly extending u to $\bar{u} \in C^{2N}$, this time by $u_{2N-n} = -u_n$. This gives us the values of all \bar{g}_n . In particular, $\bar{g}_0 = \bar{g}_N = 0$ since $T^{2N}\bar{u} - \lambda\bar{u} = 0$ at these points.

If $\lambda = 0$, one must care to correctly set $\text{DFT}_0 \bar{u}$ to satisfy the homogeneous boundary condition. The correct choice is $\text{DFT}_0 \bar{u} = 0$. We note, however, that the biofilm simulation never make use of this case.

Homogeneous Neumann boundary

The homogeneous Neumann boundary condition can be handled similarly to the homogeneous Dirichlet boundary case, but using the even extension. On the staggered grid, the Neumann condition can be satisfied to the second order at the left boundary by $(u_0 - u_{-1})/h = 0$. Like before, this $N + 1$ system of equations is equivalent to using the even extension on C^{2N} . On the regular grid, the Neumann condition can be satisfied to the second order at the left boundary by $(u_1 - u_{-1})/(2h) = 0$. We can again use the even extension to C^{2N} . Unlike the regular grid Dirichlet case, the values for g_0 and g_N must be provided as inputs.

When $\lambda = 0$, we have the Poisson equation $Tu = g$, which needs to satisfy the discrete analogue of the divergence theorem,

$$\sum_{n=0}^{N-1} g_n = \sum_{n=0}^{N-1} T_n u = \sum_{n=0}^{N-1} \frac{1}{h^2} [(u_{n+1} - u_n) - (u_n - u_{n-1})] \quad (\text{A.15})$$

$$= \frac{1}{h^2} [(u_N - u_{N-1}) - (u_0 - u_{-1})] = 0 \quad (\text{A.16})$$

This is a constrain on the input g . It is the user's duty to make sure that the input satisfies this condition. The biofilm code faces this situation while solving the pressure Poisson equation (3.4). On the staggered grid, we have $g_n = (\nabla \cdot v)_n = v_{n+\frac{1}{2}} - v_{n-\frac{1}{2}}$. Thus, $\sum_{n=0}^{N-1} g_n = v_{N-\frac{1}{2}} - v_{-\frac{1}{2}} = 0$ as the result of either the periodicity of \mathbf{v} , or the homogeneous Dirichlet condition in \mathbf{v} 's momentum equations. Therefore, our input satisfies the discrete divergence theorem.

Homogeneous mixed boundary

Take as an example the problem with the Dirichlet condition prescribed on the left boundary, and Neumann on the right. We apply the odd extension on the left boundary, and the even extension on the right. This reduces the problem to that of the periodic boundary condition in C^{4N} .

Quadrupling the domain seems like a waste of computations. This is the price we pay for expressing our data in the FFT frequency basis. Our example of the mixed boundary condition problem on the regular grid has the eigenmodes u^k for $k = 0, 1, \dots, N - 1$ where $u_i^k = \sin\left(\frac{i(2k-1)\pi}{2N}\right)$. To capture all these modes, one needs to perform FFT on the domain of size $4N$.

Since the data we deal with are real-valued, and the extension step yields odd or even symmetries, we can exploit these structures in the data to speed up the calculation by using a discrete sine transform (DST), discrete cosine transform (DCT), or real-to-complex FFT. At the time of writing, there is no publicly available DCT or DST package for CUDA, so we use the FFT provided by the CUFFT package. The package is highly optimized by Nvidia, and is likely faster than any DST/DCT package we might implement.

Nonhomogeneous boundary

Nonhomogeneous Dirichlet or Neumann boundary condition problem can be converted into a homogeneous boundary problem. We give a concrete example, where the left boundary is prescribed the Dirichlet condition $u(0)$, while the right boundary is prescribed the Neumann condition $u'(1)$. Start with the discrete Helmholtz equations,

$$\begin{cases} T_n u - \lambda u_n = g_n \\ u_L = u(0) \\ u'_R = u'(1) \end{cases} \quad (\text{A.17})$$

where u_L denote an extrapolation of u_n to the left boundary location. For the staggered grid, one might use $u_L := \frac{3}{2}u_0 - \frac{1}{2}u_1$. On the regular grid, one can simply use $u_L := u_0$. Similarly, u'_R denotes the discretization of u' at the right boundary.

Let x_n denote the position of data point u_n . Write $u_n = \tilde{u}_n + \ddot{u}(x_n)$, where \ddot{u} is a function that satisfies the boundary conditions, $\ddot{u}(0) = u(0)$ and $\ddot{u}'(R) = u'(1)$. For

example, one may choose $\ddot{u}(x) = u(0) + u'(1)x$. We then have,

$$\begin{cases} T_n \tilde{u} + \nabla^2 \ddot{u}(x_n) - \lambda(\tilde{u}_n + \ddot{u}(x_n)) = g_n \\ \tilde{u}_L + \ddot{u}(0) = u(0) \\ \tilde{u}'_R + \ddot{u}'(1) = u'(1) \end{cases} \quad (\text{A.18})$$

Since \ddot{u} satisfies the boundary conditions, the system becomes,

$$\begin{cases} T_n \tilde{u} - \lambda \tilde{u} = g_n + \lambda \ddot{u}(x_n) - \nabla^2 \ddot{u}(x_n) \\ \tilde{u}_L = 0 \\ \tilde{u}'_R = 0 \end{cases} \quad (\text{A.19})$$

Thus, we have reduced the nonhomogeneous boundary condition problem into a homogeneous boundary problem.

The extra function \ddot{u} can often be chosen as a linear function, thus $\nabla^2 \ddot{u} = 0$. This reduces the required computations. The only exception is when both boundaries have the Neumann condition, in which case \ddot{u} can be a parabola.

A.2 TWO AND THREE DIMENSIONS

We can extend the aforementioned method to two and three dimensions. For example, in two dimension, (A.6) becomes,

$$(\text{DFT}_{j,k} u) = (\text{DFT}_{j,k} g) / (\text{DFT}_{j,k} t - \lambda) \quad (\text{A.20})$$

where $\text{DFT}_{j,k}$ denotes the two-dimensional DFT.

Periodic boundary condition is trivial to handle. Homogeneous boundary conditions remain homogeneous, which we then solve by the odd/even extension method. It is also easy to handle constant boundary conditions in one direction, such as $u(y = 0) = \text{const}_1$ and $u'(y = N) = \text{const}_2$, by using an extra function \ddot{u} like in the 1-D case. A more general boundary condition requires a more detailed handling.

A.3 VERIFICATION

We verify the code by performing grid refinements with forcing terms. Start with arbitrary twice differentiable functions $A, B, C \in C^2(\mathbb{R}^4)$. For example,

$$\begin{cases} A(t, x, y, z) = \sin(t^2)y^2(1 - y)^2 \sin(2\pi z) \\ B(t, x, y, z) = \sin(t)y(1 - y) \\ C(t, x, y, z) = \sin(2\pi x)y^2(1 - y)^2. \end{cases} \quad (\text{A.21})$$

The tuple (A, B, C) forms a 3-D vector field, thus its curl is divergence free. We use this curl as our velocity \mathbf{v} . We choose (A, B, C) such that \mathbf{v} obeys our boundary conditions: periodic in x, z , wall at $y = 0$, and shear at $y = 1$. We algebraically compute the force term, then use it as the input for the solver. The solver's output is compared to the exact solution \mathbf{v} .

We run one test with $\mathbf{v} = \text{curl}(A, 0, 0)$, which is a 2-dimensional flow in the y, z -plane. If the temporal term is too flat, the error due to time step size Δt can be too small to determine its temporal convergence behavior. The oscillation provided by the term $\sin(t^2)$ is reasonably demanding.

We also run a refinement test on a full 3-dimensional flow $\mathbf{v} = \text{curl}(A, B, C)$. The refinement results in Table A.1 and A.2 show the second order convergence rate in both space and time. The observed reduction rate decreases at the end of the temporal refinement table because the spatial truncation error starts to dominate. Vice versa.

Note that this refinement result is for flows with a constant viscosity. In the presence of a biofilm, the viscosity varies spatially by a factor of 10^5 . We use a modified numerical scheme described in Sec.3.2, which yields a different refinement result as shown in Sec.3.6.

N	$\mathbf{v} = \text{curl}(A, 0, 0)$				$\mathbf{v} = \text{curl}(A, B, C)$			
	$\ err\ _2$	rate	$\ err\ _\infty$	rate	$\ err\ _2$	rate	$\ err\ _\infty$	rate
2	5e-17	–	10e-17	–	6e-17	–	2e-16	–
4	36.40e-3	–	66.98e-3	–	29.75e-2	–	111.78e-3	–
8	9.21e-3	3.95	21.39e-3	3.13	7.37e-3	4.04	28.74e-3	3.89
16	2.38e-3	3.87	5.73e-3	3.74	1.90e-3	3.89	7.65e-3	3.75
32	.60e-3	3.95	1.46e-3	3.93	.48e-4	3.95	1.94e-3	3.94
64	.15e-3	3.99	.37e-3	3.98	.12e-3	3.99	.49e-3	3.99
128	37.83e-6	4.00	91.65e-6	4.00	30.15e-6	4.00	.12e-3	4.00
256*	9.45e-6	4.00	22.90e-6	4.00	7.78e-6	3.88	31.41e-6	3.88
512	2.36e-6	4.01	5.70e-6	4.02				
1024	.60e-6	3.94	1.64e-6	3.48				

Table A.1: Spatial refinement of the Navier-Stokes solver on N^2 and N^3 grids with $\Delta t = 1/1024$, $t_0 = 1$ sec, $t_{end} = 4$. The extremely small error at $N = 2$ is due to the symmetry in our problem. At $N = 1024$, the temporal truncation error starts to be significant. Note (*): For the 3-D problem, the maximum grid size is 252^3 .

Δt	$\mathbf{v} = \text{curl}(A, 0, 0)$ on 1024^2 grid				$\mathbf{v} = \text{curl}(A, B, C)$ on 252^3 grid			
	$\ err\ _2$	rate	$\ err\ _\infty$	rate	$\ err\ _2$	rate	$\ err\ _\infty$	rate
1/2	19.87e-3	–	60.65e-3	–	11.51e-3	–	60.73e-3	–
1/4	7.65e-3	2.60	40.00e-3	1.52	4.48e-3	2.57	40.14e-3	1.51
1/8	2.31e-3	3.32	12.54e-3	3.19	1.35e-3	3.32	12.58e-3	3.19
1/16	.59e-3	3.90	3.94e-3	3.18	.35e-3	3.90	3.94e-3	3.19
1/32	.15e-3	3.91	1.06e-3	3.71	88.08e-6	3.92	1.06e-3	3.72
1/64	38.15e-6	3.97	.27e-3	3.91	23.01e-6	3.83	.27e-3	3.94
1/128	9.54e-6	4.00	68.14e-6	3.98	9.26e-6	2.49	70.44e-6	3.82
1/256	2.42e-6	3.95	16.92e-6	4.03	7.82e-6	1.18	33.70e-6	2.09
1/512	.81e-6	3.00	4.25e-6	3.98	7.77e-6	1.01	31.34e-6	1.08
1/1024	.60e-6	1.35	1.64e-6	2.60	7.78e-6	1.00	31.41e-6	1.00
1/2048	.59e-6	1.01	1.43e-6	1.15				

Table A.2: Temporal refinement of the Navier-Stokes solver with $t_0 = 1$ sec and $t_{end} = 4$. The grid size is 1024^2 for the 2-D problem and 252^3 for the 3-D problem. Errors in the last few lines are dominated by the spatial truncation errors.

APPENDIX B

GRID REFINEMENT ANALYSIS

Given the limitation of our computing capacity, we currently can compute up to only about 240^3 grid points for the viscoelastic model, and 256^3 grid points for the viscous model in 3-D. This poses a challenge for us to conduct an extensive 3-D spatial refinement test. If we refine the mesh by halving the grid sizes, then the coarse grids might be too coarse to show a clear order of convergence. We thus discuss a practical testing methodology in this appendix.

B.1 CONVERGENCE RATE

When an exact solution is known, performing a grid refinement analysis is trivial. Say we want to compute a solution u of an equation system using a numerical scheme of spatial order p . Let u_0 be the exact solution, and u_h be the computed result at grid spacing h . We have,

$$u_h = u_0 + \epsilon_h \quad \text{where } \epsilon_h = Ch^p + \text{higher order terms.} \quad (\text{B.1})$$

Thus $\log \epsilon_h \approx p \log(h) + C$. We can plot $\log \epsilon_h$ against $\log h$ and use its slope as the observed order p . If we compute u at grid spacings h_1 and h_2 , then

$$p \approx \frac{\log((u_{h_2} - u_0)/(u_{h_1} - u_0))}{\log(h_2/h_1)}. \quad (\text{B.2})$$

When the exact solution is not available, however, we compute u at three grid spacings $h_1 < h_2 < h_3$. We can use the solution obtained at the mesh size h_1 as the reference solution, assuming $u_{h_1} \approx u_0$, then applying the previous formula using h_2 and h_3 yields a p . However, this can yield a misleading result. For example, assume that we have a

first order scheme and a numerical solution $u_h = u_0 + Ch$, and use the grid spacings $h_1, h_2, h_3 = 1/64, 1/128, 1/256$. Going from $u_{1/64}$ to $u_{1/128}$, the observed error will reduce by 3 folds: $\frac{u_{h_2} - u_{h_3}}{u_{h_1} - u_{h_3}} = \frac{\epsilon_{h_2} - \epsilon_{h_3}}{\epsilon_{h_1} - \epsilon_{h_3}} = (\frac{1}{64} - \frac{1}{256}) / (\frac{1}{128} - \frac{1}{256}) = 3$. The rate formula yields $p = \frac{\log 3}{\log 2} = 1.58$, which overstates the true convergence rate $p = 1$.

Let's start over from (B.1). If the grid spacings $h_1 < h_2 < h_3$ form a geometric progression ($r := h_3/h_2 = h_2/h_1$), then we have,

$$\frac{\epsilon_{32}}{\epsilon_{21}} := \frac{u_{h_3} - u_{h_2}}{u_{h_2} - u_{h_1}} \approx \frac{C(h_3^p - h_2^p)}{C(h_2^p - h_1^p)} = \left(\frac{h_2}{h_1}\right)^p \frac{\left(\frac{h_3}{h_2}\right)^p - 1}{\left(\frac{h_2}{h_1}\right)^p - 1} = \left(\frac{h_2}{h_1}\right)^p = r^p. \quad (\text{B.3})$$

Thus,

$$p \approx \frac{\log(\epsilon_{32}/\epsilon_{21})}{\log(r)} = \frac{\log((u_{h_3} - u_{h_2})/(u_{h_2} - u_{h_1}))}{\log(h_2/h_1)}. \quad (\text{B.4})$$

Comparing this to (B.2), the moral is: in order to compute the convergence rate when the exact solution is not known, the results should *not* be compared to the finest grid. It should be compared to the next grid size. This formula yields the accurate rate for the above example.

B.2 GLOBAL ERROR

In the finite difference method (FDM), a grid Ω_h consists of M points. For uniform grids, any reasonable definition of $\|\cdot\|_{k,\Omega_h}$ agrees with the following definition up to the second order,

$$\|f\|_{k,\Omega_h} = \left(\sum_{x \in \Omega_h} (\omega(x)f(x))^k \right)^{1/k} \quad (\text{B.5})$$

where $\omega(x)$ is the weight of each point in FDM. For a uniform grid, we typically have $\omega(x) = h^{\text{dimension}}$, perhaps with a factor of $\frac{1}{2}$, $\frac{1}{4}$ or $\frac{1}{8}$ for points on the boundaries.

For each $x \in \Omega_h$, we have the truncation error estimate $u_h(x) = u_0(x) + C(x)h^p + \dots$.

The local (pointwise) errors $(u_h - u_0)(x)$ and the global errors $\|u_h - u_0\|_{k,\Omega_h}$ are related,

$$\frac{\|u_{h_3} - u_{h_2}\|_{k,\Omega_h}}{\|u_{h_2} - u_{h_1}\|_{k,\Omega_h}} \approx \frac{\left(\sum_{x \in \Omega_h} (\omega(x)C(x)(h_3^p - h_2^p))^k\right)^{1/k}}{\left(\sum_{x \in \Omega_h} (\omega(x)C(x)(h_2^p - h_1^p))^k\right)^{1/k}} \quad (\text{B.6})$$

$$= \frac{\|C\|_{k,\Omega_h} (h_3^p - h_2^p)}{\|C\|_{k,\Omega_h} (h_2^p - h_1^p)} = \frac{(h_3^p - h_2^p)}{(h_2^p - h_1^p)} = r^p. \quad (\text{B.7})$$

This formula for the global convergence rate is the same as the one for local convergence (B.2). One might even say that this should be obvious, since we can consider $\|u_h - u_0\|_k$ as just another scalar value. It is worth noting that $\|C\|_{k,\Omega_h}$ is grid dependent. Back to the example $h_1, h_2, h_3 = 1/64, 1/128, 1/256$. It is often convenient to compute $\|u_{1/64} - u_{1/128}\|$ on the $1/64$ grid and compute $\|u_{1/128} - u_{1/256}\|$ on the $1/128$ grid. However, this will introduce a small error in the refinement result since the value $\|C\|_{\Omega_{1/64}} / \|C\|_{\Omega_{1/128}}$ in (B.7) is not exactly one. The problem subsides at very fine grids, as $\|C\|_{k,\Omega_h}$ converges to the continuum $\|C\|_k$.

If we consider the next higher order term, we have $u_h(x) = u_0(x) + C(x)h^p + D(x)h^q + \dots$. Thus,

$$\frac{\|u_{h_3} - u_{h_2}\|}{\|u_{h_2} - u_{h_1}\|} \approx \frac{\|C(h_3^p - h_2^p) + D(h_3^q - h_2^q)\|}{\|C(h_2^p - h_1^p) + D(h_2^q - h_1^q)\|} = \frac{\|C + D((h_3^q - h_2^q)/(h_3^p - h_2^p))\| (h_3^p - h_2^p)}{\|C + D((h_2^q - h_1^q)/(h_2^p - h_1^p))\| (h_2^p - h_1^p)}$$

Assuming $r := h_3/h_2 = h_2/h_1$, we get,

$$\frac{\|u_{h_3} - u_{h_2}\|}{\|u_{h_2} - u_{h_1}\|} = \frac{\|C + Dr^{q-p}(r^q - 1)/(r^p - 1)\|}{\|C + D(r^q - 1)/(r^p - 1)\|} r^p$$

The two norms no longer cancel exactly. Thus we do not have a nice formula for the convergence order.

B.3 NONCONSTANT REFINEMENT FACTORS

One occasionally needs to perform a mesh refinement on grid spacings $h_1 < h_2 < h_3$ which do not form a geometric series. Let $r_{ij} = h_i/h_j$. We can solve one of the following equations for p ,

$$\frac{\epsilon_{31}}{\epsilon_{21}} := \frac{u_{h_3} - u_{h_1}}{u_{h_2} - u_{h_1}} \approx \frac{h_3^p - h_1^p}{h_2^p - h_1^p} = \frac{\left(\frac{h_3}{h_1}\right)^p - 1}{\left(\frac{h_2}{h_1}\right)^p - 1} = \frac{r_{31}^p - 1}{r_{21}^p - 1} \quad (\text{B.8})$$

or

$$\frac{\epsilon_{32}}{\epsilon_{21}} := \frac{u_{h_3} - u_{h_2}}{u_{h_2} - u_{h_1}} \approx \frac{h_3^p - h_2^p}{h_2^p - h_1^p} = \left(\frac{h_2}{h_1}\right)^p \frac{\left(\frac{h_3}{h_2}\right)^p - 1}{\left(\frac{h_2}{h_1}\right)^p - 1} = r_{21}^p \frac{r_{32}^p - 1}{r_{21}^p - 1}. \quad (\text{B.9})$$

The solution p exists for (B.8) iff $\epsilon_{31}/\epsilon_{21} > 1$ and is positive iff $\epsilon_{31}/\epsilon_{21} > \log(\frac{h_3}{h_1})/\log(\frac{h_2}{h_1}) = \log_{r_{21}} r_{31}$. The solution for (B.9) exists iff $\epsilon_{32}/\epsilon_{21} > 0$, and is positive iff $\epsilon_{32}/\epsilon_{21} > \log_{r_{21}} r_{32}$. Among these two formula, the latter one is more prevalent in CFD literature.

The CFD community has worked out a more advanced grid refinement methodology based on Richardson extrapolation (RE) and Grid Convergence Index (GCI). Some useful references can be found in [30] [6] [60] [55].