Spring 1-1-2015

Simulations and Experiments for Fouling Mitigation on Patterned Nano-Imprint Lithography Ultra Filtration Membranes

John Mersch IV
University of Colorado Boulder, jmerschiv@gmail.com

SIMULATIONS AND EXPERIMENTS FOR FOULING MITIGATION

ON PATTERNED NANO-IMPRINT LITHOGRAPHY ULTRA FILTRATION MEMBRANES

By

John Mersch IV

B.A., University of California, Berkeley, 2012

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Masters of Science

Department of Mechanical Engineering

2015

This thesis entitled:
Simulations and experiments for fouling mitigation on patterned nano-imprint lithography ultra-filtration membranes
written by John Mersch IV
has been approved for the Department of Mechanical Engineering

_____
John Pellegrino

_____
Peter Hamlington

*Date*_____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
(This statement must be included on the signature page)

IRB protocol # _____
(Must be included if your research involved the use of human subjects)

IACUC protocol # _____

Abstract

Mersch IV, John (M.S., Mechanical Engineering)

Simulations and experiments for fouling mitigation on patterned nano-imprint lithography ultra-filtration membranes

Thesis directed by Professor John Pellegrino

Nano Imprint Lithography (NIL) endows Ultra Filtration (UF) membranes with a plethora of filtration benefits. This research is to study for purposes of optimization the fundamental physics behind the fouling mitigation properties derived from the NIL patterns on UF membranes. Simple silicon particle experiments have been performed initially and have generated a series of criteria believed to affect the fouling mitigation with patterned membranes. These factors are: pattern height, permeation rate, cross flow velocity, and angle of attack. Factors from literature that affect these criteria are shear rate, gradient of shear rate and the non-linearity parameter, which is a ratio of the shear rate and it's gradient. My work covers using computation fluid dynamic (CFD) simulations in an attempt to understand the underlying physics involved. Furthermore, I perform milk filtration experiments to test the NIL UF membrane's capabilities to handle complex fluids. Milk is one of the most readily fouling substances that are filtered with UF membranes commercially. The hypothesis put forth and substantiated here follows the principles that it is a combination of non-linearity parameter and gradient of the shear rate that contribute most significantly to the fluid dynamic aspect of fouling reduction through the mechanism of shear induced diffusion.

Acknowledgements

The author wishes to thank Professor Thomas Hauser who taught me the fundamentals of the meshing tool GMSH and who provided a lot of guidance and advice with the program OpenFOAM. I'd also like to thank him for all the technical support and strings he pulled so that I could run the simulations on Janus in a timely fashion. I would also like to thank Timothy Dunn for his constant help and support with Janus run errors. I am grateful that Sajjad Maruf performed the experiments and made the patterned membranes for us to use in experiments. Most importantly I would like to thank John Pellegrino, Thomas Hauser, and Peter Hamlington for being very patient with all the errors and mistakes I made along the way.

CONTENTS

CHAPTER

TABLES

Table

FIGURES

Figure

CHAPTER I

Arrangement of Thesis

Chapter 2

Details the background information necessary to have familiarity with the experimental subject matters. It contains summary information on ultrafiltration (UF) membranes, fouling and cleaning, history of patterned membranes, computational fluid dynamics (CFD), milk filtration and the reference experiments on which the simulations are based. The background on the UF membranes will cover membrane making techniques, cost, uses, lifetimes, modes of operation, as well as how membranes operate. Fouling will consist of an explanation into the four types of fouling that can occur while cleaning will focus briefly on cleaning theory and then list examples of cleaning methods to emphasize how costly and time intensive they can be. The history of the membrane will consist of a brief timeline of patterning technology, with a focus on some examples of how micron and nano sized patterns are made for both on and off membranes. The CFD background will consist of general process, the three solving algorithms, meshing, and specific OpenFOAM files. The milk experimental background will consist of a review of literature regarding milk in industry, standard cleaning protocols, and our modified cleaning protocols along with some statistics about use and cost. Lastly the experiments and results achieved by Maruf whose experimental models we simulated will be set forth as the guideline for what we aim to achieve through experimentation.

Chapter 3

While there were a huge number (around 600) simulations completed, the vast majority of those will not be covered due to the errors in the simulations or the lack of their relevancy other than as a learning tool. To cover them all would be well beyond the scope of the thesis as

they only contained personal learning or mechanisms of a poorly documented program. What this section will contain are the three solver validation simulations, the jet flow simulations and the cross flow simulations. The solver validations simulations consist of three standard analytic solution cases: moving top wall, slit flow, and cavity flow. The jet flow cell consists of two regimes. The first is the whole bulk flow cell where everything is calculated; the second is the nano scale pattern as a small piece of the bulk flow system using the bulk flow for boundary conditions. The cross flow follows the same set up with a bulk flow section and a flow over imprint section.

Chapter 4

This chapter will feel incomplete by nature, as the experimental work was not finished either. Due to the incomplete nature of the experiments, the first section will focus on the protocols I developed. The second section will go over each set of experimental results even though the results may be inconclusive.

Chapter 5

This brief chapter will summarize the results that were achieved in the experimental sections. It will end with a proposal of future work topics and ideas.

Chapter 2

Background Information

Ultrafiltration Membranes

Membranes are semipermeable barriers that allow certain particles through while rejecting others. Some membranes involve active transport systems such as those used in cell membranes. Other membranes, such as those we will be concerned with, use a system of size exclusion through the use of pores to reject particles. Of this type there are sometimes additional features, which allow for other types of selectivity including charge and polarization. Depending upon pore sizes membranes can be classified as microfiltration (MF), ultrafiltration (UF), nano-filtration (NF), and reverse osmosis (RO) (see Figure 1). The membranes used in our labs and experiments are UF membranes and so I will focus on those for the background, although most of the general information applies to membranes as a whole. There are four main type of membrane configurations: spiral wound membranes, tubular membranes, hollow fiber membranes and plate and frame membranes [1]. Our lab uses a combination of plate and frame membranes and tubular membranes. However, for the work I did, only sheet membranes were used. Spiral wound membranes consists of flexible sheets of membrane, spacer and inlet feeds rolled together forming a spiral. Tubular membranes are single layer membranes forming a circle with structure support underneath. They can run with complicated solutions due to their ease of being cleaned and the high shear rate that can be generated. Tubular membranes are used in the dairy industry for concentrating milk to be used in cheese making. Hollow fiber membranes really maximize surface area but work best for low viscous fluids. Sheet membranes as we use them in the lab are single layer flat membranes held by plates as a single physical barrier. They are the easiest to do experimental work on. Due to their large use of space, sheet

membranes configurations are typically not used in industry. Figure 2 shows the three commercial types of membranes and their filtration process. Figure 3 shows an actual cross section of an UF membrane.

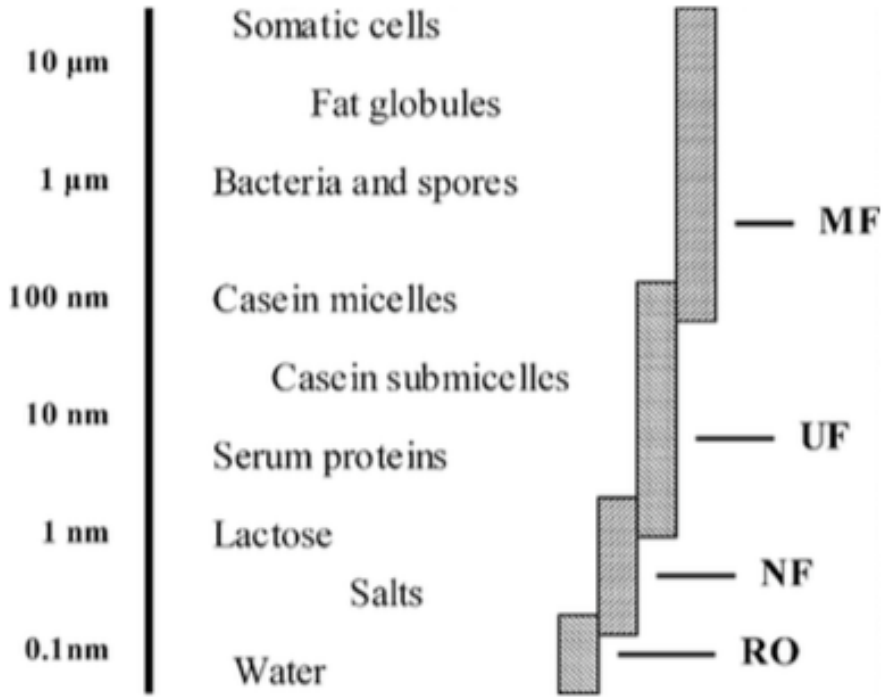
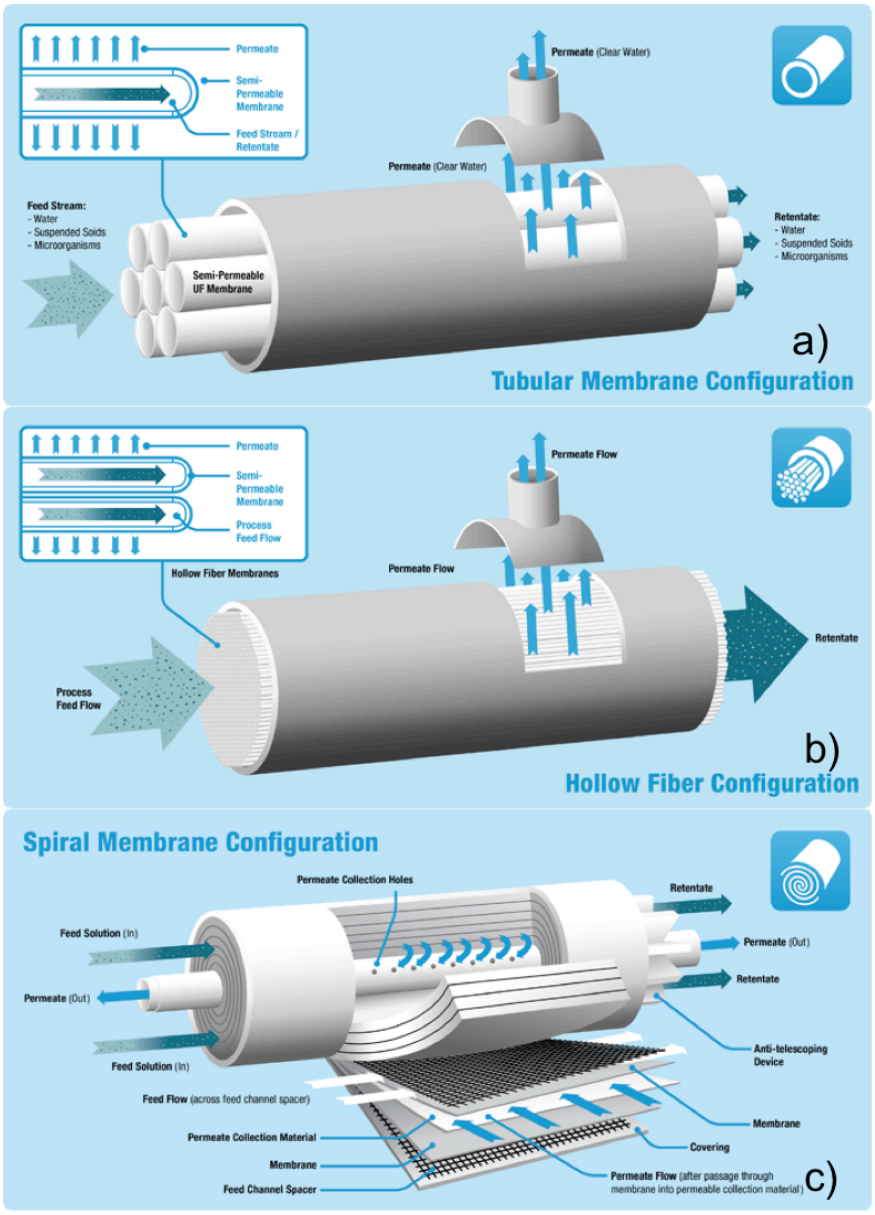


Figure 1: Classification of membranes by pore size and common foulants or permeates [2].



[1]

Figure 2: Displays the three commercial types of membrane configurations a) tubular, b) hollow fiber, and c) spiral wound.

[1] http://www.kochmembrane.com/Learning-Center/Configurations.aspx

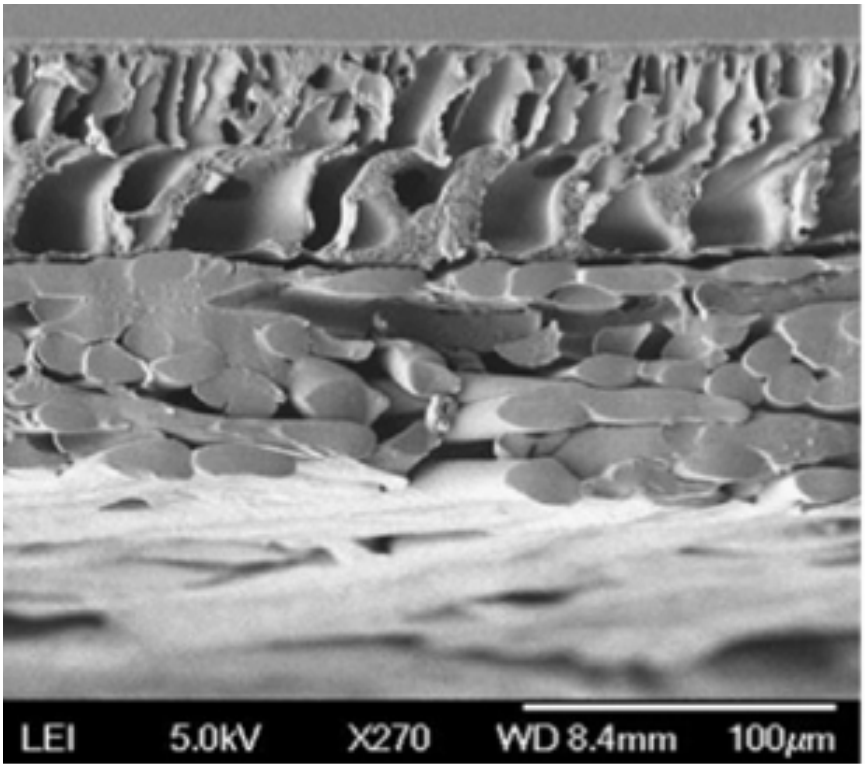


Figure 3: Cross section SEM image of polyethersulfone (PES) UF membrane [3].

Membranes operate by a pressure gradient across the membrane driving fluid to flow through the pores. The higher the pressure gradient or trans membrane pressure (TMP) the faster the fluid will flow through (permeate). The fluid that is unable or that does not go through the membrane is called the retentate. A solution that is being used with a membrane consists of a solvent and several solutes. In a filtration system the carrier solvent is always able to pass through the membrane although it does meet resistance. Then there are the desired and undesired solutes. By choosing a specific membrane you can control which solutes (to a given degree of control) can pass through. Those solutes play a role in fouling, but the measurement of them and the solvent that passes through the membrane is referred to as flux. Those that are rejected are concentrated near the surface as a result leading to concentration polarization, increased osmotic pressure near the surface of the membrane (leading to a decreased trans membrane pressure) and fouling. Permeability is a characteristic of a membrane that determines

how easy it is for particles to move through and exit the membrane. Another key quantity about membranes is termed membrane resistance. Membrane resistance is a way of incorporating fouling into a predictive model like equation 1.1 [4]. (J is the flux, P is the pressure, μ is the permeate viscosity, R_m is the intrinsic membrane resistance, R_f is the fouling resistance)

$$J = \Delta P / \left(\mu R_m + R_f \right) \tag{1.1}$$

There are two modes of operation for a membrane system: cross flow and dead end. Dead end flow consists of forcing through high-pressure fluid through the membrane. There is no bulk fluid flow during this and if there is it is solely towards the membrane. In a cross flow system, an applied pressure drives the fluid flow across the surface of the membrane and then the pressure difference between the fluid and the other side of the membrane drives some tangential flow through the membrane itself.

As mentioned, membranes function by as separation systems separating out particles by size. This size is characterized by a molecular weight cut off which describes the molecular weight for a mole of the material. If the molecular weight is lower than the molecular weight cut off (MWCO) than the substance will pass through the membrane. For UF membranes molecular cutoff can range from 1-1000kg/mol (or kilo Daltons). Their average lifetime is 3-5 years and they typically cost between \$3-\$5/cm^2. They have a wide range of separation uses. Some typical uses include: water purification, dairy industry cheese and whey powder making, enzyme recover, particle separation, and Dialysis and other blood treatments.

The making of membranes is a topic in and of itself. However it is beyond the scope of the research and not needed for this paper. Techniques of membrane pattern making however will be covered in a later section including a membrane making technique called phase inversion.

Fouling, Mitigation and Cleaning

Three factors affect membrane cost and use in industry: membrane production, operating TMP, and fouling and cleaning. Here we will focus on fouling and cleaning as my research pertains to that approach at cost reduction. I will explain the four types of fouling, the description of membrane resistance, and summarize the types of cleaning commonly practiced.

In order to understand fouling there are several terms that need to be defined. The first term is the word fouling itself. In simple English, it just means to make dirty or pollute, but for our purposes it is the process by which particles temporarily or permanently adhere in or on a membrane. The next term, concentration polarization, and details an extra energy cost associated with running a permeation system. Concentration polarization is an accumulation of solutes in a mass boundary layer near the surface of a membrane as a result of membrane operation [5]. This extra energy rears its head by increasing the osmotic pressure meaning we have to apply additional pressure to overcome that barrier. Osmotic pressure is the minimum pressure needed to be applied to prevent the backflow of water across a semipermeable membrane [5]

Next are the four types of fouling: adsorption, pore blockage, deposit and gel. Adsorption is a fouling mechanism caused by the attractive interactions between the solute and the membrane[5]. It is able to happen even when there is no permeate and no TMP. Fat free milk exhibits this type of fouling behavior. Pore blockage occurs from the partial or full closing of membrane pores. Pore blockage occurs internally like adsorption but is a physically blocking rather than a physical attraction that sets the particles [5]. Deposit is the growth of particle layer on the surface of the membrane [5]. These particles from the solvent are too large to enter and block the pores and traditionally form a several layers thick deposit called a "cake layer". Gel is occurs only with certain molecules when the concentration from concentration polarization

exceeds a specific quantity dependent on the molecules and solvent and forms a gel on the surface of the membrane [5]. Dispersive forces keep these four fouling methods from always clogging the membranes. Figure 4 illustrates the reduced shear rate brought about by fouling. This will play an important part in understanding later when I get to the fouling mitigation section.

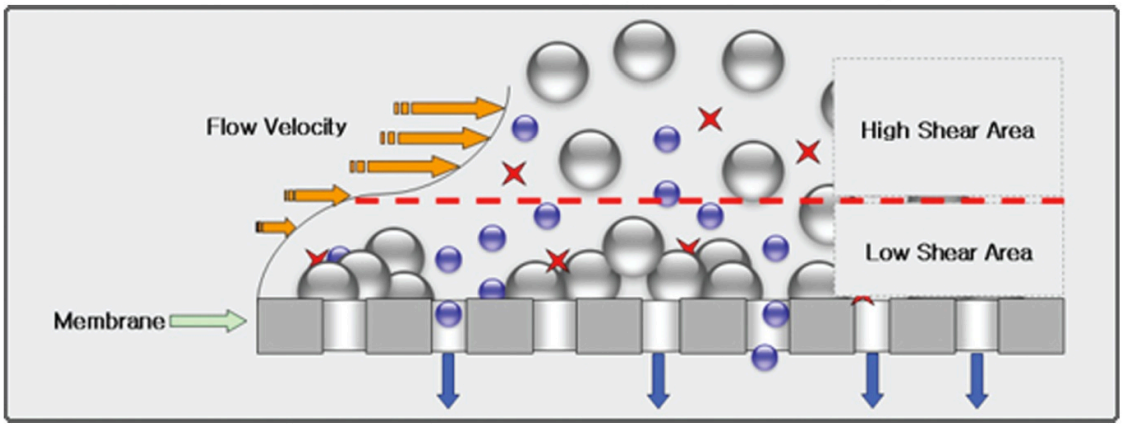


Figure 4: Illustration of fouling and its affect on flow velocity and shear.

The last two important concepts for fouling are critical flux and limiting flux (see Figure 5). Critical flux has two definitions: a strong form and a week form. The strong form dictates that the critical flux occurs when flux at a given TMP deviates from the pure water permeance flux vs. TMP line. The weak form of the critical flux is when the flux vs. TMP line deviates from linearity as at all TMP it is never matching the pure water permeance line outside of a TMP of 0. Limiting flux is the flux at which an increase in TMP no longer generates an increase in flux. Limiting flux depends on a variety of conditions from membrane, temperature, solution composition and solvent concentration and species. The first paper where the idea of critical and limiting flux began to take shame began in 1986 with Cohen and Probsetin [6]. A critical flux is defined as the flux and conditions at which no fouling occurs. A critical flux is unique for each membrane, solution temperature, and solution composition. A critical flux is referred to by its

flux and by the pressure at which it occurs. The critical flux functions at conditions on which the surface repulsive forces match the mass transport to the surface of the membranes.

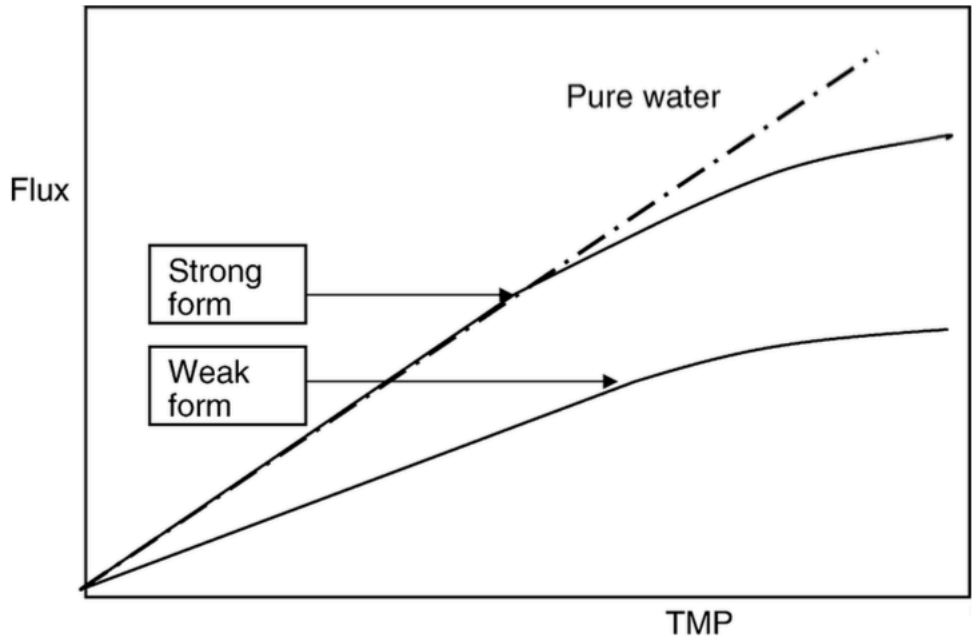


Figure 5: Representation of the two definitions of critical flux. [5]

$$N = JC - D\frac{dC}{dy} + p(\zeta) + q(\tau) \qquad (1.2)$$

The net flux of material to the surface of the membrane can be summarized with where D is the Brownian diffusion coefficient, $p(\zeta)$ is the migration due to surface interaction term, $q(\tau)$ is the effect from hydrodynamics (this is the term we aim to affect via the nano scale patterns), $\frac{dC}{dy}$ is the concentration gradient along the axis perpendicular to the surface of the membrane, J is the flux, C the concentration, N the net movement of particles [5].

As a result fouling mitigation and membrane cleaning are very important. For now will focus on fouling mitigation techniques and mechanics. Fouling can be reduced by chemical modifications to the surface or membrane, or by changing the fluid flow around the membrane. Starting with chemical alterations there isn't much that can be done since membrane materials are

already limited due to rejection choice and run conditions necessary. But optimization of membrane selection is of key importance. The more manageable and supplementary choice of making the surface of the membrane more hydrophilic is often employed though. This can cause problems as well if you have solutes that are also highly polar as it makes the surface more attractive of a binding location. Since our lab does not focus on chemical alterations to the membrane I will stop here on background as it is outside the scope of needed knowledge.

The next important thing to cover is the mechanisms behind fouling mitigation before we go about altering fluid flow to favor those situations. There are currently four main mechanisms believed to control fouling mitigation. More might exist but these four currently have enough evidence behind them but as I propose later another mechanism is needed to describe fouling mitigation when periodic nano-scale patterns are involved. The four current mechanisms are: Brownian diffusion, shear induced diffusion, inertial lift and surface transport [7]. Brownian diffusion is the movement of particles by collision and a random walk. As can be imagined it is more effective on moving smaller particles and in fact is primarily only important with submicron-sized particles. There are theories and models presented in Belfort, but it is also mentioned that those models are shown to not accurately represent experiments. As such we can only take out of this that Brownian diffusion helps offset concentration polarization but to the degree we can't determine, nor need to for the purpose of this paper, as the mechanics behind it are the least likely to relate to our effects from the pattern. Next is shear-induced diffusion. Shear induced diffusion is the movement of particles away from the surface by a non linear shear gradient near the surface of the membrane and its action upon particles. This will be discussed in further detail later. Inertial lift is a model that depends upon nonlinear interactions of a particle with the flow field. While very convoluted it simplifies as fluid dynamic forces acting unequally

on a particle in a transverse flow field cause the particle to rise. Additional details will be covered later in the paper. Surface transport theory dictates particles slide and roll along the surface of a membrane in transverse flow and that the cake layer itself can leave the domain and is continuously replaced by a new cake layer.

For fouling mitigation, common techniques are to introduce flow instabilities, vortices, and turbulence. While a high Re causes turbulence, it can be caused at lower Re values through appropriate use of flow conditions such curvilinear flow. Whole system turbulence though causes a bigger use in energy instead of just near surface mixing and fouling reduction and as such is not the focus of our work. Instead we will focus on the flow instabilities and mixing vortices. Instabilities are created in three general ways: geometry change to the flow channel (figure 6a, b, c), pulsatile flow (Figure 6d), and curvilinear flow under the correct conditions (Figure 6e, f) [7]. Pulsatile flow (Figure 6d) has the advantages of mimicking the cardiovascular system and pairs well with surface roughness. Unfortunately it has a large energy cost, reduced net cross flow, and doesn't scale up well. On the other hand curvilinear flow was actually studied for a commercial attempt. Curvilinear Taylor vortices (Figure 6e) had very large wall shear rates and mixed the bulk fluid exceptionally well, but you could not backwash to clean (will cover that later), extremely difficult to repair, and touch to scale up, it had an exorbitantly high energy cost to constantly rotate at a high enough speed to create mixing. Curvilinear Dean vortices (Figure 6f) on the other hand were easy to scale up and successfully reduced solute build up at membrane surface but were too expensive to be used in large systems. Geometry modification is which our work actually belongs in. In fact we are a new type that closely resembles protuberance, but I'll get to that soon. One of my personal favorites, furrowed surface (Figure 6b), does superb mixing but as can be imagined, it is extremely difficult to scale and

manufacture this geometry. Inserts (Figure 6c) do not decrease membrane surface area unlike protuberance and cause large vortices but they aren't at the surface where they are needed to remove particles. It also causes a large pressure drop, which means a lower TMP and permeates. And it doesn't scale up well as you imagine. Protuberance creates vortices and mixing at the surface and has the smallest energy drop from only near surface instabilities. Unfortunately with this model you have reduced surface area for permeation and only small vortices. Our system, which I will call "patterned", is a seventh type. The patterned type looses the disadvantage of surface area reduction and for micron scale patterns increases the surface area of the membrane. However, with our nano scale patterns due to current manufacturing techniques there is a small drop in permeation due to compression of parts of the membrane causing local drops in the permeance of the membrane. Furthermore, thanks to the simulations, it looks like while this type (depending upon the surface type and cross flow rates) can generate vortices on the surface, it appears the flow instabilities only create and upward movement of particles away from the surface and on the nano scale form an additional steric hindrance to surface attachment.

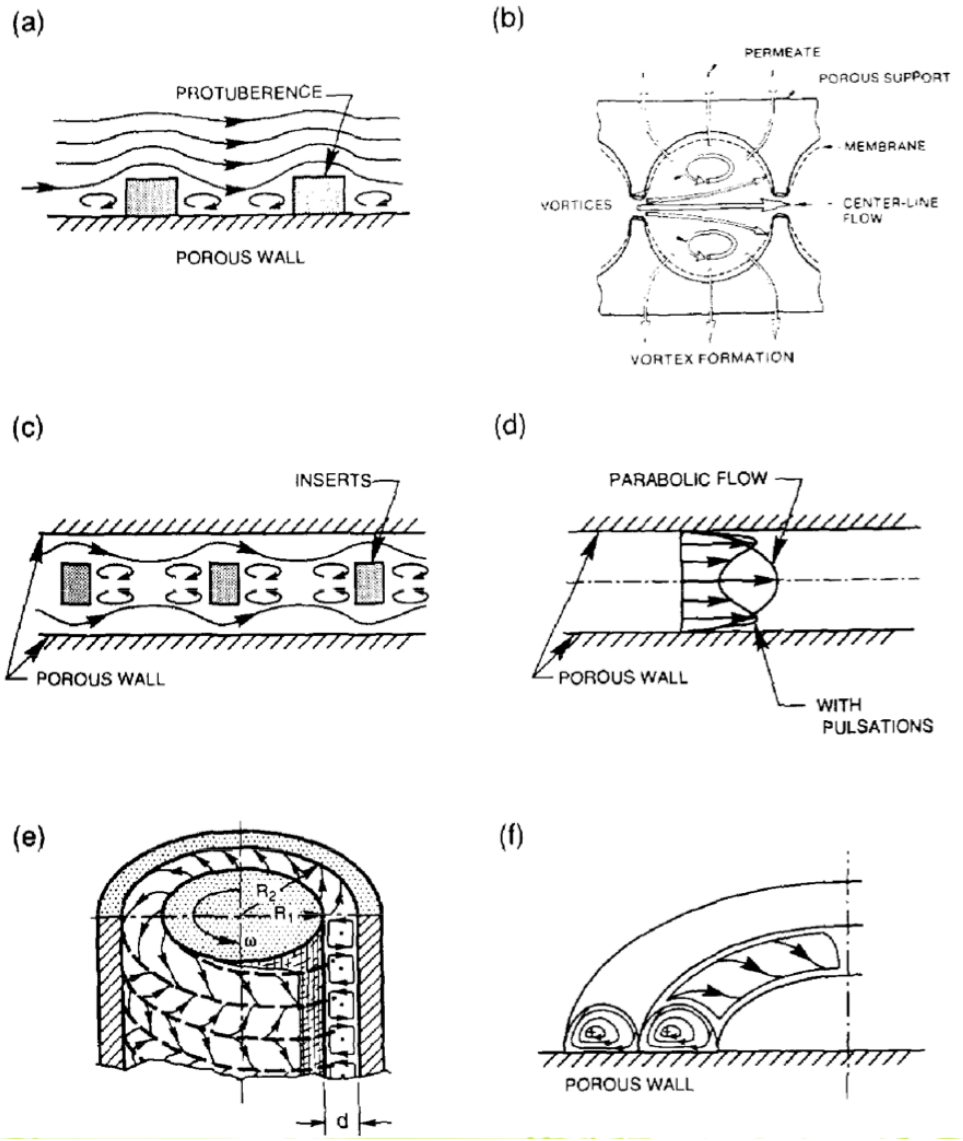


Figure 6: Each method induces at least either instability or vortex formation a) Protuberance-vortex formation and instabilities, b) Furrowed Surface (Corrugations) -vortex formation and instabilities c) inserts-vortex formations and instabilities d) pulsatile flow-instabilities, e) Taylor Vortices- vortex formation, f) Dean Vortices- vortex formation [7].

Lastly is the section on cleaning. Besides operation this can be the most expensive part of membrane system operations. For example, the dairy industry when making cheese uses a UF membrane filter called HFK328. In order to make cheese they have to concentrate the milk by removing water from the system. During this process, every day they have to stop for 7 hours to clean their membranes with acid, bases and enzymes. That wastes a lot of time and has a large environmental impact even if they use cheap chemicals like they do. As such systems such as the

patterns that can improve ease of cleaning are useful in and of themselves. Now the two types of cleaning are physical and chemical. Typically most industries use both. Physical cleaning requires using deionized (DI) water to remove foulants by either fluid dynamic flow or by diffusion of concentrated foulants on the surface into very clean DI water (for example, our lab uses 18MΩ DI water). The first method where you flow water backwards through a membrane so is called back flow and is effective but not doable in all membrane systems. Relaxation is the name where you let pure water soak the membranes and let foulants diffuse out. Another approach is pulsatile and high agitation flow but as mentioned it is rather costly to do. Lastly there is a cross flow purge, which uses an increased surface shear to remove non-tightly adhered foulants. In our milk experiments we will use relaxation and cross flow to clean out our membranes in our protocols. In regards to chemical cleaning it really depends upon what the membrane is and what the foulant is. But for example in milk, which the primary foulant is casein micelle, a combination of acids, bases and enzymes are used to denature the protein and make it easier to remove (actually the acid step only serves to deactivate the enzyme given the acid they use, but more on that later in the milk section).

OpenFOAM

OpenFOAM is an open source C++ library for fluid dynamic simulations. While the most basic uses are covered in an instruction manual, the lack of a gooey interface requires the user to be well versed in C++, mathematics and fluid dynamics to make advanced use of. They are unfortunately known for having a harsh learning curve. As such I'm going to cover the basics of OpenFOAM usage and leave the appendix for the detailed guide and the experimental section for how to duplicate my results.

There are three algorithms that handle the actual solution-solving portion. They are the SIMPLE (Semi-Implicit Method for Pressure Linked Equations), PIMPLE (It doesn't actually stand for anything. It is just a combination of PISO and SIMPLE so they merged the letters), and PISO (Pressure Implicit with Splitting of Operator) algorithms. In our actual simulations we used SIMPLE for all the final results but used PIMPLE for some transient testing we did. The primary differences between PISO and SIMPLE besides the transient vs. steady state is that in PISO no under relaxation is applied and the momentum correction step can be performed more than once. The PISO steps and a detailed OpenFOAM implementation of it can be found here[2]. PIMPLE is basically PISO but it is more robust than PISO and allows the user to input a control courant number and let the algorithm adjust the time step. PISO you put the time step and need to make sure it will allow the courant number to be below one.

I used the simpleFoam program to handle all the solving. It uses the SIMPLE algorithm, so I will go into more details here of their processes. SimpleFoam is a steady state solver designed for turbulent flows but with the ability to turn on the turbulent model and run laminar state systems. It is unfortunately not designed for low Re solutions but with some solution control file modification it can run accurately if slightly slowly at low Re. As usual it solves a modified navier-stokes equation and the continuity equation (equations 1.3 & 1.4). A few notes, R is a modified term that encompasses the turbulent behavior by modification of the viscous stress but in our laminar case is $\nabla \bullet \mathbf{R} = -\upsilon \nabla^2 \mathbf{U}$ and so simplifies to the normal navier-stokes. The terms are defined as follows: U is the velocity vector, ν is the kinematic viscosity, and p is the kinematic pressure.

$$\nabla \bullet (\mathbf{UU}) + \nabla \bullet \mathbf{R} = -\nabla p \qquad (1.3)$$

[2] https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM

$$\nabla \bullet \mathbf{U} \tag{1.4}$$

The SIMPLE algorithm is a standard segregated solution method meaning that U and p are solved separately and then coupled. Convergence occurs when the residuals for each variable is reduced to below the convergence criteria specified by the user. The residuals are all actually normalized such that the residual is 1 on the first iteration (see equation 1.5 for residual). U is the velocity vector at a given iteration given by the subscript while r is the residual.

$$r_n = \frac{U_n - U_{n-1}}{U_1 - U_0} \tag{1.5}$$

The SIMPLE algorithm solves the breaks the navier stokes into several discretized matrix equations that it can actually solve. Equation 1.6 is the pressure equation, 1.7 is the continuity equation, 1.8 is the momentum equation and 1.9 while not having a name represents the neighboring cells and unsteady terms. (a_p is a grouping of coefficients from the discretized velocity equations. p is the kinematic pressure, H(U) is an unnamed term that handles neighboring cells and unsteady terms, S is the outward facing face area vector, U_f is the velocity on the face, Δt is the time step) [3]

$$\nabla \bullet \left(\frac{1}{a_p} \nabla p \right) = \nabla \bullet \left(\frac{H(\mathbf{U})}{a_p} \right) \tag{1.6}$$

$$\nabla \bullet \mathbf{U} = \sum_f \mathbf{S} \mathbf{U}_f = 0 \tag{1.7}$$

$$\mathbf{U}_p = \frac{H(\mathbf{U})}{a_p} - \frac{\nabla p}{a_p} \tag{1.8}$$

[3] http://openfoamwiki.net/index.php/The_SIMPLE_algorithm_in_OpenFOAM

$$H(\mathbf{U}) = -\sum_n a_n \mathbf{U}_n + \frac{\mathbf{U}^o}{\Delta t} \tag{1.9}$$

The SIMPLE algorithm is the following steps:[4]

1. Set boundary conditions
2. Solve discretized momentum equation to compute the intermediate velocity field
3. Compute mass flux at each cell face
4. Solver pressure equation and apply under-relaxation (under relaxation decreases the change between iterations as a mean to prevent overshooting the solution and increase solution stability at the cost of slowing down the rate at which the solution converges)
5. Correct mass fluxes
6. Correct velocities based off new pressure field (accuracy in pressure is the critical factor in SIMPLE so spend time in the fvSchemes and fvSolution file on the pressure terms)
7. Update boundary conditions
8. Repeat till convergence criteria is met for all variables

Now I will explain the general premise of operation of OpenFOAM. The first thing you do is creating your mesh. Next you describe the initial conditions and boundary conditions in the "0" folder in your case file. Each variable will have conditions. For a simple laminar flow only velocity and pressure are required to be described. Next is the "constant" folder. It consists of a

[4] http://openfoamwiki.net/index.php/The_SIMPLE_algorithm_in_OpenFOAM
https://en.wikipedia.org/wiki/SIMPLE_algorithm

subfolder called "polyMesh" where that mesh you created in step 1 goes. It also contains your transport, turbulence and RAS properties. In our Newtonian laminar flow transport is just the kinematic viscosity. Turbulence and RAS properties need to be turned off for the laminar flow. Lastly the "systems" folder contains much of the more complicated documents. It contains the "controlDict", "decomposeParDict", "fvSchemes", "fvSolution", and "sampleDict". These dictionaries and files determine the convergence, run time and accuracy of the solution along with post processing and parallelization. More detailed information about everything can be found in the appendix; however, the velocity solver PBICG and pressure solver GAMG deserve a little more attention.

The velocity solver, PBICG (Preconditioned Bi-Conjugate Gradient Solver for Asymmetric matrices) is more efficient for pressure when using over 1024 processors (which we were not). It is a Krylov type solver. Krylov Subspace solvers solve the matrix by use of orthogonal subsets that span the space. They use schemes such as Lanczos iteration for hermitian matrices and Arnoldi iteration for general matrices to describe the space. Lanczos is an adaption of the power methods to get the eigenvalues and eigenvectors of the space. Arnoldi is also an adaption of the power methods but with general matrices. As a result it produces a non-orthogonal basis that is than orthanormalized by the Gram-Schmidt method. [8]

OpenFOAM stores its results as a single Matrix output of a list of values in order of stored points. As such to gather and export data a subroutine called "sample" was used (hence the sample dictionary "sampleDict" in the systems folder). Sample is one of the better defined and easier to use subroutines in OpenFOAM. Sample functions by telling the type of information you want pulled along with all the various locations you want it pulled and what type of file format you want the exported data stored in. Since OpenFOAM is a cell-based

system you also have to tell it an interpolation scheme and that can affect the values you receive so be careful and consistent. Post processing is handled visually in "paraFoam" or the paraview program or a third party software. But the default post processing system does not have an easily accessible export system so come prepared with that foreknowledge.

OpenFOAM has a few meshing requirements. The peskiest one is that no face can be shared by more than two cells. This means that for hexahedral mesh that is generated in OpenFOAM is unable to have large meshing concentration gradients. Instead one must have dense meshing throughout the mesh. I believe that uniform mesh is actually the best because it helps with post processing as well, but due to CPU hour constraints I understand the need to prioritize density in prime locations. As such third party software should be used for meshing either complicated geometry or geometry that requires significant concentration gradients. I have additional information in the appendix on how to actually mesh in GMSH and open source meshing program and OpenFOAM directly along with the qualities of good mesh such as skewness, non-orthogonality and aspect ratio.

Milk Experiments

Milk is a complex fluid consisting of minerals, sugars, fats, and both soluble and insoluble proteins. As a complex fluid it was an asset for experimentation to further understanding of the industrial benefits of imprinted membranes. Previous filtration had focused solely with silicon particles, which while informative, do not behave like most standard solutions that do need the filtration. In addition, the dairy industry is a substantial user of membranes in its industrial processes. The stage of filtration that our experiments are studying is an UF step where the products are an enriched casein micelles solution that is used for making cheese and a solution of soluble proteins, minerals, and lactose that is used for making whey [9]. One of the

reasons milk is being studied is that even now the fouling of skim milk is still not fully understood [10]. However, that isn't to say that we don't have several ideas of what is going on; I will address what is currently known later in the paper. But first I would like to address where industry stands in its membrane and cleaning use. Frequency of cleaning varied but the least frequently cleaned example was cleaning for 6 hours for every 25 hours of actual filtration [11]. The problem is that frankly milk is highly fouling and it takes a lot of time and resources to clean it in place. In addition, concerns for hygiene play a role in the frequency of cleaning [11]. This frequent use of cleaning cycles has an effect on both the lifetime (2-3 years) but also leads to a sizable decline in membrane performance during the early stages of its use [11]. Furthermore, the cleaning stages are not only expensive and time consuming but also contribute 1/3 of the negative environmental impacts of the whole process [10].

While a complex solution, milk is a substance that experiences irreversible fouling while operating below the critical flux, even if that fouling is markedly reduced [12]. To make matters worse a significant portion of fouling can occur almost immediately [13]. In fact the fouling can occur just by dipping the membrane into milk [13]. This initial fouling is caused by protein adsorption on the surface of the membranes but tapers off within the first ten minutes of exposure [13]. To further exacerbate the problem there are several other fouling mechanisms at play and not all groups agree upon which mechanisms are dominant because several play different roles at different times. Some groups like Youravong and Metsamuuronen demonstrate that concentration polarization plays a major role in reduced permeate [12, 14]. Other groups like Rabiller illustrate that the gel layer that forms is ruled by casein micelles and how they deposit and bind to the surface [15]. He also put forward and agreed with Youravong that proteins caused the irreversible fouling and so cleaning should focus on that [15]. I think James does a

good job of describing the whole process. Protein adsorption onto the surface and in the membrane can restrict pore access and ease of flow. But the rejected proteins and other particles accumulate near the surface due to slow back mass transport (which is something we aim to fix with our patterns) and a concentration polarization occurs. Then the concentration polarization can become dense enough to form a gel layer. The gel layer can either further compact or grow restricting permeation [13]. Berg proposes that the three main solutes that cause fouling are salts, lactose (although the lactose can easily be washed away by a standard wash) and proteins. His experiments showed that while salt didn't foul itself it aided the fouling of the lactose and proteins [11]. He noted that the acid, which is used to dissolve inorganic salts such as calcium phosphate and wash them away, had a negligible effect on the development of resistance [11]. Jimenzlopez isolated Casein micelles as the primary irreversible foulant, but that various other factors (such as soluble proteins or salts) contributed to how much fouling occurred. In particular he showed that the casein micelles (diameter 187nm±7nm) networked with calcium ions to bond to each other and the surface [9].

Seeing how big fouling is, operating below the critical flux is often a goal of milk experimentation, even though UF of skimmed milk is often done in the limiting flux region far beyond the critical flux (see figure 7) [12]. Critical flux increases with wall shear stress but some work shows that protein denaturing from significant amounts of repeated cycles and high shear to cause an accumulation of membrane fouling leading to the postulation that their is an optimum cross flow velocity beyond which the critical flux decreases [12, 14].

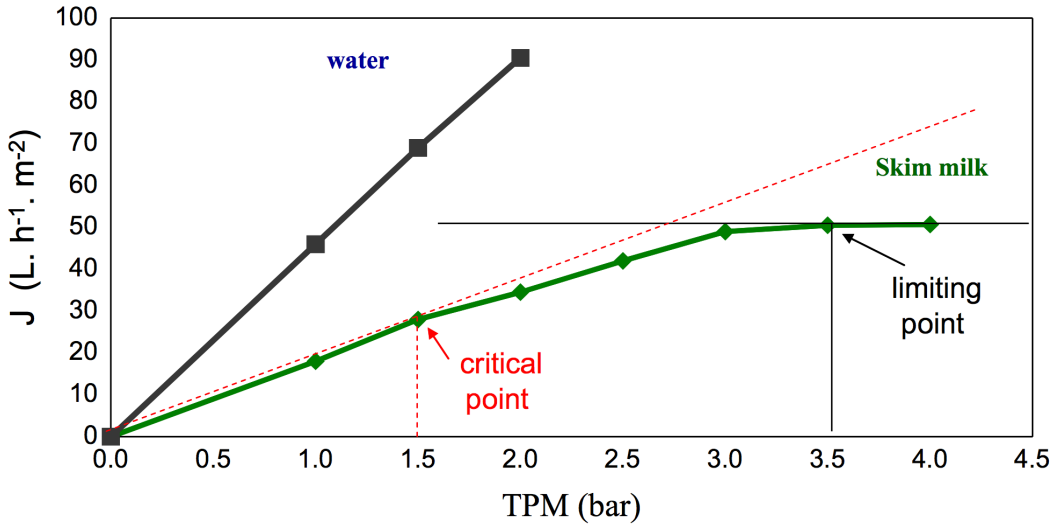


Figure 7: Critical and limiting flux of skim milk plotted in traditional TMP vs. Flux format [10].

Lastly we will focus on industry cleaning and then the literature behind our own modified cleaning cycles. The standard cleaning cycle follows similar to figure 8 but with significantly longer time and more repeated cycles. Meanwhile figure 9 shows the flux recovery observed from each of the stages. Typically the milk in industry will be run at 50ºC [12] but this particular paper did not. We do see that the enzyme does the majority of cleaning before appearing that the acid significantly cleans the surface. I will now remind people that the acid serves to turn off the enzymes and to dissolves salts, however, other work showed that salts themselves don't particularly foul the membrane.

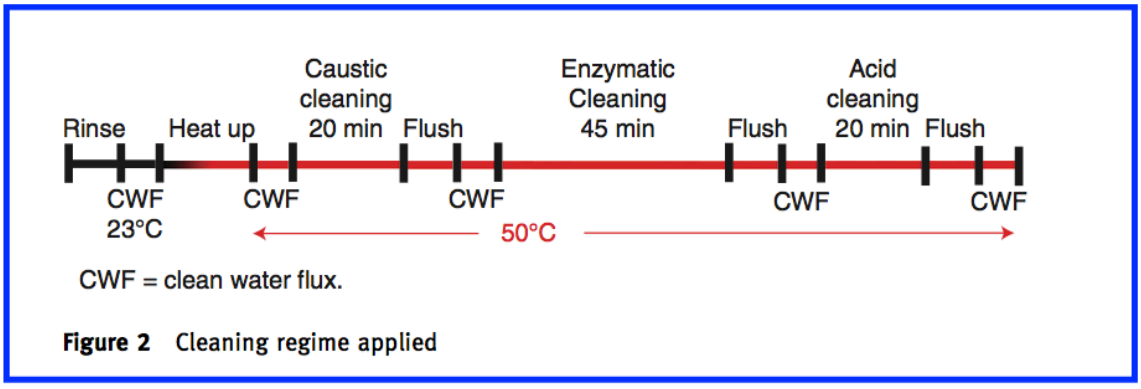


Figure 8: Standard lab cleaning cycle mirroring industry-cleaning procedures with reduced times [11].

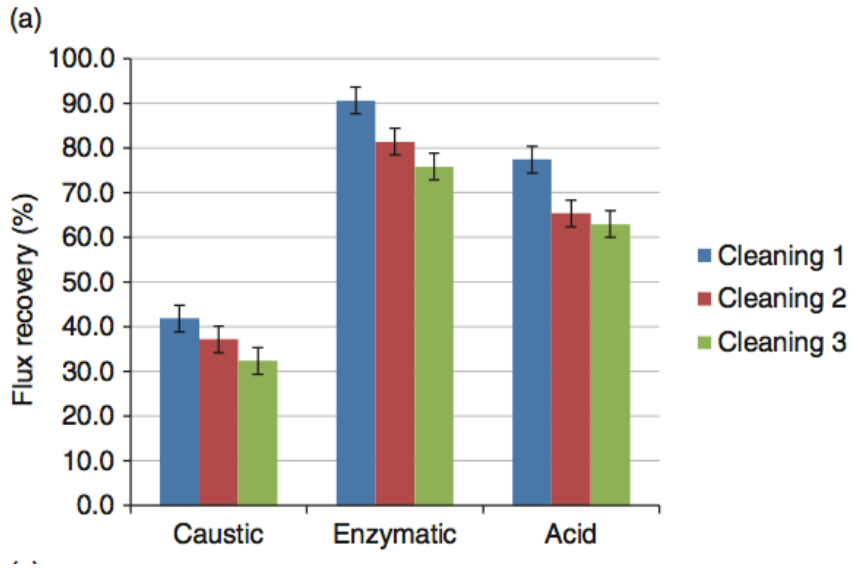


Figure 9: Demonstrates the experimentally recovered fluxes after each of the individual steps [11].

As such Paugam's group decided to investigate and their discovery was remarkable. They found that nitric acid only increased the flux but did not clean the membrane by removing proteins (Figure 10). In fact they found that just rinsing with water after the fat free milk UF that there were no minerals left (done by SEM-EDX) [16]. Now while all flux recovery is useful for industry, it isn't useful for laboratory experiments. Since our lab did not have access to the enzyme and their proprietary surfactants and chelators [11] this work shows that the acid step should be left out. This is because the higher the efficacy of the caustic cleaning, the lower the impact from nitric acid; however, the worse a job, the more nitric acid over estimates the cleaning efficiency[16]. This is due to the increase in overall hydrophobicity of the membrane due to the adsorption of the nitrate on proteins. In face you can see acids like Citric acid remove proteins as well, but are unfortunately more expensive since they are derived from foods[16].

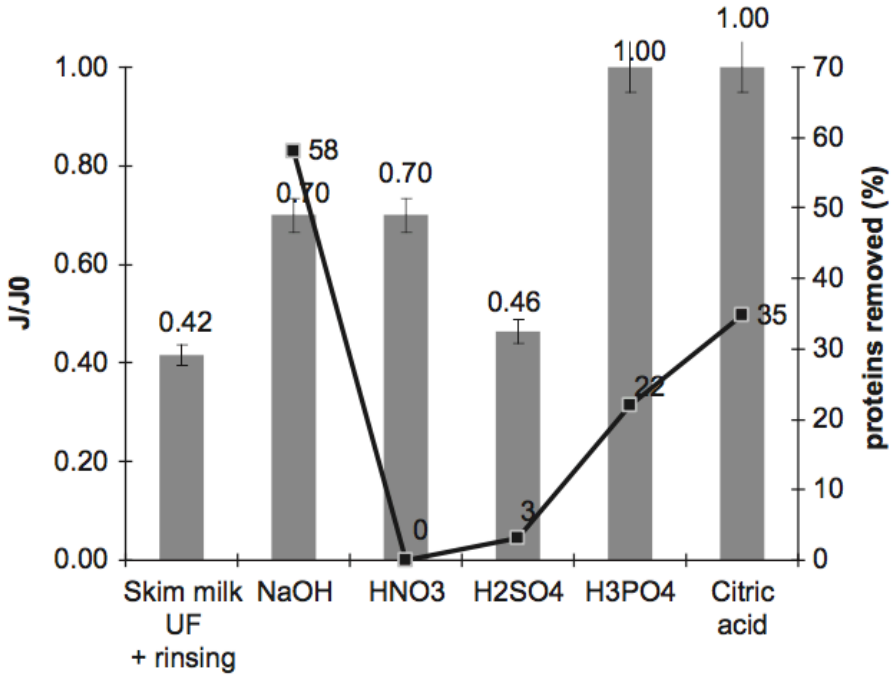


Figure 10: DI water flux recovery (bar charts) and % of removed proteins (black squares) after respective treatments [16].

Motivation

This section will seek to address why we are interested in nano scale imprints given that random roughness is known to increase fouling. To see what makes patterned periodic roughness different we must examine the work done by Maruf who studied experimentally nano scale patterned membranes. The benefits to the membranes can be summarized as increased flux rate, increased critical flux, later onset of fouling, slower growth of cake layer, and higher flux recovery. I will now go figure by figure with why each is beneficial to the membrane filtration industry spending special attention with the implications with the dairy industry.

Figure 11 shows that unlike micron sized patterns made with the phase inversion process, the increase in surface area does not directly translate to an increase in permeance. In phase inversion the membrane is formed directly in contact with the a patterned surface substrate so when the membrane is finished forming it already has the pattern on it and the surface pores are

completely normal. This allows for higher surface area to increase the permeation. With nano scale imprints, which cannot be imparted by the phase inversion process and are instead imparted through our NIL technology cause some deformation of the surface pores leading to an increased rejection. What happens is on the surface that is compressed there is a slight decrease in porosity [3]. This translates to a slightly lower pure water permeance even with the increase in surface area. However, the property that is more valuable than pure water permeance is flux during filtration. Figure 11 illustrates that the patterned membrane has a higher permeate flux for a given TMP. This is most likely due to the decrease in fouling and surface build up rather than the change to porosity given the results of pure water permeance but it hasn't been shown explicitly. A factor we also see is the size dependence of the particle. These experiments were all performed on membranes with patterns seen in figure 12 with a width of 417nm and height of 110nm. Changes to these parameters will impact the effect on particles. This is important to the dairy industry and as well as other groups as it means more products for the same energy input and time.

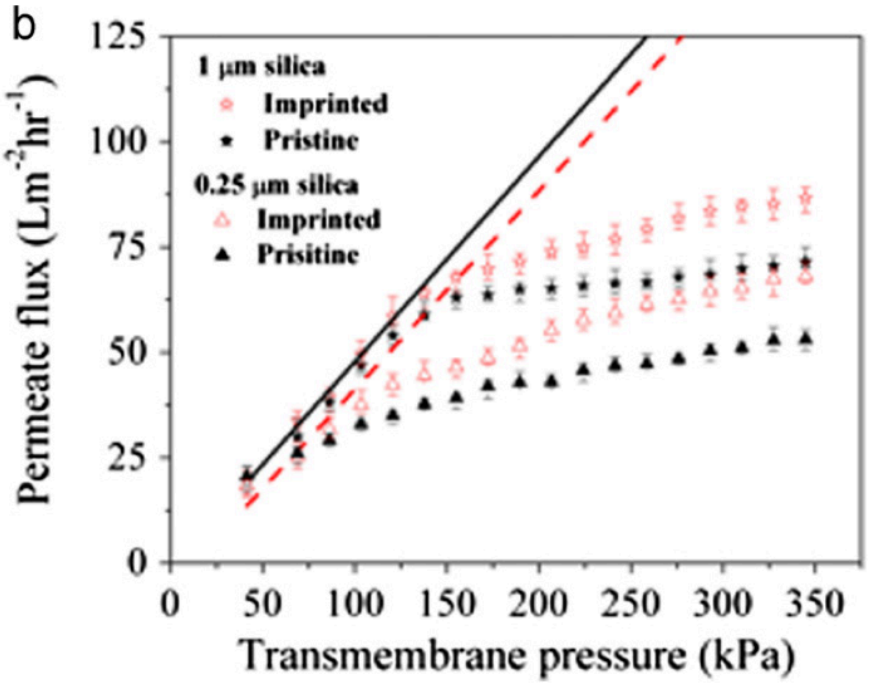


Figure 11: Pure water permeance is plotted along the strait lines while filtrations done with two different sized particles were also performed and represented by the noted shapes [17]

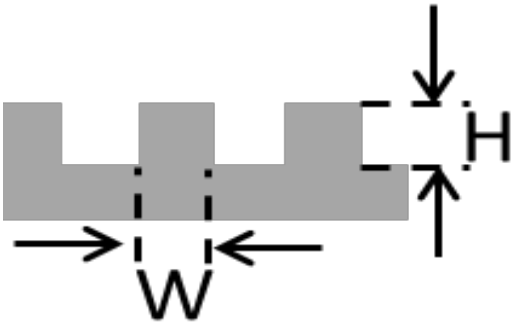


Figure 12: The spacing of the nano-scale imprinted membrane[17].

Figure 13 compares critical flux values as well as fouling behavior above them. The critical flux was effectively doubled between patterned and unpatterned membrane. For reasons explained with critical flux this means that while the energy cost increases you can run at a much higher flux rate, increasing the product per time ratio without having to do increased cleaning also entailing a profit for the industry. As seen when run above the critical flux fouling will cause the permeate to drop below the critical flux value explaining the logic behind why most systems

desire to run below the critical flux. Unfortunately not all systems have clear critical fluxes. As previously explained milk will foul under all circumstances but the fouling reduction is still useful but no longer a mandatory reason to remain below it.

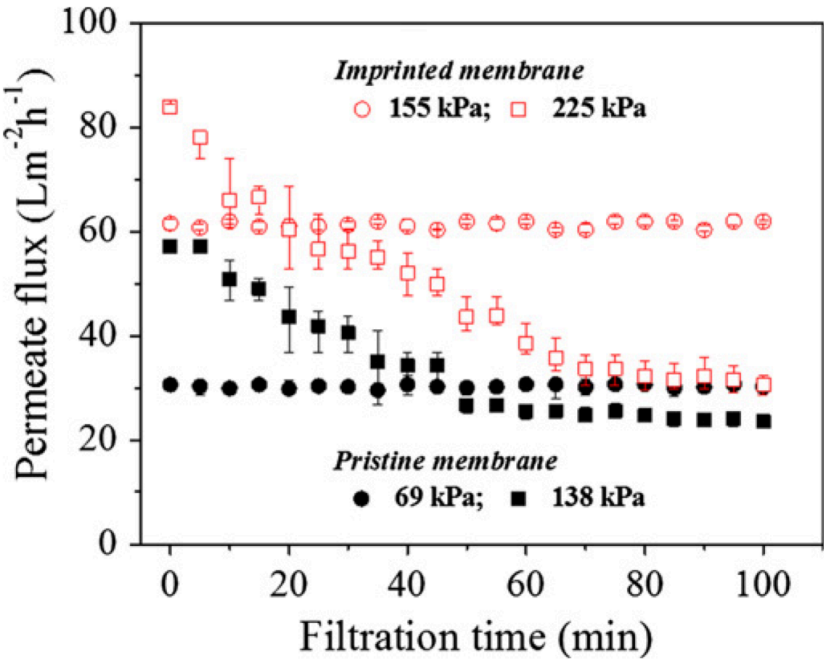


Figure 13: Experimental plot demonstrating the higher critical flux of the patterned membrane and the benefits of running below the critical flux for overall production capabilities.[17]

In figure 14 the onset of fouling is directly related to the higher critical flux value for the patterned membrane. The new information presented here is the decreased growth rate of the cake layer. This corresponds to the ability to run for a longer period of time before having to do cleaning. This would be particularly beneficial to the dairy industry but this particular property does not apply to complex fluids with milk's unique fouling capabilities, but will still be useful for a variety of other filtration applications.

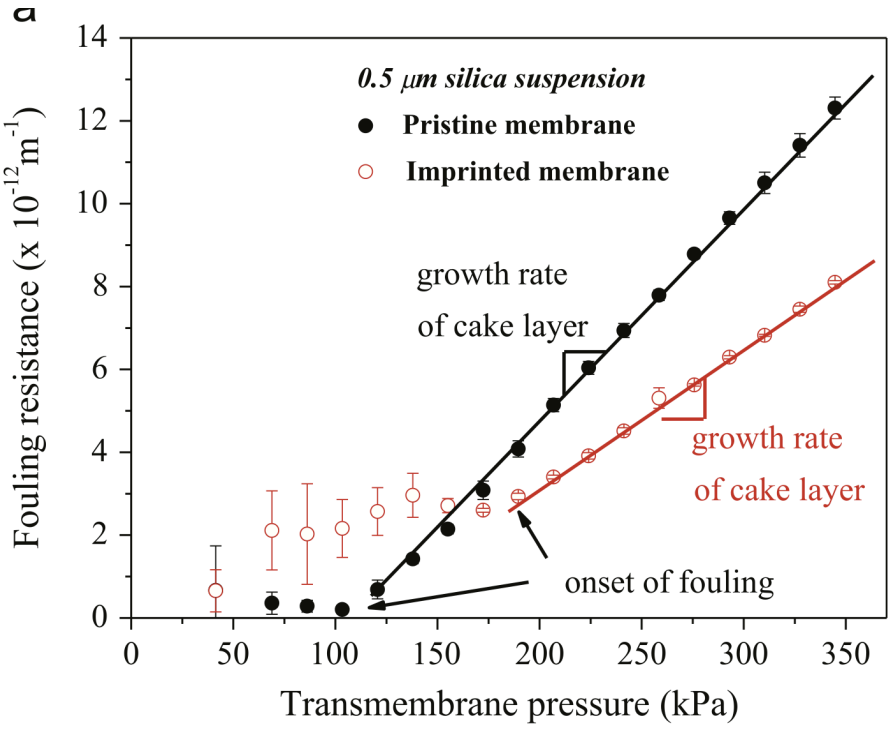


Figure 14: Illustrates the delay of cake growth and the delay in the onset of fouling.[17]

Figure 15 illustrates the real gold value for groups like the dairy industry, a more effective cleaning. As mentioned previously cleaning is very expensive, takes a long time and has to be done frequently. The only result we have so far obtained experimentally is that flux recovery is improved with the patterns and milk filtration. However, for semi complex fluids the overall benefit to filtration industries is phenomenal.

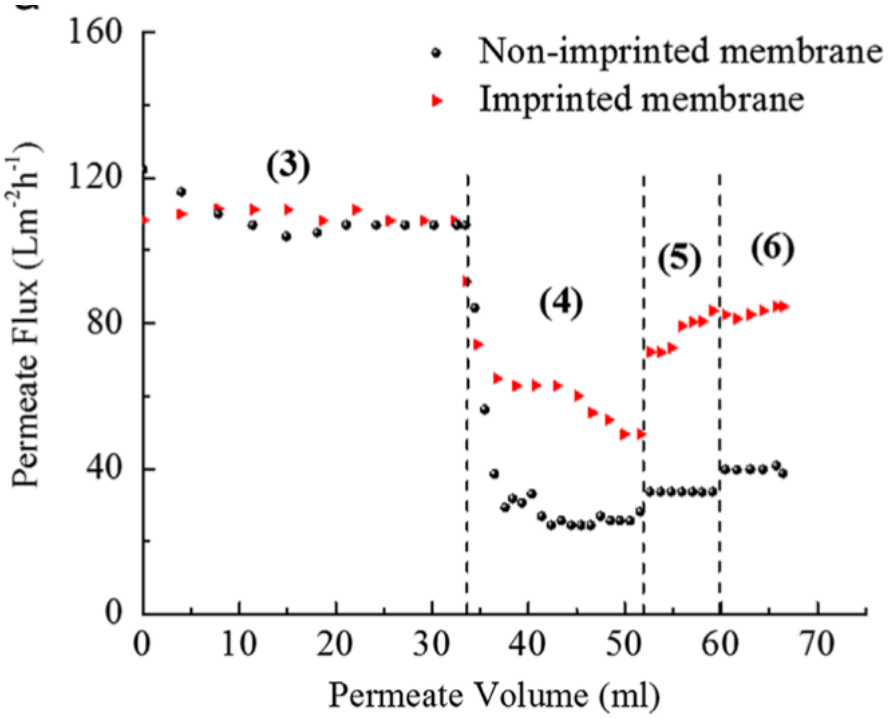
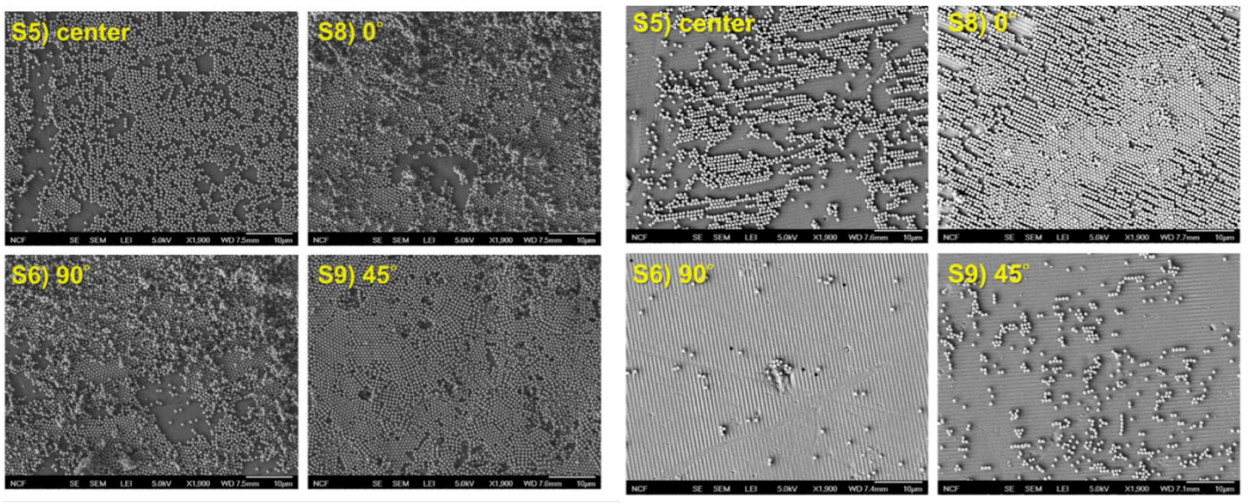


Figure 15: Illustrates the recovery effect of the patterning. Section three is a PBS filtration at 276 kPa. Section four is a PBS and BSA filtration at 276 kPa. After section four there is a membrane cleaning cycle. Section 5 is a fresh PBS filtration at 276 kPa. Then there is a DI water flush. Lastly in section seven there is another PBS filtration at 276 kPa. [3]

I will now focus on why we chose the parameters that we did to study: the angle of attack, permeation rate, pattern height, and cross flow speed. Some has already been explained. We theorized that different particle sizes would require different spacing's and so we decided to vary pattern height. As already mentioned fouling mechanisms relate to cross flow speed and so that was varied. Due to critical flux we decided to vary the permeation rate with our CFD work. Angle of attack had initial demonstration that flow direction mattered with a patterned surface (figure 16). However, all of these parameters were also studied with experiments by Maruf and so serve as a comparison point for our CFD solutions.



a) Control Membrane

b) Imprinted Membrane

Figure 16: Depicts the fouling conditions and angles of attacks between a patterned and pristine membrane [17].

Chapter 3

CFD Experiments

Solver Validation

When using a new CFD program it is important to validate the program's solutions with analytic cases before moving onto simulations with unknown answers. The better this is done the more one can trust the results of the simulations whose answers are unknown. As such it is important to validate the program under conditions similar to unknown solutions. OpenFOAM as a program had a wide variety of solvers to choose from but no steady state laminar solver. This was initially a problem since we were trying to solve that type of problem. However, OpenFOAM is readily modifiable for both code and case procedures. With a significant amount of trial and error, settings were found that produced accurate results matching the analytic solution to these three cases: biharmonic equation (cavity with a lid), slit flow and lid driven flow. In addition to the analytic cases, we will be doing a mesh sizing comparison of the actual experiment to make sure that all the particulars of the fluid flow are fully captured and the mesh is refined enough for our purposes.

I will save the final specifics of the settings for the methods section of the experiment and instead walk through the analytic cases and our results. The biharmonic equation (1.10)[18] in fluid flow is used to describe the flow and formation of vortices in a closed cavity with a moving lid. The analytic solution of this case gives the location of the center of the recirculation centers as seen in figure 17 and table 1. We will validate the location of the first and second vortex but only visually confirm the existence of the corner vortices.

$$\nabla^4 \psi = 0 \qquad (1.10)$$

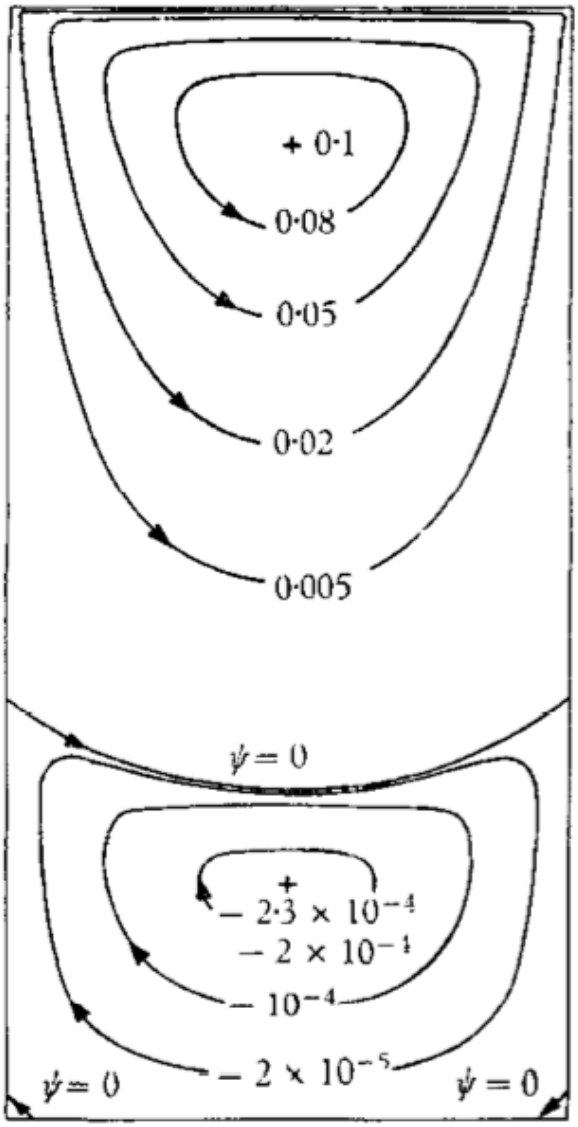


Figure 17: Visual representation of the location of the vortices in the analytic solution and their values for a given aspect ratio of 2 [18].

		Vortex centre x/D		
$A \equiv H/D$	h (mesh size)	1st	2nd	3rd
0·25	0·0125	0·0875	—	—
0·50	0·0125	0·1628	—	—
1·0	0·01	0·24†	—	—
2·0	0·025	0·25	1·575	—

Table 1: Biharmonic equation solution for moving lid in a cavity. List of locations of the vortexes as a function of aspect ratio and mesh size. Our simulation uses the aspect ratio of 2[18].

The cavity simulation was run with the solver simpleFoam with an absolute convergence tolerance of 10^{-9}. This took 140,000 iterations to arrive at the solution. Due to the coding of OpenFOAM's stream functions the starting points and scaling factors of stream functions will be different then in the literature. However, since we are seeking only the locations of the vortices that will be acceptable. Figure 18 shows the whole domain that was simulated. Due to axis choices mine is horizontal and not vertical. Still the left wall is the wall that moves upward. Meanwhile, Figures 19-21 will show the vortices mentioned in Figure 17 and Table 1. The error for both locations is less than 1% and as such is an acceptable value.

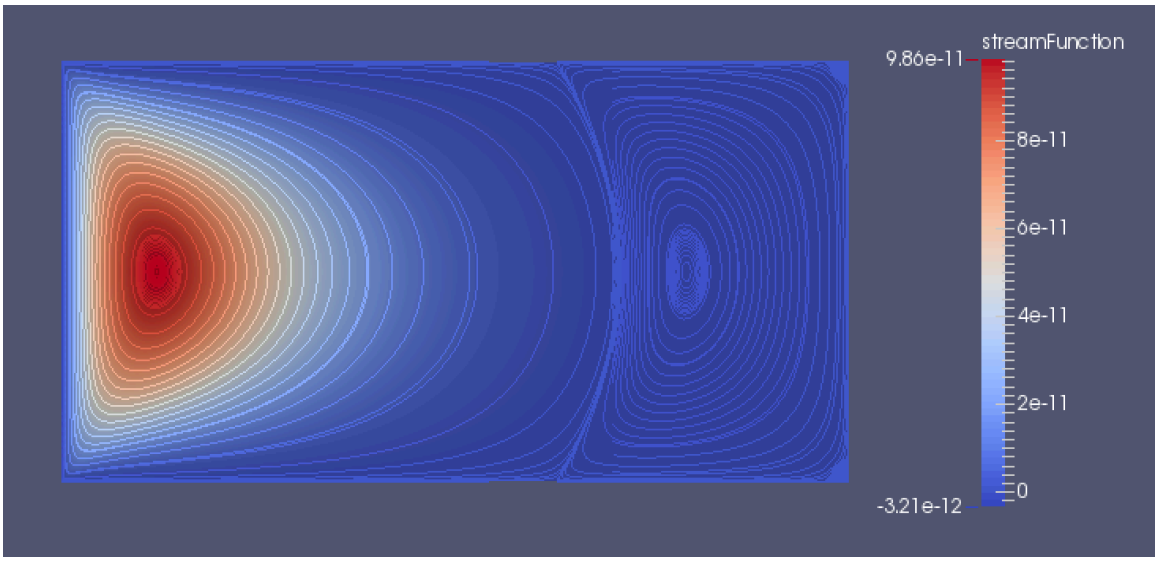


Fig 18: Shows the complete domain for an aspect ratio 2 in a cavity with a moving lid. The picture depicts the various streamlines that are generated.

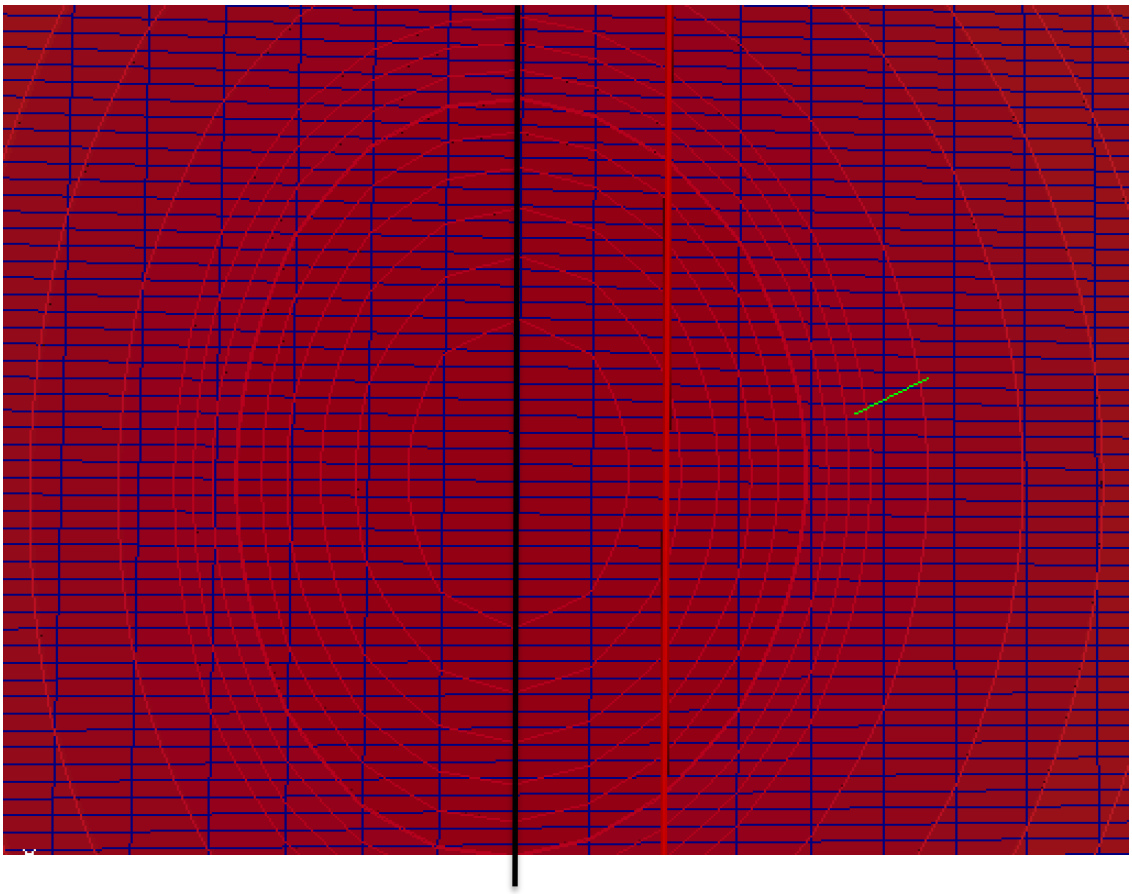


Fig 19: The first vortex generated by the simulation. The black line represents the center of the vortex from the simulation while the red line indicates the anaytic center of the origin. In this simulation it was only 2 mesh points away. That corresponds to only a 0.4% error in the location of the vortex.

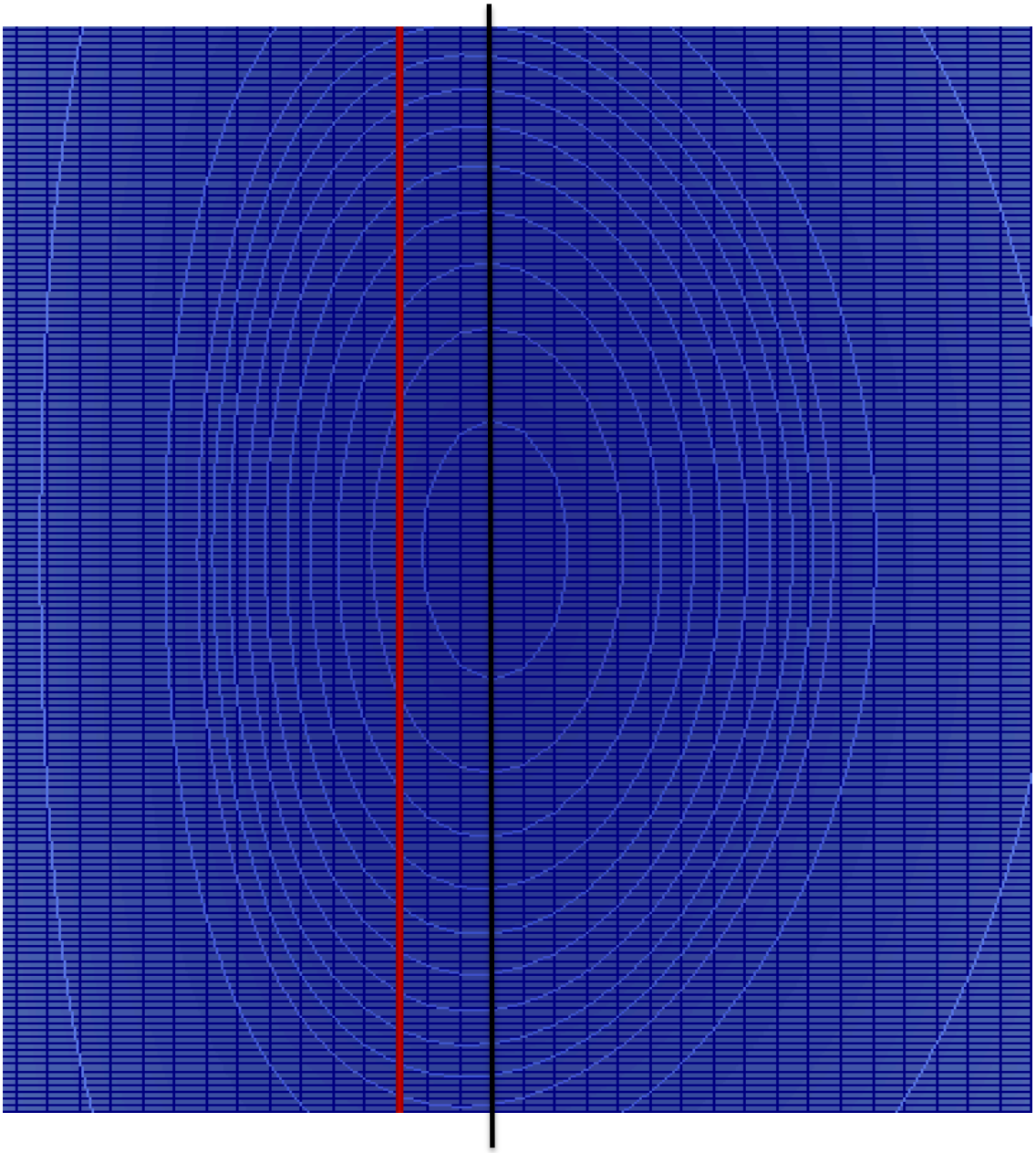


Fig 20: The second vortex is off by 3 mesh points. Here the vortex center is 79.5% of the way down leading to a 0.75% error.

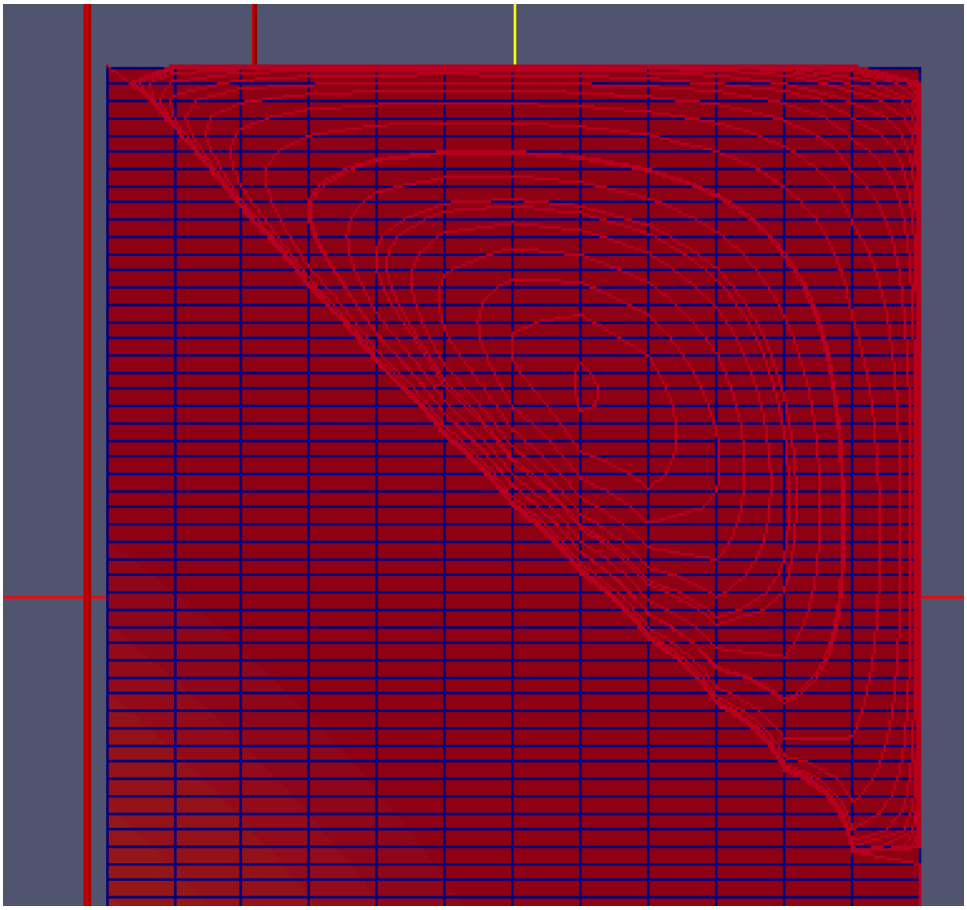


Fig 21: The corner vortex, while not having a specified location center does appear to show up like in the analytic solution at both corners. However, it is at the scale where additional mesh refinement would prove useful.

Overall simpleFoam, which is designed as a steady state turbulent flow solver, accurately predicts the locations of the vortices along with flow features. The original case where this flow had the initial default conditions took 1.4 million iterations to reach the same result. However, in the slit flow case the analytic solution was unachievable under the initial conditions.

Slit flow is also known as plane Poiseuille flow (figure 22). It is flow between two parallel plates where the width between two plates is much smaller than the length of the simulation. The flow profile is parabolic and is driven by a pressure gradient but can also have a moving top plate. In this simulation we used the simplest case, which is driven by only a pressure gradient. This was represented by a uniform inlet velocity boundary to provide the

initial mass flow and to develop into the full parabolic solution. This was done to make it easy to identify the max velocity which is 3/2 the average velocity which is also the inlet velocity.

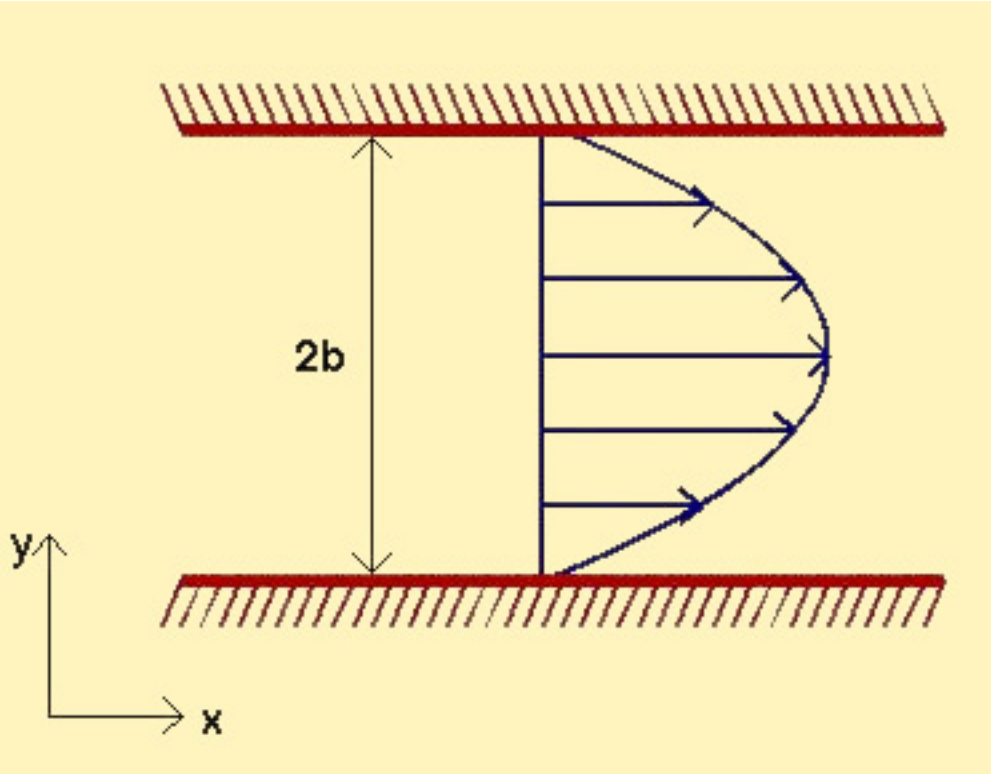


Figure 22: The Plane Poiseuille flow case we are simulating which involves fixed walls[5]

The initial results with the default conditions provided by OpenFOAM yielded the following disaster seen in figure 23 and 24. This is why validation must be done. It should be noted that the solution converged by OpenFOAM standards for simpleFoam under the default conditions. By modifying the fvSchemes and fvSolution files to what is seen in the appendix and discussed in great details later in the methods section, I was able to then correct this to the analytic solution as seen in figures 24 and 25. The fluid dynamic conditions applied in this flow is an inlet velocity of 0.25m/s with a Re of 1 and an aspect ratio of 1/2. I will note that in Figure 27 we had been experimenting with the idea of non-uniform density of mesh spacing. While it provides accurate results near the surface of the membrane our post processing system doesn't

[5] https://uwaterloo.ca/applied-mathematics/sites/ca.applied-mathematics/files/uploads/images/poiseuille_flow.jpg

handle smooth interpolation schemes well and so the data becomes rough and deviates as it gets farther away from the "membrane" surface.

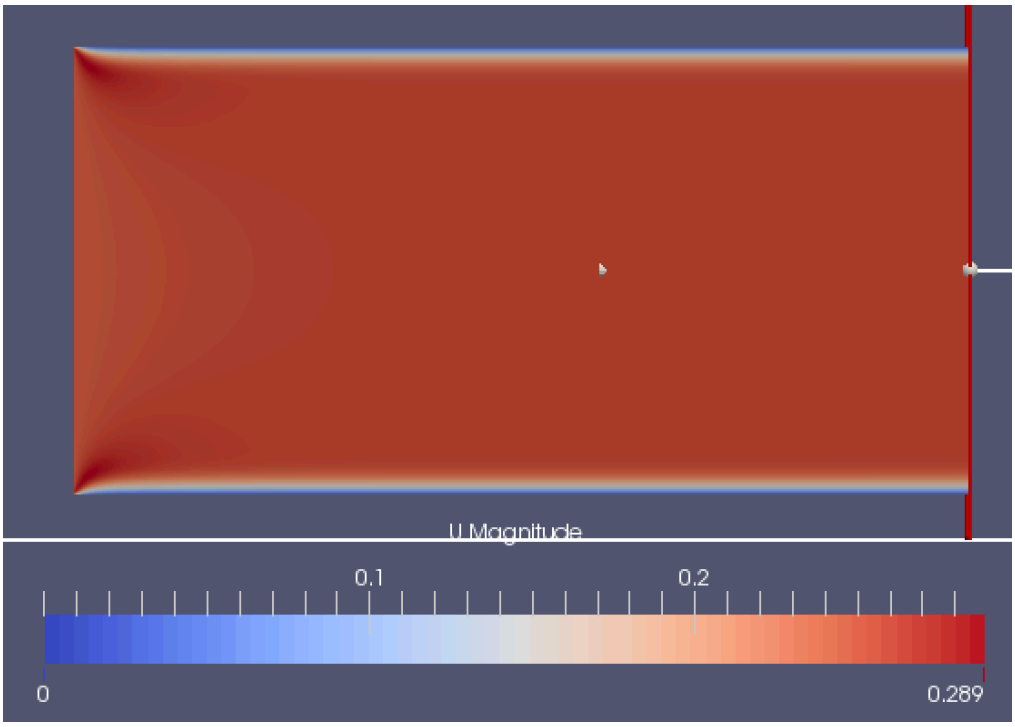


Figure 23: The full velocity profile of the plane Poiseuille flow under default conditions. It is fairly obvious that while the CFD is converged, that the simulation was not fully developed at all.

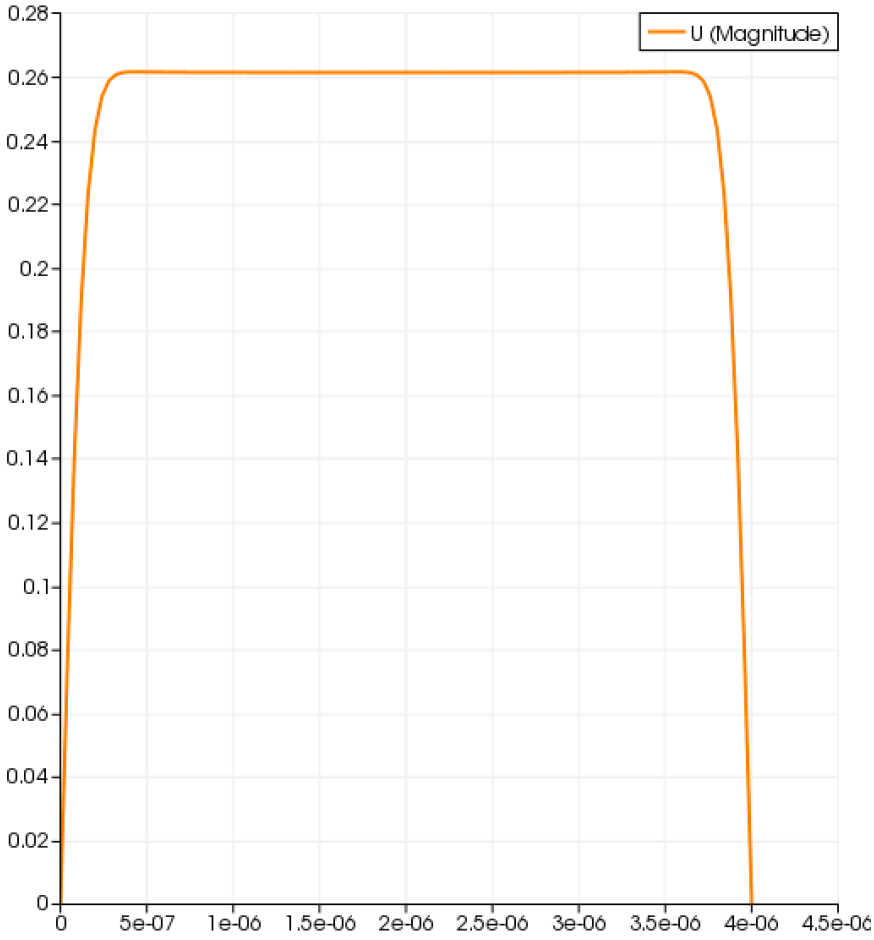


Figure 24: This is the velocity profile at the slice indicated by the red line in the previous picture. As is seen, the flow is strictly a plug flow and not slit flow result.

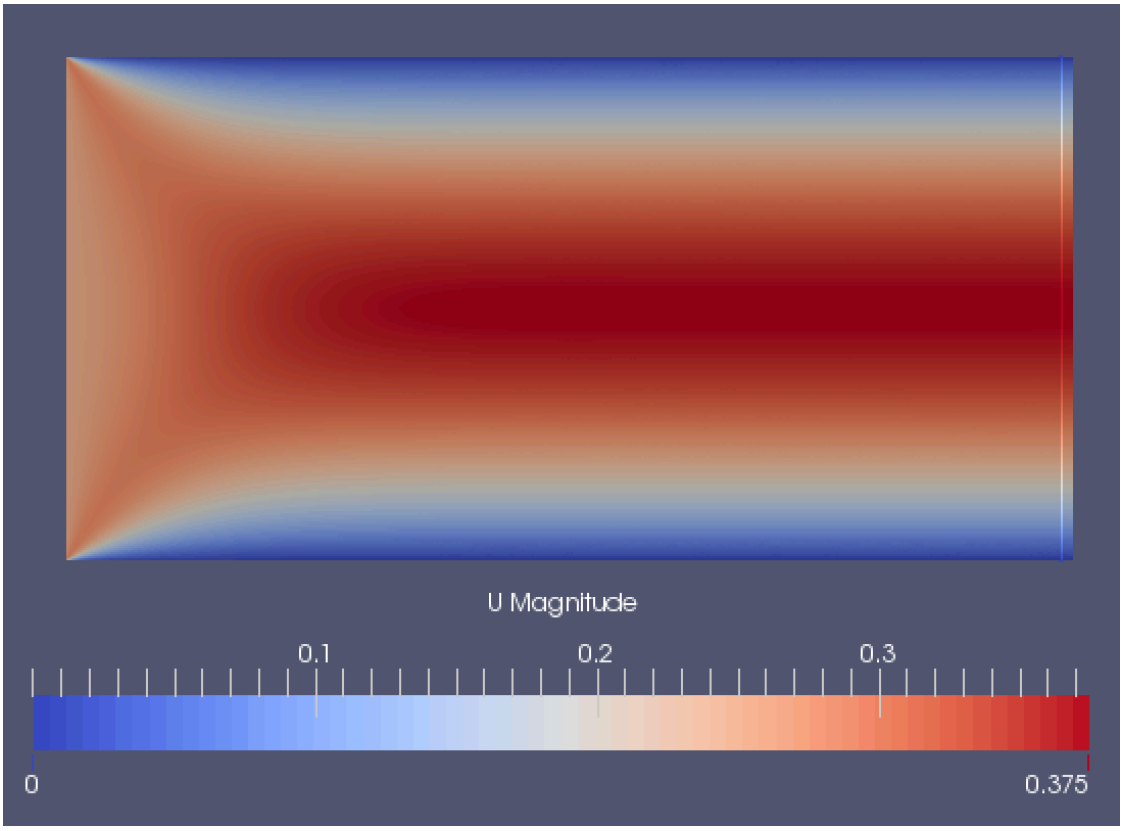


Figure 25: Velocity flow profile for slit flow after adjustments were made to the fvSchemes and fvSolutions files.

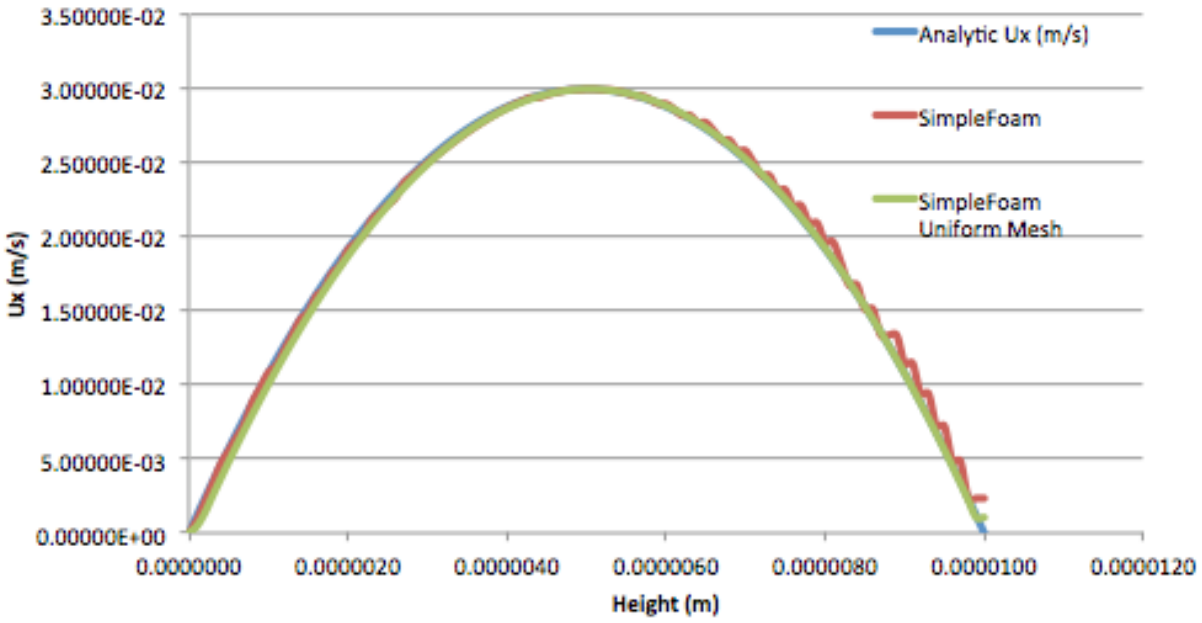


Figure 26: the velocity profile from the slit flow after the modifications to the fvSolution and fvSchemes file. Details the correct result matching the analytic solution to within 0.5%

Using the same systems folder files the moving slit experiment was done with an inlet velocity of 20m/s and an aspect ratio of 2. The analytic solution matched fairly well with our experimental results (Fig 27). Figure 28 is included as a validation that it works for the whole domain. This concludes our matching to analytic solutions. Up next is mesh refinement.

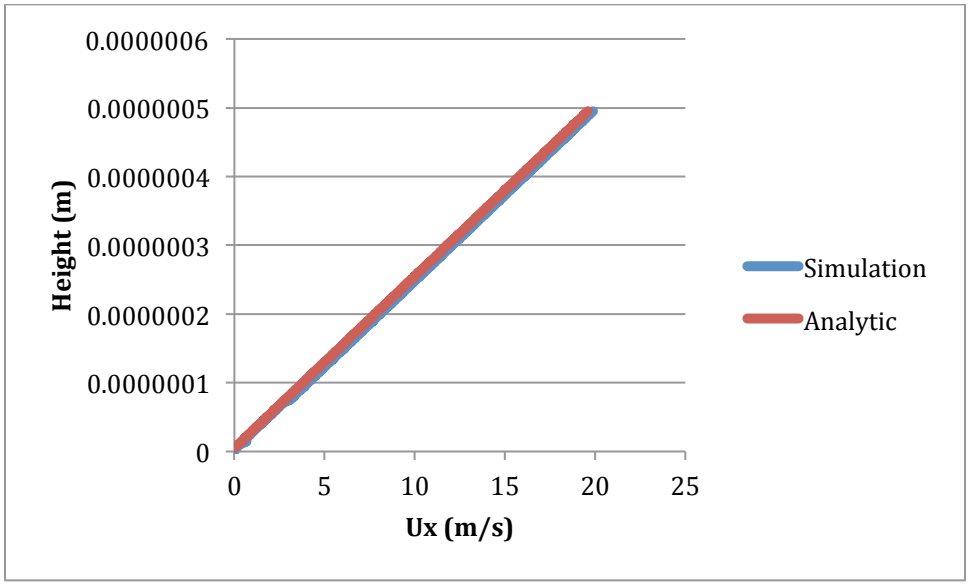


Figure 27: The velocity profile for the analytic comparison of the moving top flow.

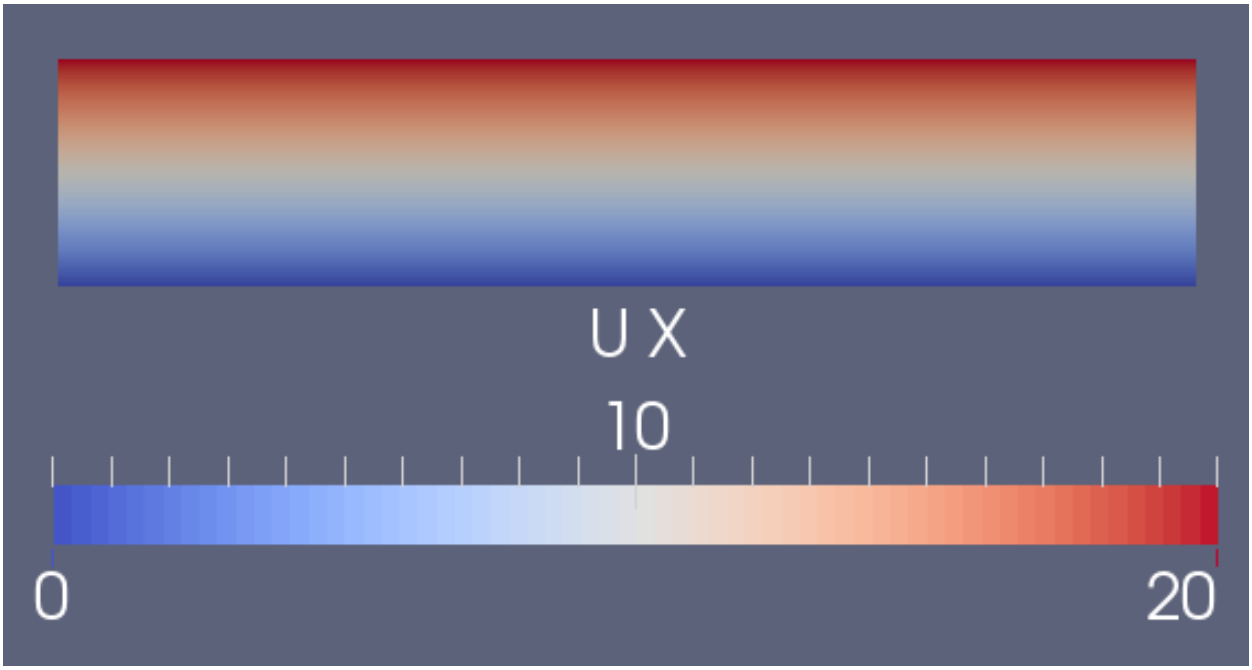


Figure 28: The horizontal velocity of a lid driven cavity at velocity of 20m/s with 5nm vertical mesh spacing.

Considering mesh refinement is an essential task for the validation of CFD results. The nature of high cost of CFD means we want to make spacing as large as possible while still resolving the essential flow features. What follows is the result of three simulations of varying meshes densities (~10nm, ~20nm, and ~30nm). Jumping to the point, in figure 29 they plainly overlap. Furthermore, I checked error variation and while the error discrepancy was fairly large for the last points because my mesh got coarser than my sampling (two points were sampled in the same cell). However, the average discrepancy was ~1% between coarsest ~30nm spacing and the most refined ~10nm spacing. As such, I am confident that my mesh spacing is refined enough to ensure accurate results. Further mesh refinement would have only served to make the sampling easier, but is limited by CFD capabilities and the necessity to do the simulations in 3D.

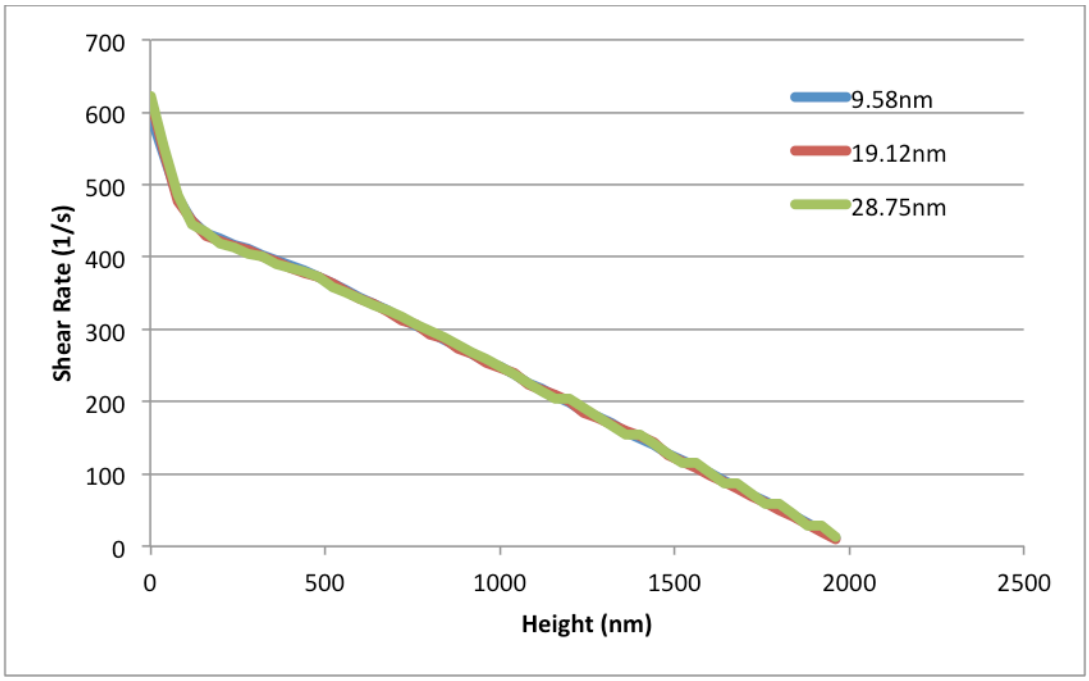


Figure 29: Demonstration of a random slice of the 160nm patterned membrane 90-degree case with a Re of 239.

Motivation on Fouling

Earlier I promised more detailed information on a variety of subjects. I will now address those topics here. There will be a little bit of overlap, but much greater details here, especially

concerning modeling. So the CFD focusing on examining previous experimental work [3, 4, 17] on the hydrodynamics in and around submicron patterns, which were imprinted on a commercial UF membranes. Experimentally it has been shown that with nano-scale patterns the fouling mitigation depends upon angle of attack, pattern height and cross flow speed, and permeation rate for simple solutions (silica particles of varying size). In order to do simulation experiments with the critical flux we'd need a model that included particles. That is currently outside the scope of our work but might be done with future work. Our lab developed novel techniques to imprint nano-scale patterns on a membrane surface and retain its porosity replacing the older techniques of phase inversion that did not support nano scale patterns [19-22]. The motivation for the development of these membranes, and these modeling studies, is that flux-decline and fouling are major limiting factors for the continuous operation of pressure-driven separation processes such as ultrafiltration (UF), especially during the filtration of protein solutions [23, 24]. Flux decline, and its major subset "fouling", is a natural result of selective membrane-based separations [5] and is a hindrance. The deposition of retained particles, macromolecules, inorganic and biological materials, at the membrane surface and/or inside the pores [25, 26], often can only be partially removed under harsh chemical treatment [27, 28] causing more energy consumption, loss of productivity, and shortened membrane lifetime [25]. Mitigation of membrane fouling still remains a grand challenge for most membrane applications.

Many methods for fouling mitigation do exist. The reason they have not been widely implemented in industry is due to flaws within each method that increase the difficulty, cost, or environmental impacts from the previous mitigation methods. Controlling interactions between the membrane surface and the feed solution is critical for fouling mitigation [26, 29]. A popular and partially used method is modifying the surface of the membrane. By increasing its

hydrophilicty you'll have an increased water permeance [26]. The downside is that it doesn't affect the internal parts of the membrane nor any solutes that are desired permeate that are less hydrophilic. In other cases this technique is an undesired affect as it artificially makes the membranes look cleaner but does not largely contribute to the functionality of separation in membranes. An example of this is nitric acid, which is used during the cleaning of membranes from milk filtration. It doesn't actually clean the membrane and is used only for shutting down an enzyme used for cleaning in the previous step[16]. Lastly while the cost of nitric acid isn't too bad, its not a particularly environmentally friendly chemical, especially in the concentrations used. It takes a lot of resources to properly dispose of it. Another possible solution is surface and flow system topology. By altering the membrane topology or the topology in the flow cell one can altar the fluid dynamics near the membrane surface or in the bulk flow. By doing so many groups aim to reduce the fouling. Several of these methods do indeed work but also have flaws, usually in the scale up to industry quantity. By altering the surface topology one can create near surface vortices, promote mixing and increase shear-induced diffusion. Increased shear has been studied without the use of surface patterns and has been shown to be an effective way to reduce fouling[30]. Using patterns, the geometry can significantly affect the micron-scale pattern's positive effect. Prism and triangle trenches, where recirculation vortices make up the bulk of the pattern [31-33], are not as effective as the sinusoidal patterns [34] where flow follows the surface. While micron-sized patterns for particle fouling is still being investigated, the effect of these patterns on biofouling have been very successful [35, 36]. One pattern, not on a membrane, by the name of Sharklet® is already commercialized for resistance to bacteria settling on the surface [35]. Unfortunately, little study has been done with nano-scale patterns, where the pattern features are smaller than the primary chemical foulant. In fact of the many groups

running CFD simulations over patterned surface none simulate on nanometer sized patterns. A group from Korea [33], runs simulations and experiments on membranes with 400μm x 200μm prism patterns. This is an entire three orders of magnitude larger than ours. As a result different phenomena would play different roles between their work and ours. Unfortunately current literature focuses on the hundreds of micron scales and the atomic scales at the moment meaning our results are all novel in nature.

Models for fouling mitigation

In the 1970's Eckstein and his colleagues first quantified the phenomena that would later be known as shear induced diffusion. In their first paper, they related the self-diffusion coefficient (D) to concentration and several other terms in a dimensionless relation. While the full equation (1.11) shows a multitude of terms, the concentration (ϕ) and the gradient of the strain rate $\frac{a\frac{\partial^2 u}{\partial y^2}}{\omega}$ are the two most important quantities to us [37]. Unlike the nonlinearity term discussed later ω is a constant. That concentration as the paper shows is the heaviest term is highly logical. This term measures the gradient of the shear rate over the cross-section of a particle. It gives the steady state slip velocity and the transverse lifting force [38]. The transverse lifting force in the patterned membranes is what we currently hypothesize to reduce the fouling (particle deposition). Specifically, our results to do not agree with the inertial lift theory (see equation 1.12) that is an alternative to the shear-induced diffusion models [39] nor with the critical flux modeling that had experimental fitting (equation 1.13)[4]. However under their experimental conditions the term that is of interest for the CFD, the strain rate term, is approximated as zero [37].

$$\frac{D}{a^2\omega} = \wp\left[\phi, \frac{a}{w}, \frac{a}{y}, \frac{\rho_f a^2 \omega}{\mu}, \frac{a\frac{\partial^2 u}{\partial y^2}}{\omega}, \frac{g\left(\rho_s - \rho_f\right)a}{\mu\omega}\right] \tag{1.11}$$

$$v_{L,0} = \frac{\rho_0 r^3 \dot{\gamma}_0^2 f(\hat{y})}{16\eta_0} \tag{1.12}$$

$$J_c = 0.051\mathrm{Re}^{0.433} Sc_i^{0.863} \frac{0.3\dot{\gamma} d_i^2}{d_h} \tag{1.13}$$

Ingber developed an improved transport model for shear-induced diffusion under physical conditions similar to those used in our experiment and simulations. Adding a slip boundary at the wall and modeling the diffusion coefficients as linear functions of a non-linearity parameter make these improvements. The result is a non-linearity parameter that directly and linearly correlates with shear induced diffusion that can be used as a model for our data. A particular finding of Ingber's is that for neutrally buoyant particles, the particles migrate from a high shear rate to a low shear rate [40]. Our results show that over certain regions of the patterned membrane, near the surface there is an increase in shear rate over a flat membrane. This is part of what contributes to particle migration away from the surface more effectively than in pristine membranes. A previous study by Ingber in 2008 [41] showed the particle migration to areas of low shear when the flow had nonlinear shear. In a smooth membrane the shear is of course linear. Our patterns cause near surface nonlinearity guaranteeing this benefit and that the non-linear parameter (1.14) is valid for our case examinations.

$$\xi_{nl} = \frac{a\left|\nabla\dot{\gamma}\right|}{\left(\dot{\gamma} + \dot{\gamma}_{NL}\right)} \tag{1.14}$$

$$\dot{\gamma}_{NL} = \frac{a}{R_0}\dot{\gamma}_0 \tag{1.15}$$

In Ingber's paper he illustrates his model with a Couette flow system from a stationary outer cylinder and a rotating inner cylinder. For several reasons we choose to approximate that

$\dot{\gamma}_{NL}$ is 0. One of such is that the paper admits that the non local shear rate $\dot{\gamma}_0$ plays a minimal role in Couette flow which our simulation of the surface approximates [40]. In addition, since we are only using a small part of the system, the R_0 term is significantly larger than the particle radius used in our lab experiments. As such we choose for simplification of observations that is safe to reduce the nonlinearity parameter to the simple relationship in equation 1.16. Although our simulations did not include particles, we can still validate the correlation that was seen with the particle size in the experiments.

$$\xi_{NL} = \frac{a|\nabla\dot{\gamma}|}{\dot{\gamma}} \tag{1.16}$$

Geometry

So now I need to provide the detailed information on the 4 geometries used in the case: the jet flow cell, cross flow cell and their each respective nano-scale cases. So starting with the impinging jet flow cell, low-resolution (smallest elements ~10 μm) CFD modeling of the unique flow cell system used in [3, 17] was performed to determine the boundary conditions for analysis of fluid flow at the ~10 nm scale above the membrane. The cell is axisymmetric so a two-dimensional model using radial symmetry (Figure 30a) with a high mesh density at the bottom surface (where the membrane is located) was generated. The feed entrance tube has a Reynolds number (Re) of 114, which is still in the laminar region. The second region consists of the rest of the flow cell, which has a Re of ~7.2 (also laminar flow regime). The two Re were derived from the inlet volumetric flow rate being used in the two different volume regions: the inlet tube and the free open bulk volume (see appendix). The finite volume domain mesh was created in GMSH with at total of 77584 cells giving a nearest neighbor node distance of 10 μm near the surface of the membrane. The mesh is hexagonal and has a non-orthogonality average of 7.3 and a

maximum of 31.6 (non-orthogonality is a measurement of mesh quality and a non-orthogonality below 40 is considered good mesh).

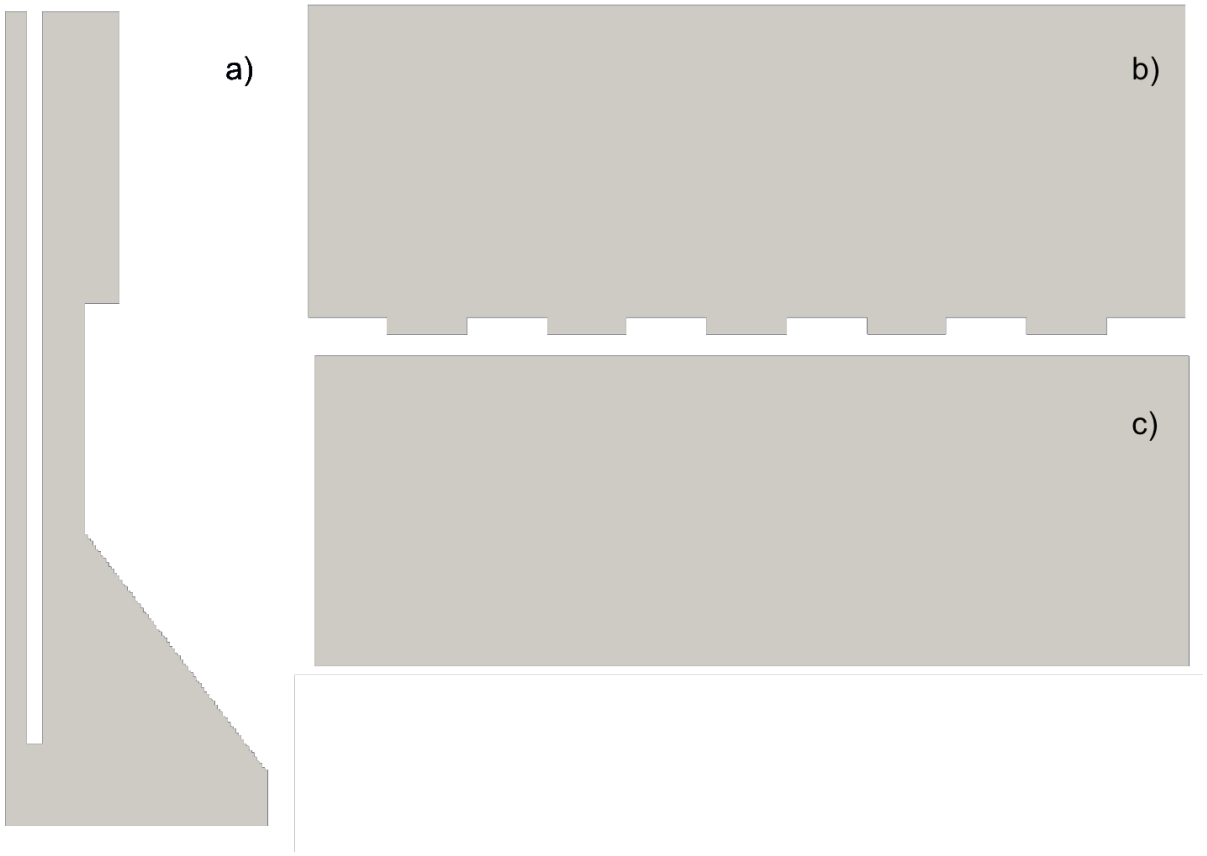


Figure 30: a) 2D axisymmetric domain of the impinging jet flow cell; b) 2D slice of the 90º and 0º patterned simulation; and c) 2D slice of the base case used for comparison. Grey portion is the fluid domain and white is the domain boundary.

The nominal geometry for the nano-scale patterned membrane used in the impinging jet flow experiments consisted of line gratings with a periodicity of 834 nm, depth of 110 nm and a line to space ratio of 1:1 (figure 30b). We examined the flow over these patterns using the domain depicted in Figure 1b. Its dimensions were 2000 nm height and 4587 nm length and width. The 90° case uses the left boundary (Figure 31) as the flow inlet surface (boundary condition), while the 0° case uses the front boundary as the flow inlet surface. The top is an open pressure and flow boundary that mimics the fact that this simulation is just a segment of a larger flow. The crenel extends down beyond the 2000 nm base. The base case (without patterns) is a

simple rectangle without the crenels and merlons at the bottom surface (Figure 31c). We incorporated the transition from bulk flow to this size pattern following figure 32.

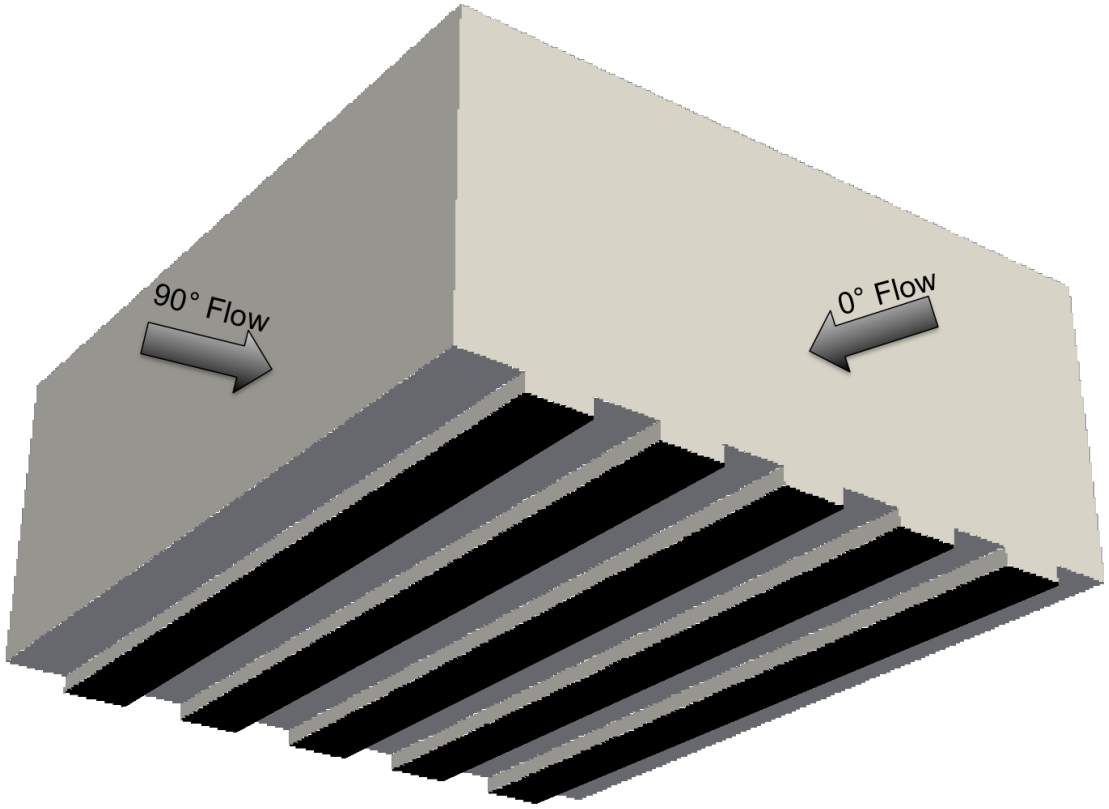


Figure 31: The 90º and 0º flow directions on the patterned mesh domain.

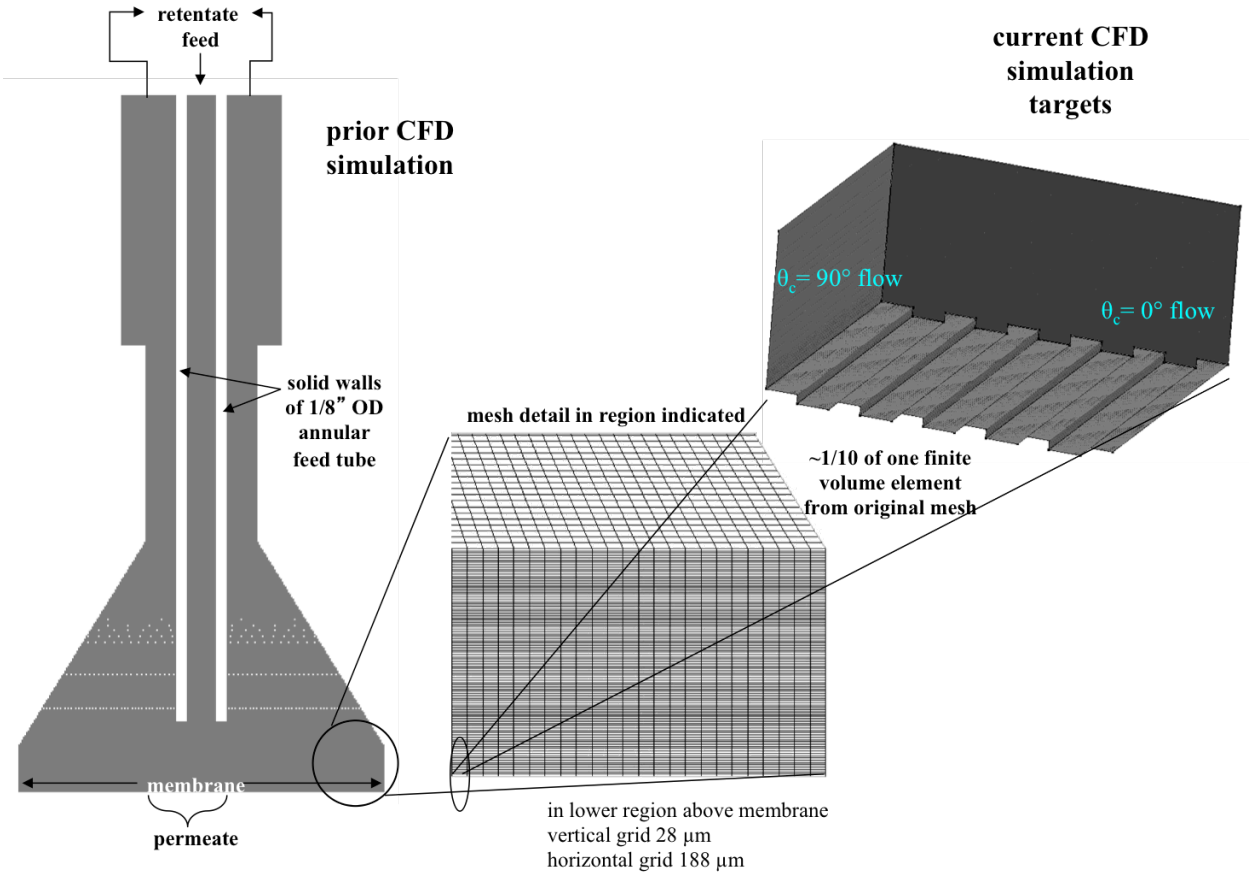


Figure 32: Impinging Jet flow cell transition to the nano scale patterned case of membrane surface.

3D cases were run for the 0°, 90° and base cases. Course uniform node spacing simulations were used to identify flow regions where high resolution was needed due to large gradients. These areas were near the membrane and pattern. The spacing and geometry was then organized to minimize CPU time cost for refined mesh runs. Periodic boundary conditions were used on the sides of the domain. Flow perpendicular to the periodic boundaries was below convergence criteria for the 90º and base case informing us that future runs could be done in 2D. All three meshes were generated in OpenFOAM with hexahedral mesh and have a non-orthogonality of 0. The base cases had 690,000 cells with a node's nearest neighbor an average of 10nm away (all internode distance refers to nearest neighbor). The 90º cases had a total of

2,864,400 cells with a node an average of every 10nm. The 0º case had 1,848,000 cells with a node an average of every 30nm. Both 0º and 90º in the crenels have mesh nodes every 10nm.

The cross flow cell fixture (Figure 33a) used in later experiments [3, 25] had a cavity at the top (for a stirrer bar to promote turbulent mixing) that was just left open (without a stirrer) to the flow in these experiments. This fixture's simulation domain was also created in GMSH but with prism meshing and has an average node spacing of 0.17 μm with a total of 3609275 cells. The non-orthogonality of this mesh is about the same as the other domains with an average of 5.6 and maximum of 33.7. In this cross flow cell fixture the membrane's pattern height and the Re were varied based on the specific experiments and had the Re's 120, 180, 239, 299 and 358 [4].

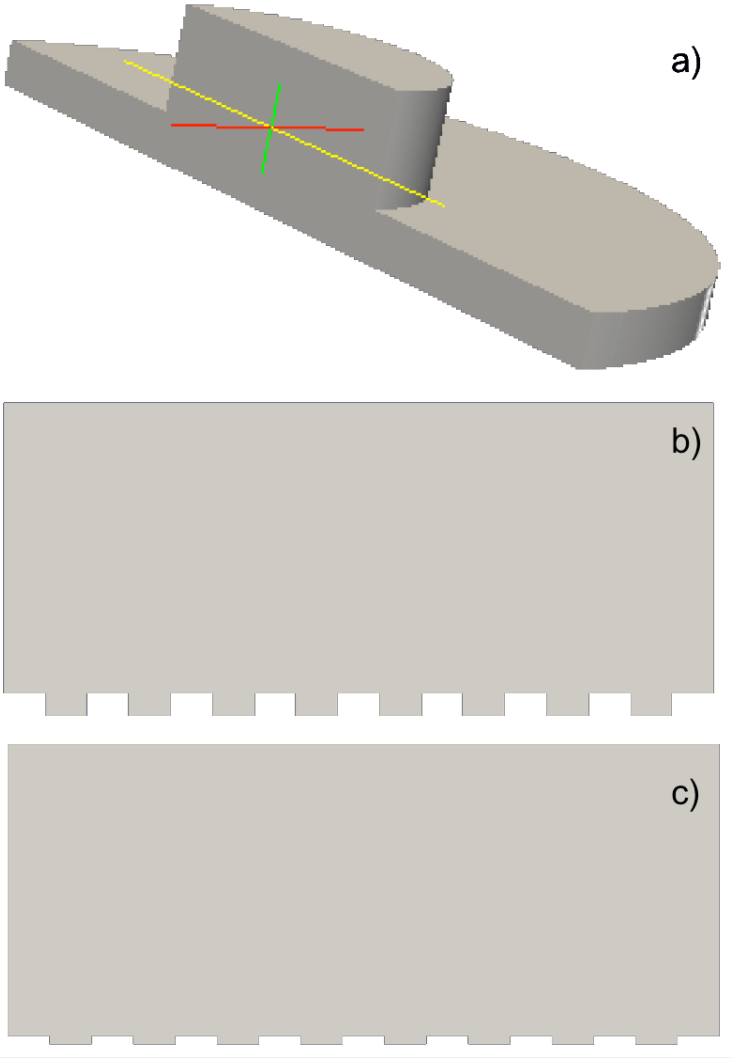


Figure 33: a) shows the 3D half-domain used in the cross flow cell. To reduce computational time, a symmetric plane boundary condition was used and the simulation was run over half the cell as shown. b) 2D slice of the membrane surface with the high pattern height. c) 2D slice of the membrane surface with the low pattern height. Grey portion is the fluid domain and white is the domain boundary.

Using the simulations from the cross flow filtration cell a linear inlet velocity profile was created by a best fit from the previous simulation (Figure 34 for meshing change). Like the other near surface simulations, periodic boundary conditions are used on the sides and the top is an open domain condition. For comparison purposes the number of crenels was chosen so that the overall length of the domain would be similar to the near surface simulations of the jet flow cell. There is five types of cases run here: a base case with no patterns, and both 90º and 0º with low

and high pattern heights (low is 60 nm, high is 160nm) (Figure 33b,c). High 0º has 1,052,250 hexahedral cells with an average density of 8,000 nm^3 near the surface. High 90º has 210,450 hexahedral cells with an average nearest neighbor node distance of 20 nm^3 near the surface. Base has 172500 hexahedral cells with an average nearest neighbor node distance of 20 nm^3. Low 90º has 204450 cells with an average nearest neighbor node distance of 20 nm^3. Low 0º has 1,022,250 cells with an average nearest neighbor node distance of 34 nm^3.

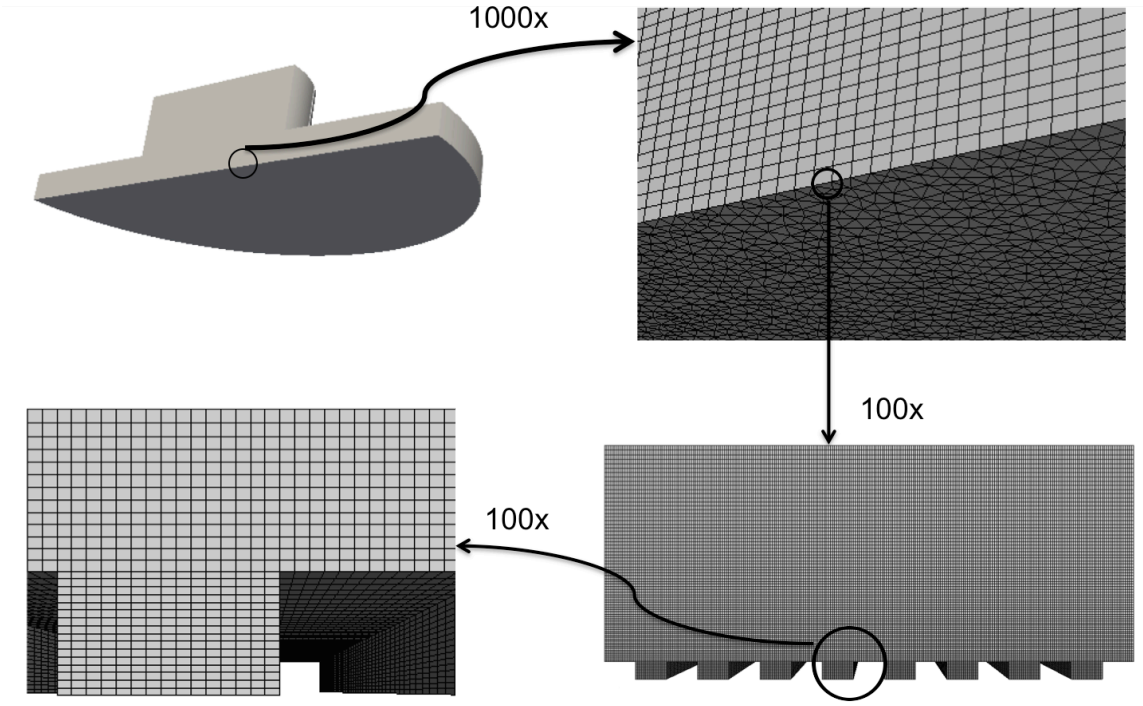


Figure 34: The magnification and progression of mesh from the flow cell to the near surface patterns.

Solver Specifications

The experimental results from an impinging jet flow cell system and a cross flow cell system are the subject of our studies. Two open source programs are used for creating the mesh geometry and running the simulations: GMSH[6] and OpenFOAM[7] (a C++ finite volume method library, with no graphical user interface (GUI), version 2.1.1 – Ubuntu pack). The simulations

6 GMSH, 2012 (http://geuz.org/gmsh/)

7 OpenFOAM, 2012 (http://www.openfoam.org)

were run on a parallel research computer cluster ("Janus")[8]. Membrane and filtration parameters that were studied are permeation rate (normal velocity through the membrane); pattern height; near surface, cross flow rate, and angle-of-attack. Two angles-of-attack were used: 0° and 90° to the pattern's merlons (figure 35). Generating a suitable mesh for periodic boundary conditions, with an angle-of-attack of 45°, has proved problematic thus far. (Note: heretofore angle-of-attack cases will be referred to simply as either 0° and 90°.)

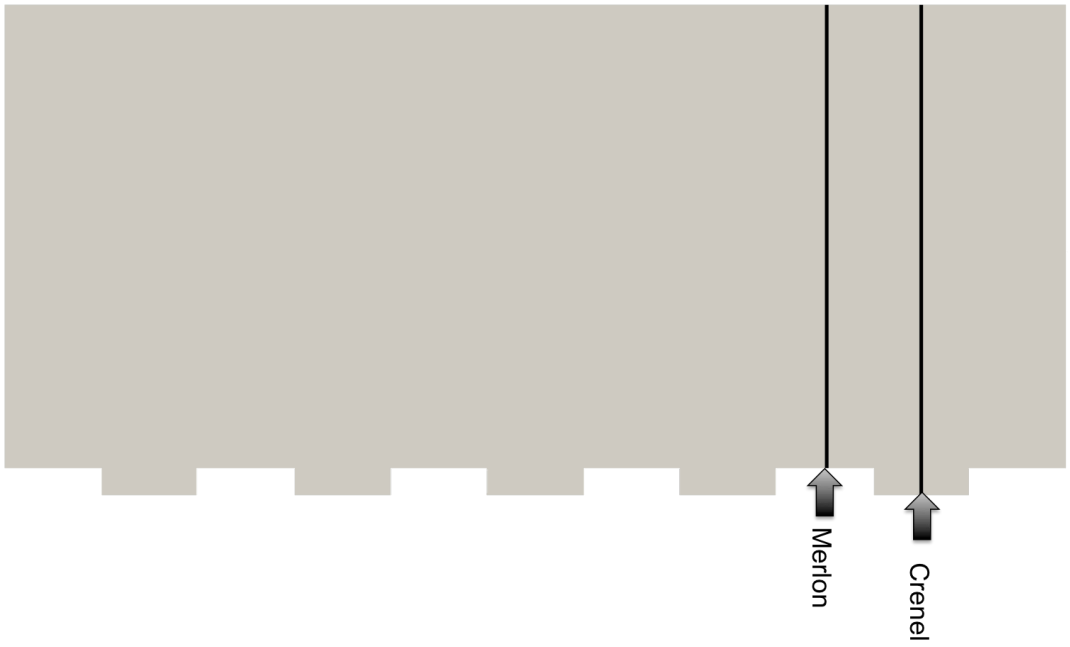


Figure 35: The Merlon is the non-depressed portion of the membrane. It has undisturbed permeability qualities. The Crenel is the low part of the membrane and here represents the compressed potion of the membrane that gives it the pattern. It suffers from a decrease in permeability caused by changes to the porosity during patterning.

Two solvers were used in the simulations: simpleFoam and pimpleFoam. SimpleFoam is designed for use as a steady state, incompressible turbulent flow solver. However the turbulent

8 This work utilized the Janus supercomputer, which is supported by the National Science Foundation (award number CNS-0821794) and the University of Colorado Boulder. The Janus supercomputer is a joint effort of the University of Colorado Boulder, the University of Colorado Denver and the National Center for Atmospheric Research.

component can be removed or turned off. SimpleFoam uses the SIMPLE algorithm[9], which is an iterative process that uses under-relaxation. PimpleFoam uses the PIMPLE algorithm[10], which is a hybrid algorithm of the SIMPLE and PISO algorithm[11]. PimpleFoam is a transient solver that is designed for large time steps and incompressible flow. It is a handy code in that you can set the Courant Number in the simulation and, thus, can exert finer control on the stability and speed of the solution and convergence. The Courant number, also referred to as the Courant-Friedrichs-Lewy condition, states that the distance traveled by a particle in a cell should not exit the cell in a given time step. The Courant number (equation 1.17) in a transient simulation should always be below 1.

$$C = \frac{u_x \Delta t}{\Delta x} + \frac{u_y \Delta t}{\Delta y} + \frac{u_z \Delta t}{\Delta z} \tag{1.17}$$

Steady state solvers were run to convergence except when unable to due to flow in the tertiary direction (perpendicular to periodic boundaries) being a number smaller than convergence levels but not fractional levels of change. On the other hand, when using the transient solvers it is often not possible to run for several domain residence times because of the computational time costs. Thus, we first ran a steady state solution initially and then performed a transient analysis to observe if there are any transient effects in the fully developed system. In general, the only transient effects observed were in the 0° case where the lack of fixed boundaries allows for time-dependent sideways oscillations transverse to the main flow direction. And now the details I promised you earlier.

---

[9]https://openfoamwiki.net/index.php/OpenFOAM_guide/The_SIMPLE_algorithm_in_OpenFOAM

[10]http://openfoamwiki.net/index.php/OpenFOAM_guide/The_PIMPLE_algorithm_in_OpenFOAM

[11]https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM

In the “fvSchemes” file the different finite volume discretization schemes are selected. There are seven different categories of schemes that need to be selected: time, gradient, divergence, laplacian, interpolation, surface normal gradient, and flux requirement. For the time scheme “steadyState” is specified for steady state systems of course, while with the transient system CrankNicolson with a value of 0.5 was used. With the interpolation scheme linear interpolation of velocity is specified while with under the flux requirement scheme no fluxes are required. The “snGradSchemes” is used to solve the Laplacian term by Gaussian integration. “Explicit non-orthogonal correction” (OpenFOAM users guide) is chosen as the condition for the surface normal gradient scheme. cellMDLimited Gauss linear 0.5 is specified for pressure and velocity for the gradient scheme, while Gauss linear corrected for the laplacian schemes. In divSchemes for the velocity component listed as div(phi,U) bounded Gauss linearUpwind grad(U) is specified but all other conditions can be set to Gauss linear.

In the “fvSolution” file algorithms, tolerances, and linear equation solvers are specified. This is one of the more important files because it details not only convergence criteria but also the actual mathematical techniques that will be used to arrive at the numerical solutions. There are three categories in this file: solvers, algorithm, and relaxation factors. There are four solver categories: pressure, pressure final, velocity, and velocity final. Depending upon your running conditions, the quality of the mesh, algorithm, and your discretization scheme, different solver options are useful. For simplicity and briefness I will just describe the settings I used and additional information on the rest can be seen in (insert citations). For pressure, the Generalized Geometric-Algebraic Multi-grid (GAMG) solver was used. It is the optimal OpenFOAM pressure choice for parallelized systems with less than 1024 processors at which point the Krylov type solvers (PBiCG and PCG) tend to do better. In fact PBiCG (preconditioned bi-conjugate

gradient) solver works fairly well for the velocity matrix manipulation. With the GAMG solver I used the DICGaussSiedel smoother with 100 cells in the coarsest levels and a tolerance and relative tolerance as 1e-11 and 0 for the final conditions. DIC (simplified diagonal-based incomplete Cholesky smoother for symmetric matrices), GaussSeidel (Gauss Seidel method is a technique used to solve a linear system of equations. The method is an improved version of the Jacobi method. Convergence is only guaranteed if the matrix is either diagonally dominant or symmetric and positive definite) are combined into the DICGaussSeidel model which is an effective smoother that runs DIC and then Gauss Seidel to smooth any irregularity peaks from DIC. Lastly for GAMG, the number of cells in the coarsest level determines that amount of smoothing that goes into the pressure. This number should be carefully chosen based off the density of mesh and the desired resolution. It corresponds to the number of cells that are temporarily merged together for a coarse pressure calculation before being fine-tuned for each of the cells in the domain. The velocity condition PBiCG is much simpler and just requires a preconditioner instead of a smoother, which I chose DILU (Simplified diagonal-based incomplete LU smoother for asymmetric matrices). Lastly, for velocity we chose a final tolerance of 1e-10 and relative tolerance of 0. This leads to the algorithm control. There are two key factors here: the Non Orthogonal Correctors which can be left as 0 for mesh geometries was a maximum mesh non-orthogonality less than 40; the second is the residual control. These numbers are the final convergence criteria. They need to be smaller than the individual pressure and velocity tolerances because otherwise the solution cannot converge. I used 1e-9, which means that my accuracy affectively is around 6 to 7 significant figures. Lastly the relaxation factors affect the stability of the solution and the speed at which it converges. For duplication I use 0.3 for pressure and 0.5 for velocity.

Metrics

Next I need to cover the metrics we used. Our metrics of examination are the shear rate $(\dot{\gamma})$, the magnitude of the gradient of the shear rate $(|\nabla\dot{\gamma}|)$, and the non linearity term (ξ_{nl}). Over all the patterned cases we examine those quantities. A wide variety of literature focuses on the shear rate and its contribution. The magnitude of the gradient appears in literature but is often dismissed by itself as not contributing much. After our work we would propose that with patterned membranes unknown mechanics that depend upon it play a more significant role. Meanwhile, the nonlinearity term which we had theorized played a significant role, instead only describes one of the phenomena changes.

Bulk Jet Flow Results

The first experiment done used the impinging jet flow cell. Its purpose was to measure the effect of angle of attack and determine the critical flux along with other properties. As such it was the first system that we simulated and gathered initial data from. I started off simulating the entire flow cell domain with a radially symmetric simulation to figure out starting points for the nano scale simulation over the pattern. Due to the size difference of the whole cell vs. the patterns it is computationally impossible even with the largest supercomputer to simulate directly. As such our simulation focuses on using it as a set up for the fouling cases. Figures 36 and 37 show the results in the relevant areas from the impinging jet flow cell. There are no recirculation values near the membrane and in fact it is heavily parallel flow near the bottom. Table 2 will summarize the initial boundary condition data we collected from data fits.

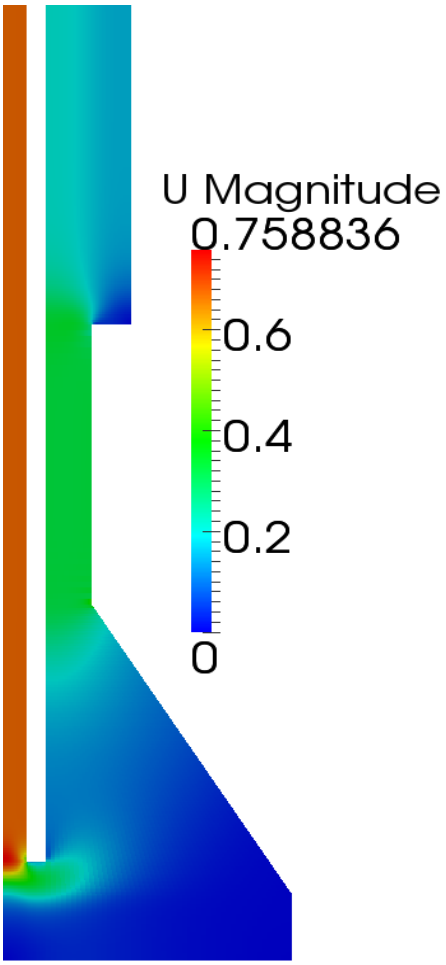


Figure 36: Shows the magnitude of the velocity over the whole domain of the impinging jet flow cell. There are no signs of recirculation in the domain and fluid flow is very standard.

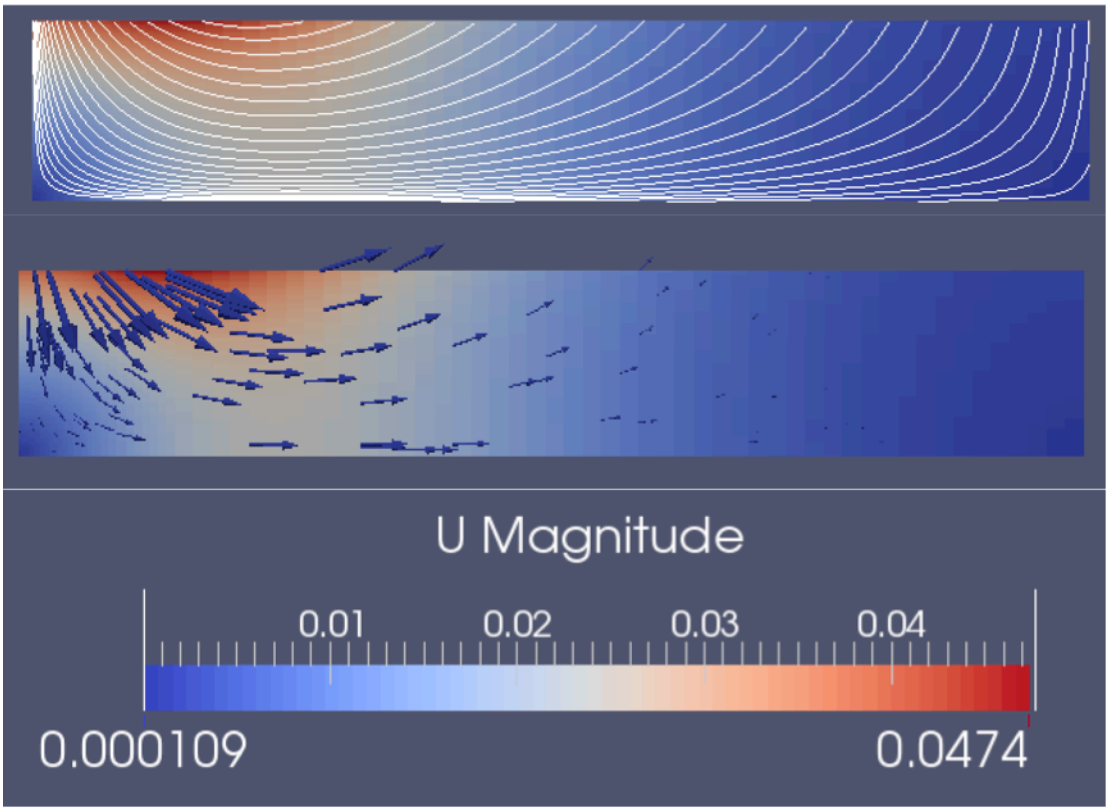


Figure 37: Shows the streamlines and the velocity vectors in the area right above the membrane.

Impinging Flow Simulation, Velocity Values from 5 Points (140 µm) Above Membrane	
Maximum Radial Velocity (cm/s)	Permeate Velocity (µm/s)
2.335	23.6
2.330	16.7
2.329	15.8
2.322	8.3
2.312	0.0

Table 2: Shows the maximum radial velocities as a function of the permeate boundary condition values. These radial velocities were used to help create the inlet condition for the detailed mesh simulations.

Nano Jet Flow Results

As with the bulk flow, I'll focus on showing the results over the whole domain first. As mentioned shear stress is the component that I focused on this stage. To make things clear, the shear rate described will always be in the flow direction with respect to distance away from the surface of the membrane. Figure 38 shows the normalized shear stress ($Pa/kg \bullet m^3$) for the base

case, 0 degree case and the 90 degree case with no permeate. Only one permeate rate is shown because as I show later, permeate plays a minute role in the change in fluid dynamics. To describe the particular flow differences I have plotted in figure 39 slices along the crenel, merlon and from the unpatterned simulation.

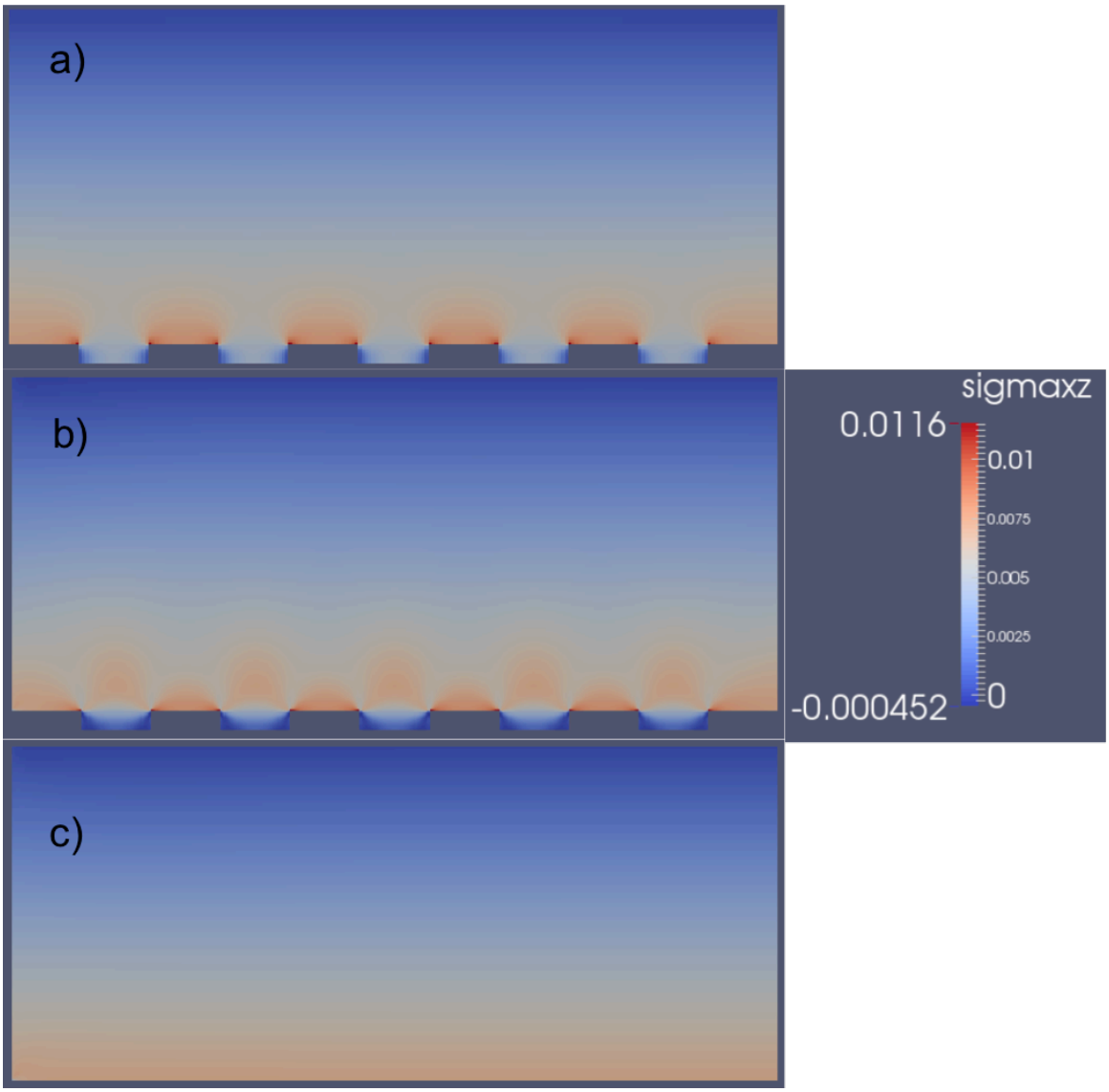


Figure 38: a) 0 degrees flow case where the flow direction is shear stress (yz), but is on the same scale as the one shown. b) Shows the 90-degree flow case where the shear stress is xz. c) Shows the base case without any patterns and the shear stress is along xz.

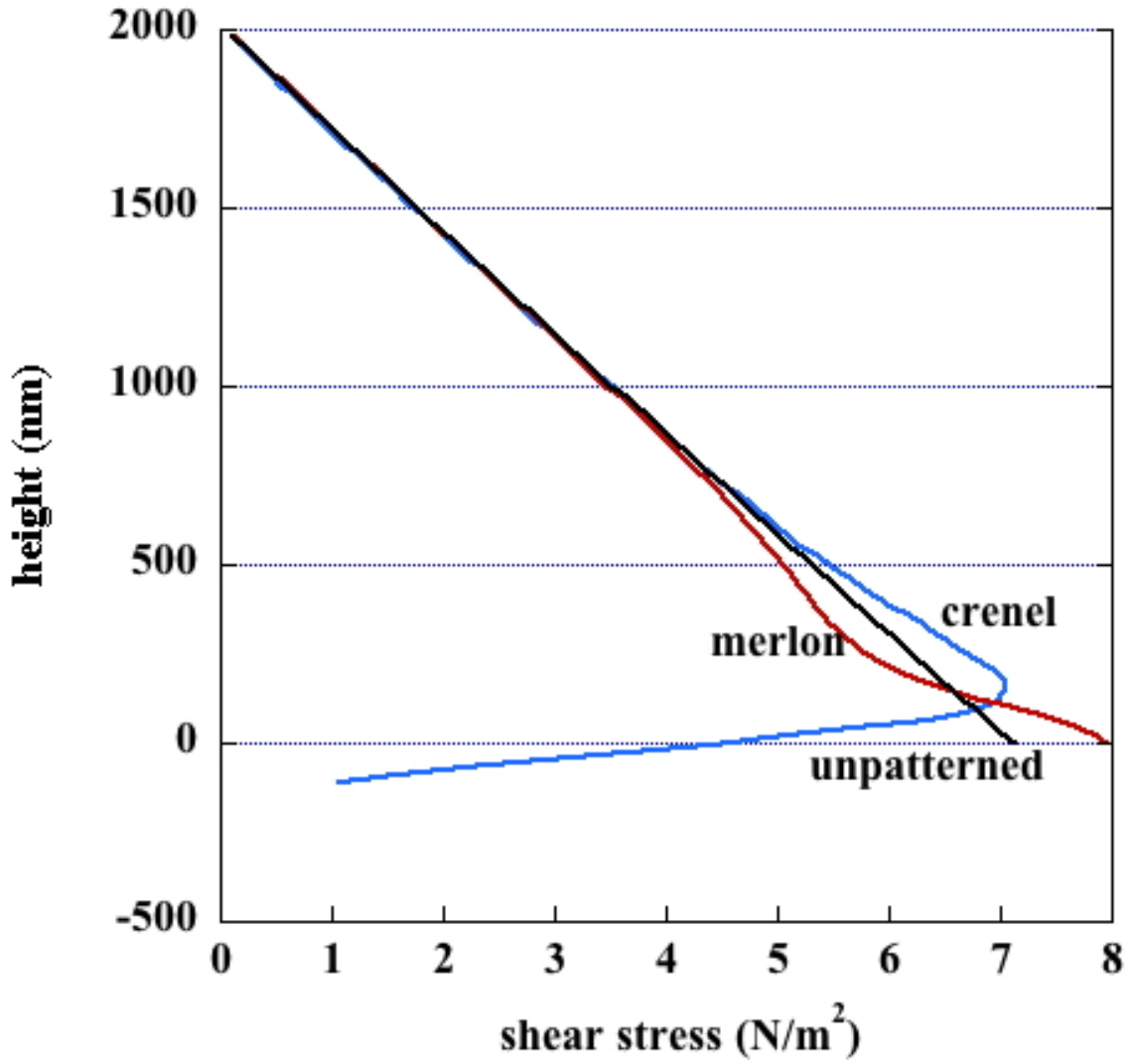


Figure 39: Shows how the merlon and crenel deviate from the unpatterned membrane. These are the fundamentals of the changes brought about by the patterning of the membrane. This is from the 0 permeate 90 degrees case.

The influence of permeation rate and angle-of-attack on the near surface flow of the NIL-imprinted membrane was studied in this enhanced view simulation of the jet flow cell. Based on previous literature results the three examined quantities are: shear rate [4, 7, 39-43], gradient of the shear rate [37]and the nonlinearity term[40, 41]. Results for near surface flow field (0 to 400nm) indicate that the permeate rate had negligible effects (Figure 40) on any of the three physical quantities correlating with fouling reduction (Figure 41). The figures show that the permeate rate does not significantly affect the shear rate, gradient of the shear rate, or the non-

linearity term. It does show that angle of attack does correlate with the gradient shear rate and the non-linearity term, but that gradient shear rate matches best so far (Figure 42).

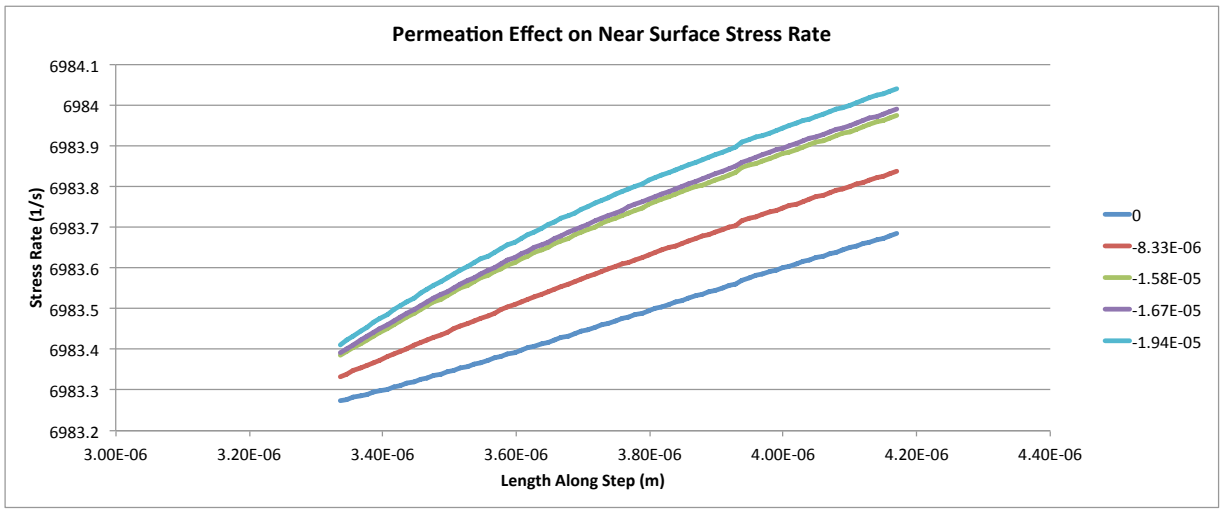


Figure 40: Flat membrane strain rate along the surface vat different permeate rates.

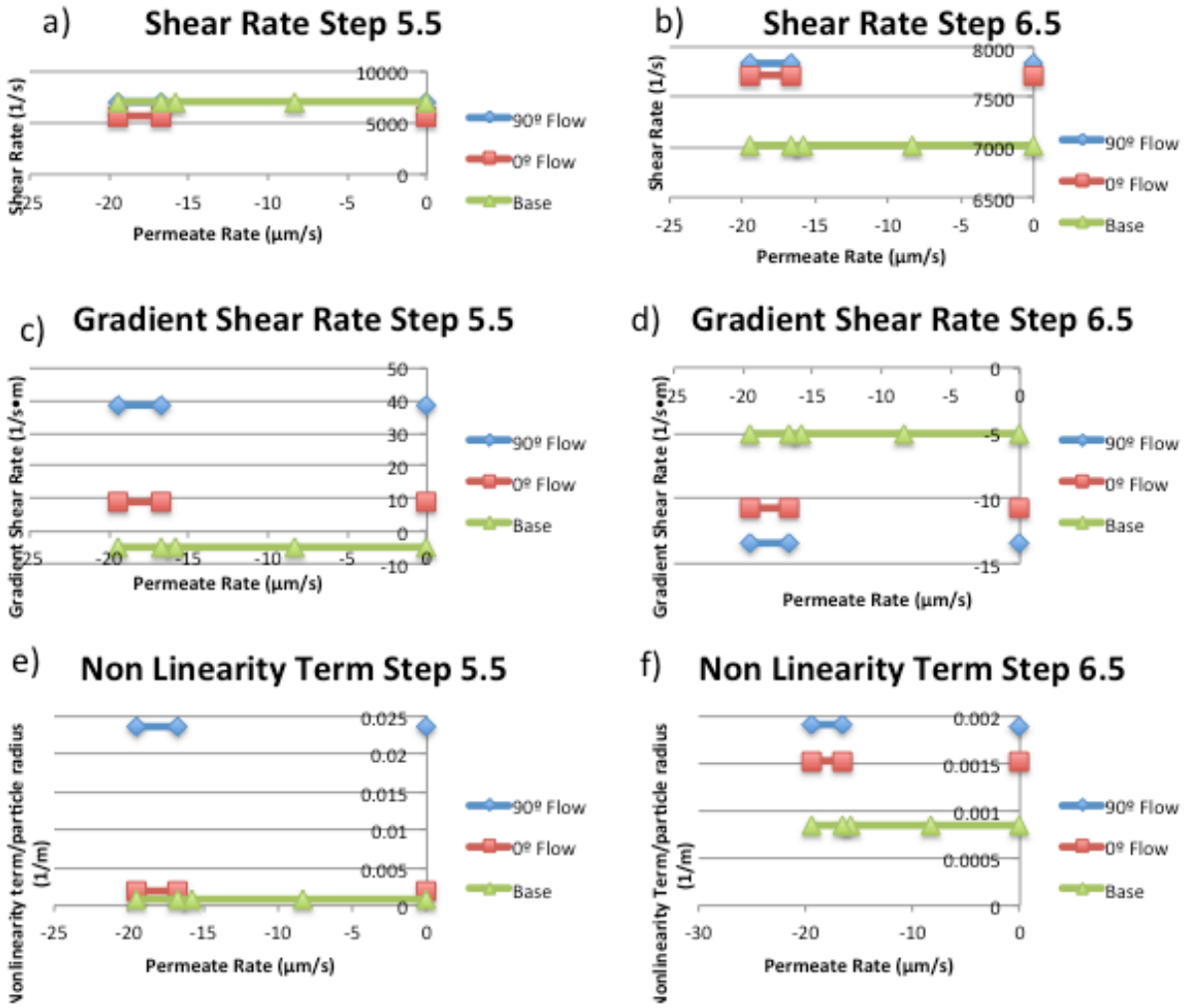


Figure 41: Impinging jet flow cell's summary of maximum data points on the merlon and crenel.

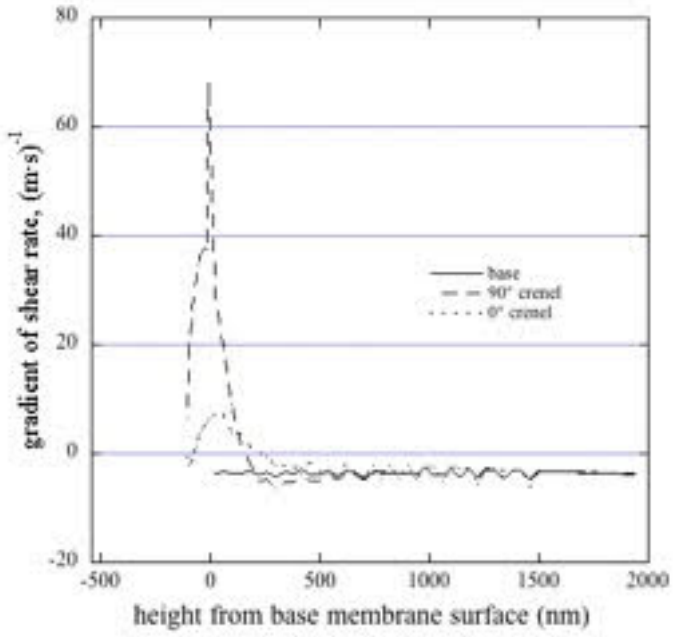


Figure 42: Summary of jet-flow membrane cell's gradient of strain rate vs. the height from the membrane. Results when varying the permeation rate would not be visible on this plot, so the no permeation case was plotted here.

Strain rate values were taken across the whole length and at four equal spacing's near the surface from 0 to 300 nm. Altogether 80 points were used to create this average. Figure 43 indicates that the permeation through the membrane has negligible influence on the average shear generated in this region, and that the flat membrane (without a pattern) had a higher average shear than even the 90-degree case, which appeared to have the least fouling by particle deposition. These averages make sense when you take several slices of the shear rate horizontally. I have done such a thing. Figure 44 is at 1nm above the membrane surface while Figure 45 is 150nm above the surface. Notice how while the base is lower than either's maximum its higher than the average height of the shear rate for the 90 and 0 degree cases.

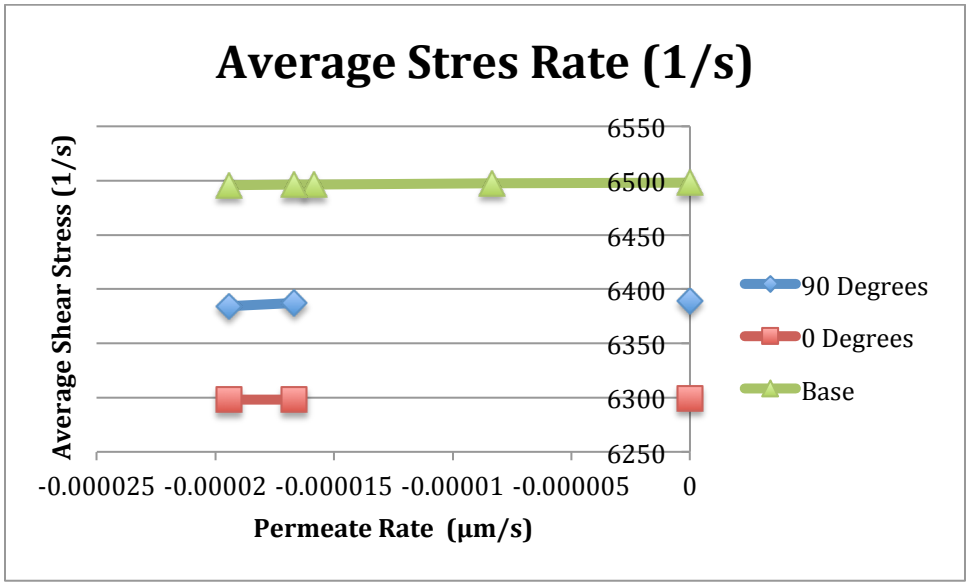


Figure 43: Average near-surface (0-400nm) strain rate for the different permeate rates.

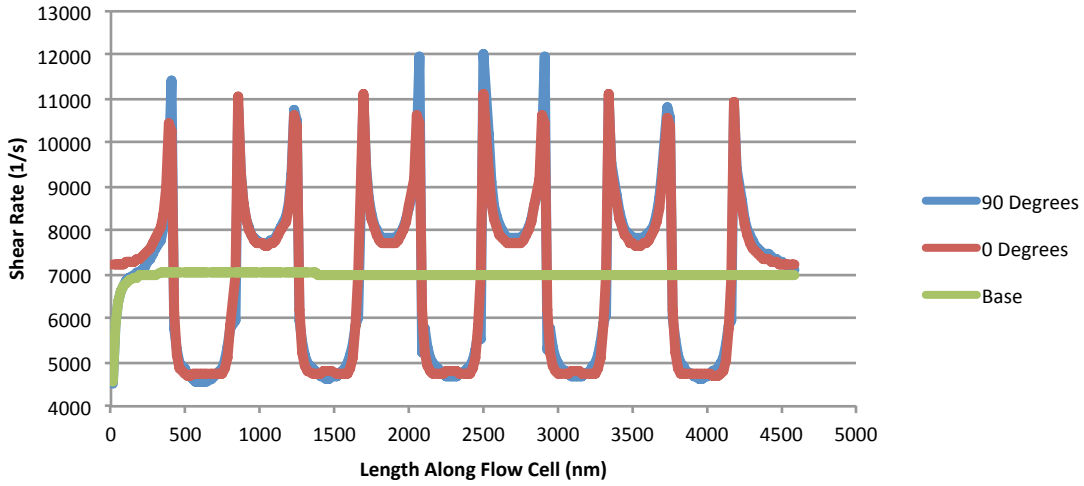


Figure 44: 1nm above the surface of the membranes for the permeate cases from the jet flow cell.

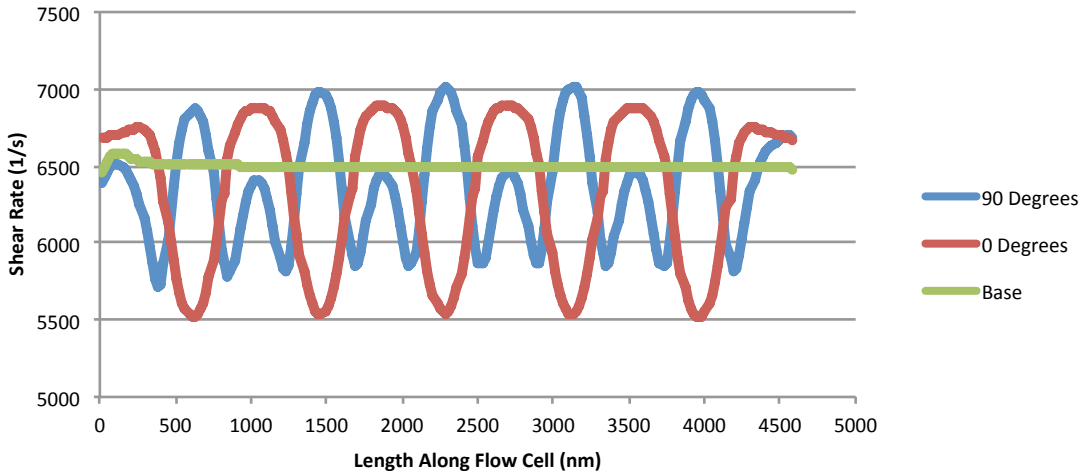


Figure 45: 150nm above the surface of the membranes for the permeate cases from the jet flow cell.

Bulk Cross Flow Results

Unlike the impinging jet cell, the cross flow cell had a particularly large recirculation at the top of the domain inside the stir bar area (no stir bar actually present, so it is just a cavity for one). However, it is certainly a much more complicated flow system and at higher Re values that the impinging jet flow cell. However, they aren't as smooth and it looks like the flow near the surface of the membrane is actually much lower than the impinging jet flow cell. Figures 46-50 are various cross sections of the domain to describe what is going on.

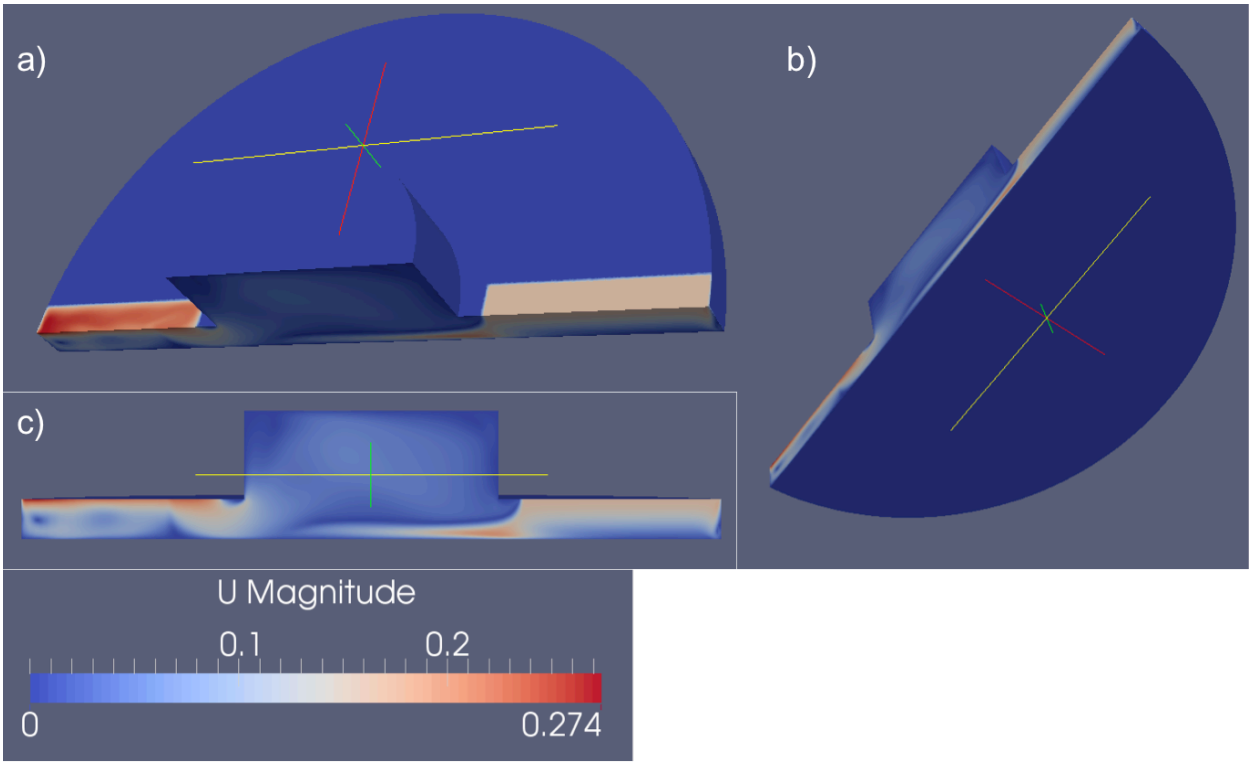


Figure 46: 3D shots of the simulated cross flow cell. The flat edge as mentioned is the symmetry plane boundary. The second important feature is to see the location of the inlet and outlet. The inlet is on the right while the outlet is on the left.

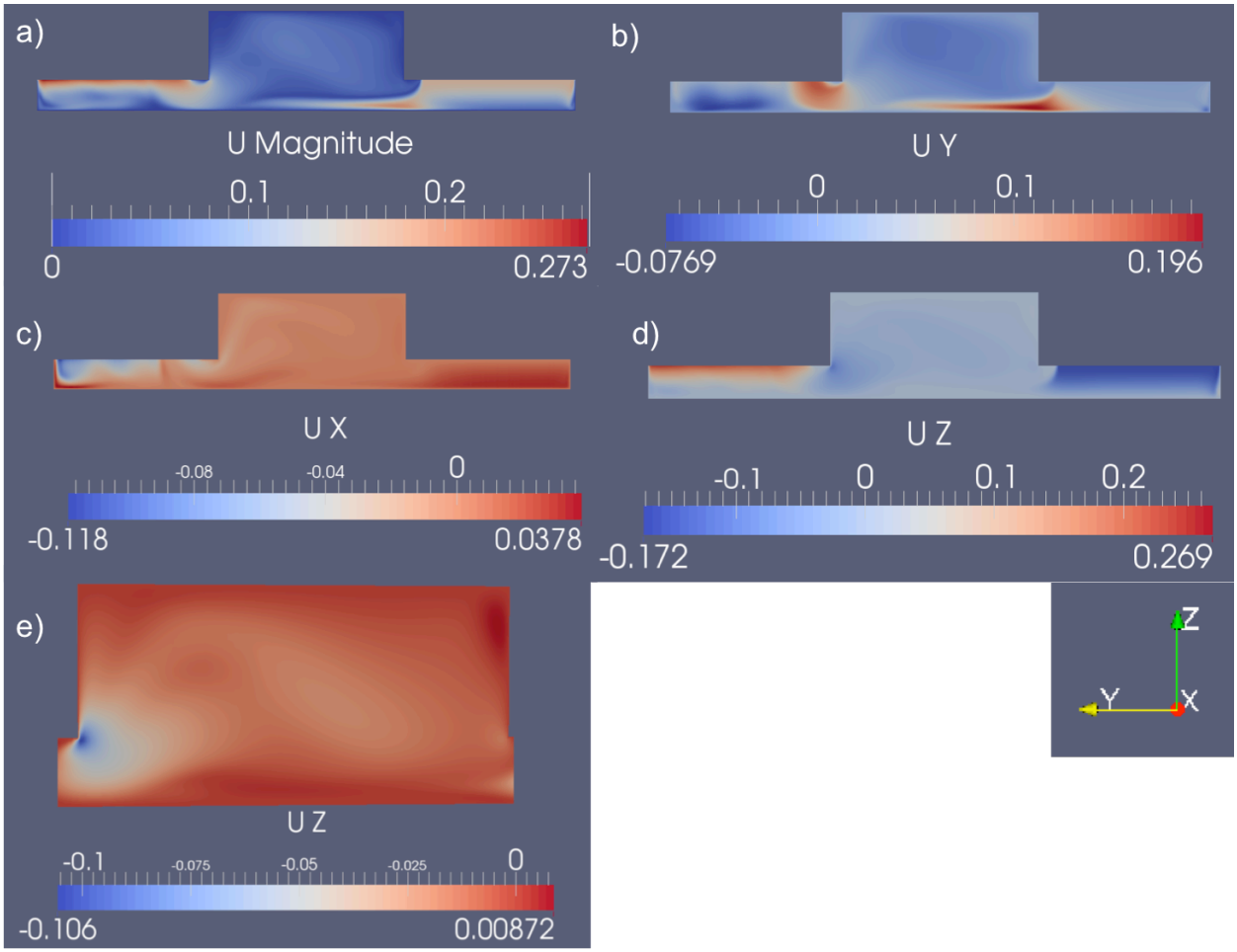


Figure 47: in all pictures: the right is the inlet, the left is the outlet. In addition, this is a slice of the flow cell in the center from entrance to exit. a) velocity magnitude b) velocity moving right to left is positive. Notice the recirculation and the unsteady flow. c) Due to the fact this slice is slightly off center, the into board direction isn't zero but it does help show the instabilities. d) the vertical direction e) zoom in of the cavity to show the instabilities.

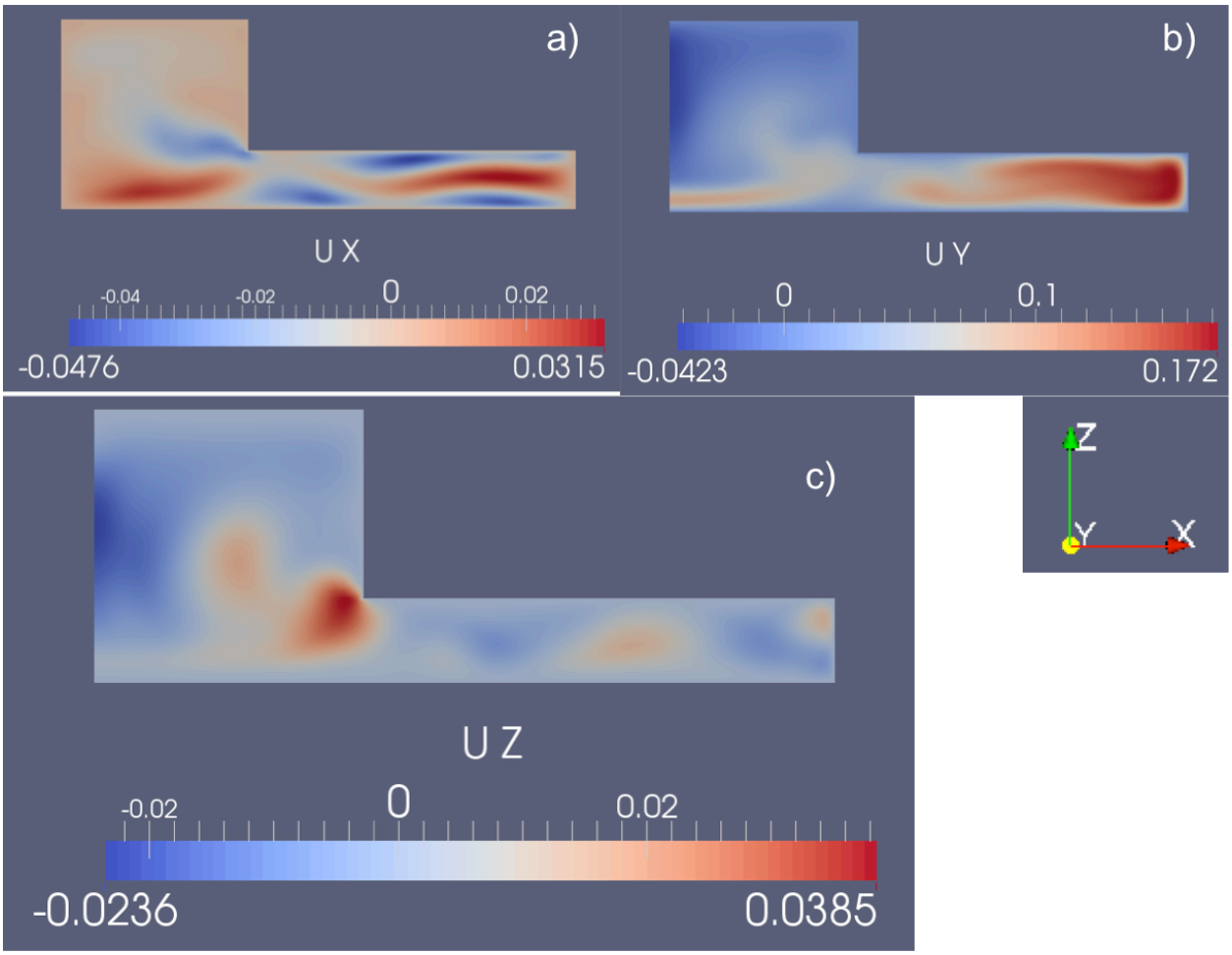


Figure 48: Showing the instabilities. This is a sidewise cut that allows you to see the flow along the circular cut. a) the horizontal component of velocity isn't stable as you see b) the forward is pretty stable but it accelerates along the edges of the flow cell c) mixing is evident

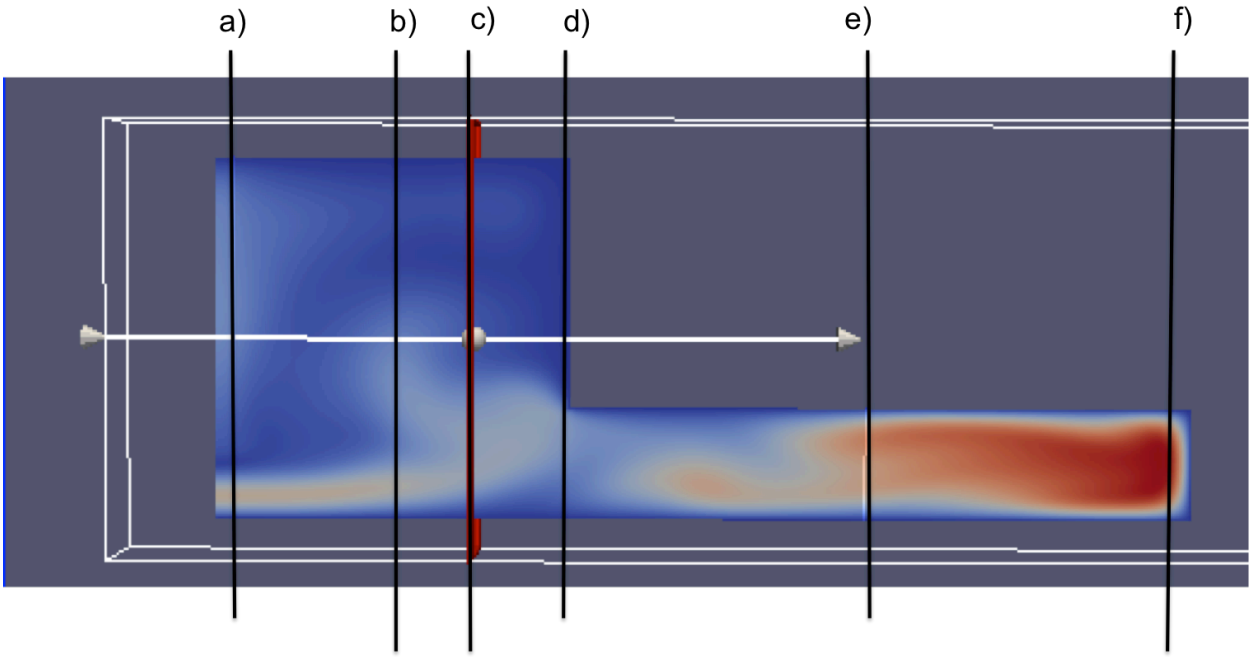


Figure 49: taking a cross section and looking at the velocity profiles along the lines. The letters correspond to the graphs in figure 50.

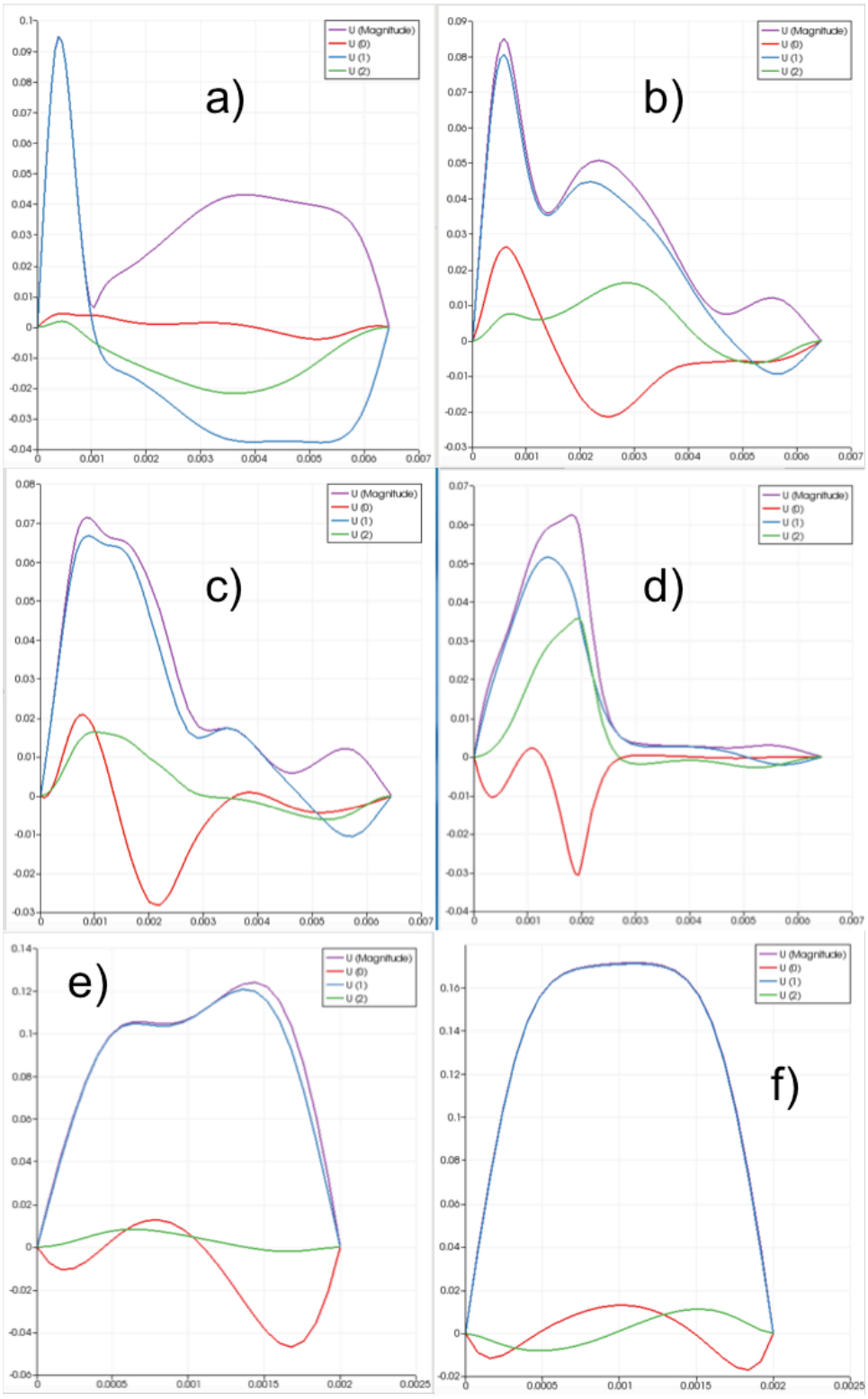


Figure 50: the velocity profiles along the lines from figure 49. Purple is velocity magnitude. Red is perpendicular to the primary flow direction. Blue is the primary flow direction and is along the y axis (entrance to exit). Green is the vertical direction. For all the graphs the horizontal axis is height and the vertical axis is velocity.

The cross flow filtration cell had several recirculation areas and in general parabolic flow, although nowhere was it full developed and it was strongly irregular in the center where the stir bar would have been. The cross flow was fit with a 3rd and 4th order polynomial going through 0 for each Re (Table 3). Those were then not used because they were too inaccurate near the surface of the membrane. So instead best fits using the bottom to points forming a line were used (Table 4). Two locations where initially examined (dead center of the cell) and center of side. Eventually slice e from figure 49 was chosen to be the inlet boundary condition location. The flow cell unfortunately shows that there is already a high degree of mixing in the cell and that near-surface flow, while more stable, is not absolutely laminar over larger distances.

Case	Outer Edge	Center
Case430	1e4(-4.4352z^2+0.0089z+0)	3.35935e5*z^3+2399*z^2+0.03606*y+4.65e-7
Case431	z^2*-5.327e4+138.2z+2.213e-6	z^3*-9.508e7+4.545e4z^2+61.61z+0.0006061
Case432	x^2*-8.37e4+208.2z-9.819e-7	2.591e11*z^4-5.789e8z^3+2.05e5x^2+107.5z+0.001356
Case433	z^2*-1.164e5z^2+252.3z+5.218e-6	z^3*-6.968e8z^3+2.13e5z^2+195.32z+0.002248
Case434	z^2*-1.566e5+270.6z+7.678e-5	z^4*-5.379e11 - 7.663e8*z^3 +229800*z^2+293.6*z+0.00335

Table 3: The original R>.98 polynomial for the near the membrane side

	Outer Edge
Case430	81.4*z
Case431	137*z
Case432	207*z
Case433	251*z
Case434	270*z

Table 4: The eventually used linear fit from the bottom two points.

Nano Cross Flow Results

In this section we finally have direct experiment comparisons to simulations. Once again we examine the simulations with regards to the shear rate, gradient of the shear rate and the non-linearity parameter. Although we previously found no correlation with the shear rate, there was the possibility of it matching with either pattern height or Reynolds number. Fist up is the domain slices showing the shear rate for the 5 separate cases. Figure 51, 52, and 53 pair the high and low simulations to emphasize visual distinctions between the pattern heights. There is a

fairly big difference between Figure 51 and 52 in that the 90 degrees case has significant shear rate over the crenel.

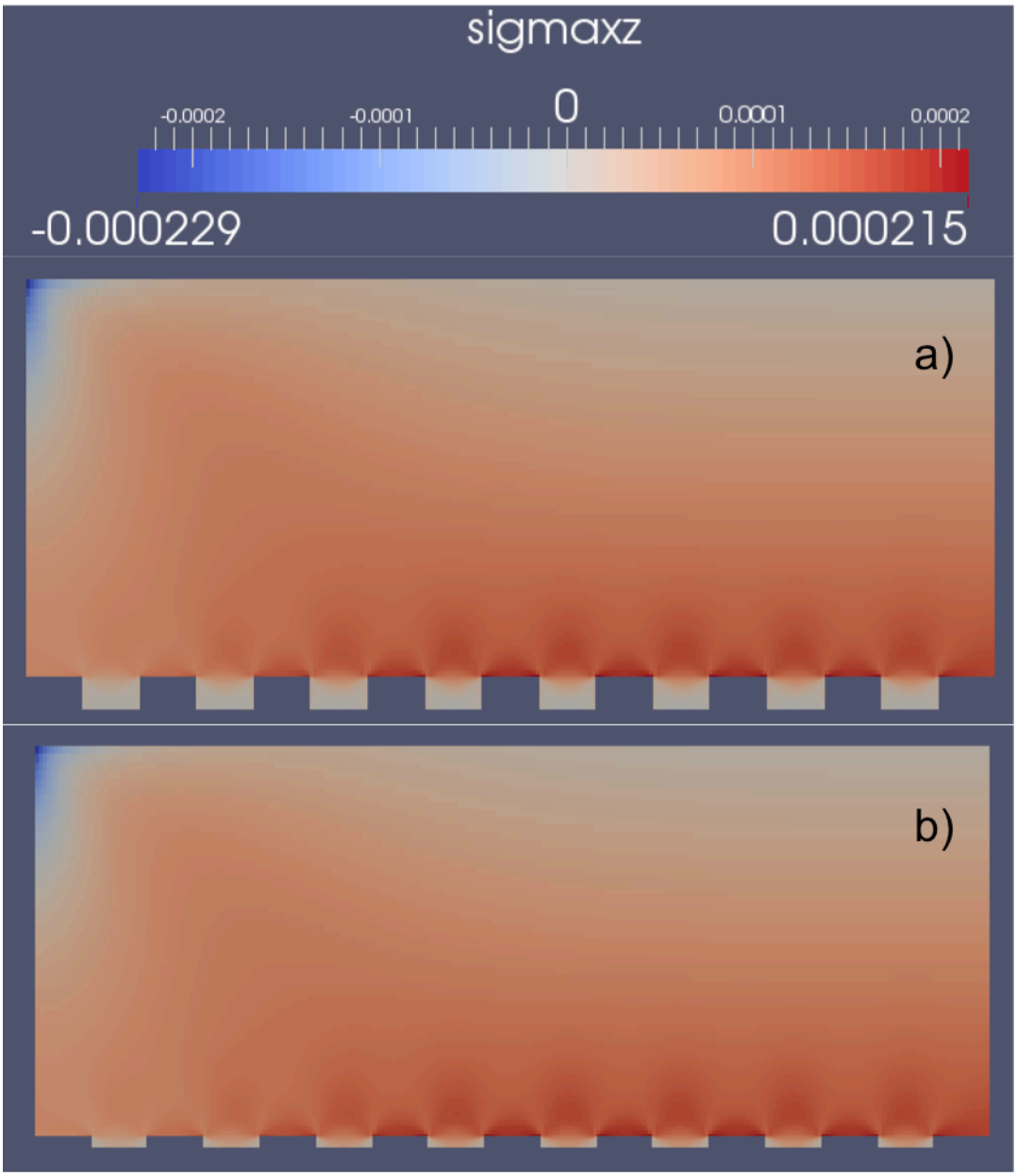


Figure 51: Re 120 90º Cases a) is the high pattern case b) is the low pattern case. They actually did not have significantly different shear rate profiles.

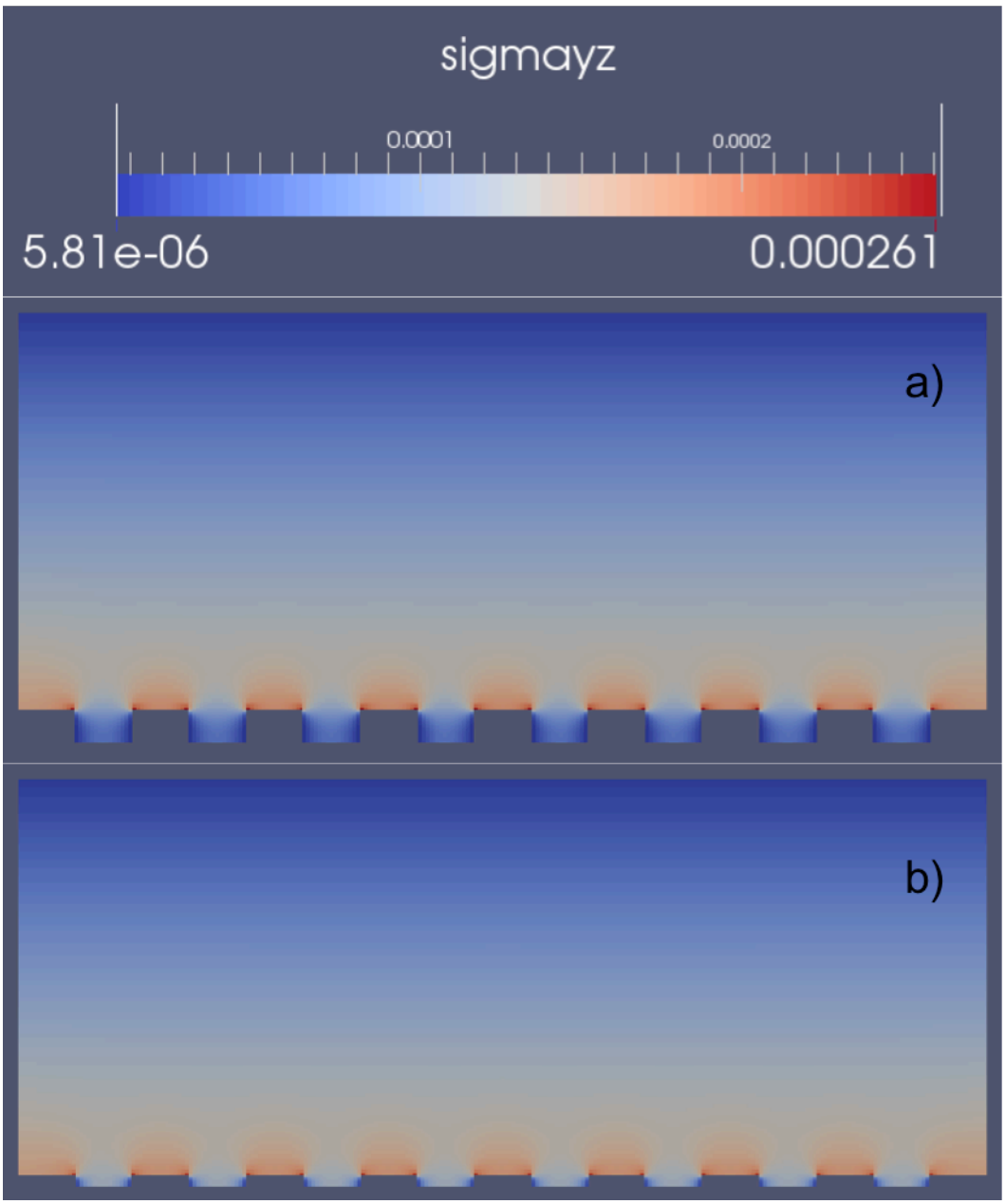


Figure 52: Re 120 0º Cases a) is the high pattern case b) is the low pattern case. They actually did not have significantly different shear rate profiles.

Figure 53: Re 120 for base case.

Figures 54 and 55 are complete summaries of the maximum values along given lines. By tabulating the data like this I was able to observe the trends along different conditions. Seeing all these conditions I was only able to conclude that the gradient of the shear rate best described the angle of attack and the Reynolds' Number. However, in Figure 56 I was able to see that pattern height is governed by the non-linearity term. Choosing those that fit the most accurately, I assemble the comparisons with the experimental work done by Maruf.

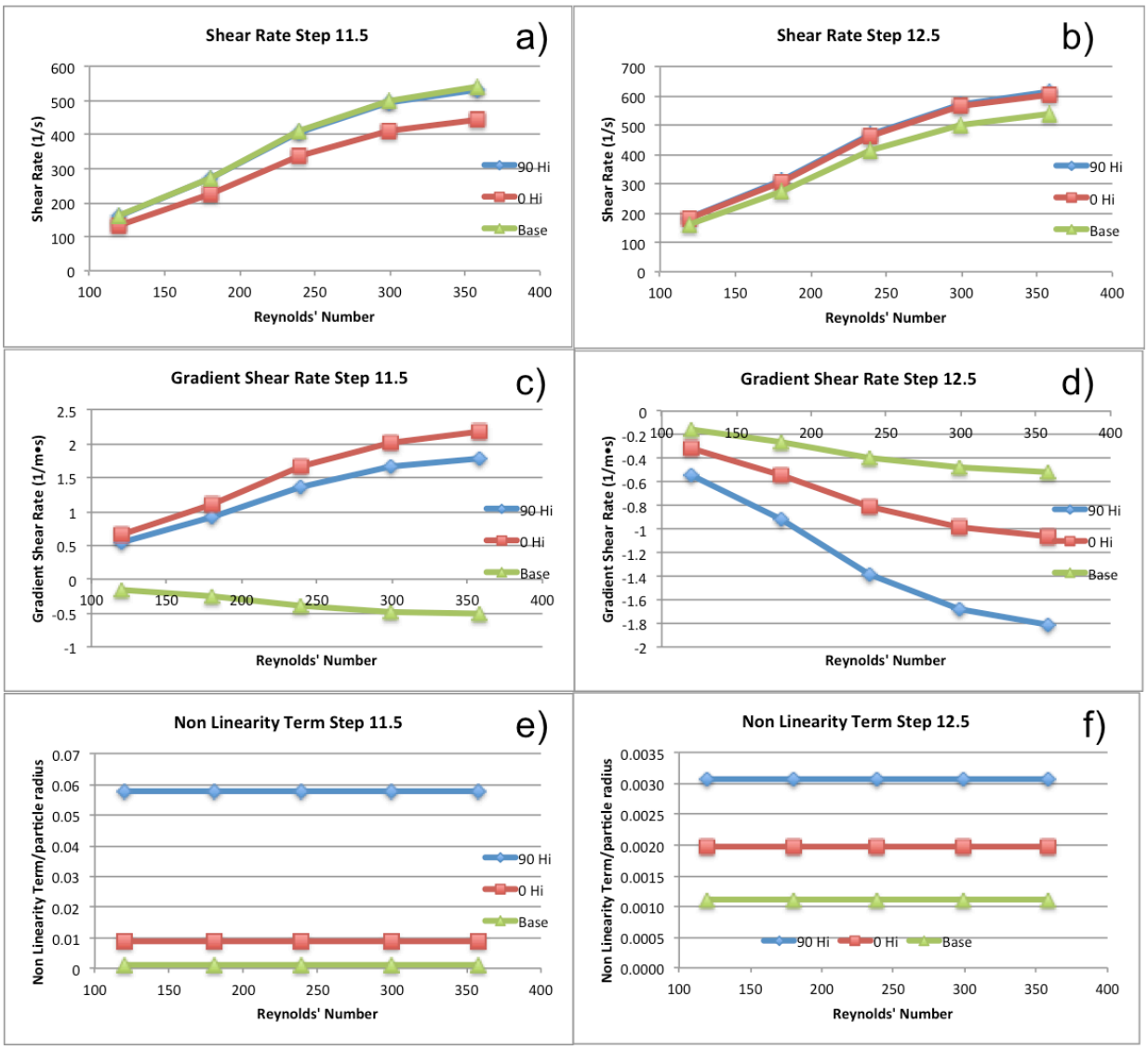


Figure 54: A summary of the cases from the High cases. Blue is 90º, Red is 0º, and Green is no pattern. The horizontal axis is Reynolds' number while the vertical axes are the respective titles. a) Shear Rate Step 11.5 b) Shear Rate Step 12.5, c) Gradient Shear Rate Step 11.5, d) Gradient Shear Rate Step 12.5, e) Non Linearity Term Step 11.5, f) Non Linearity Term Step 12.5

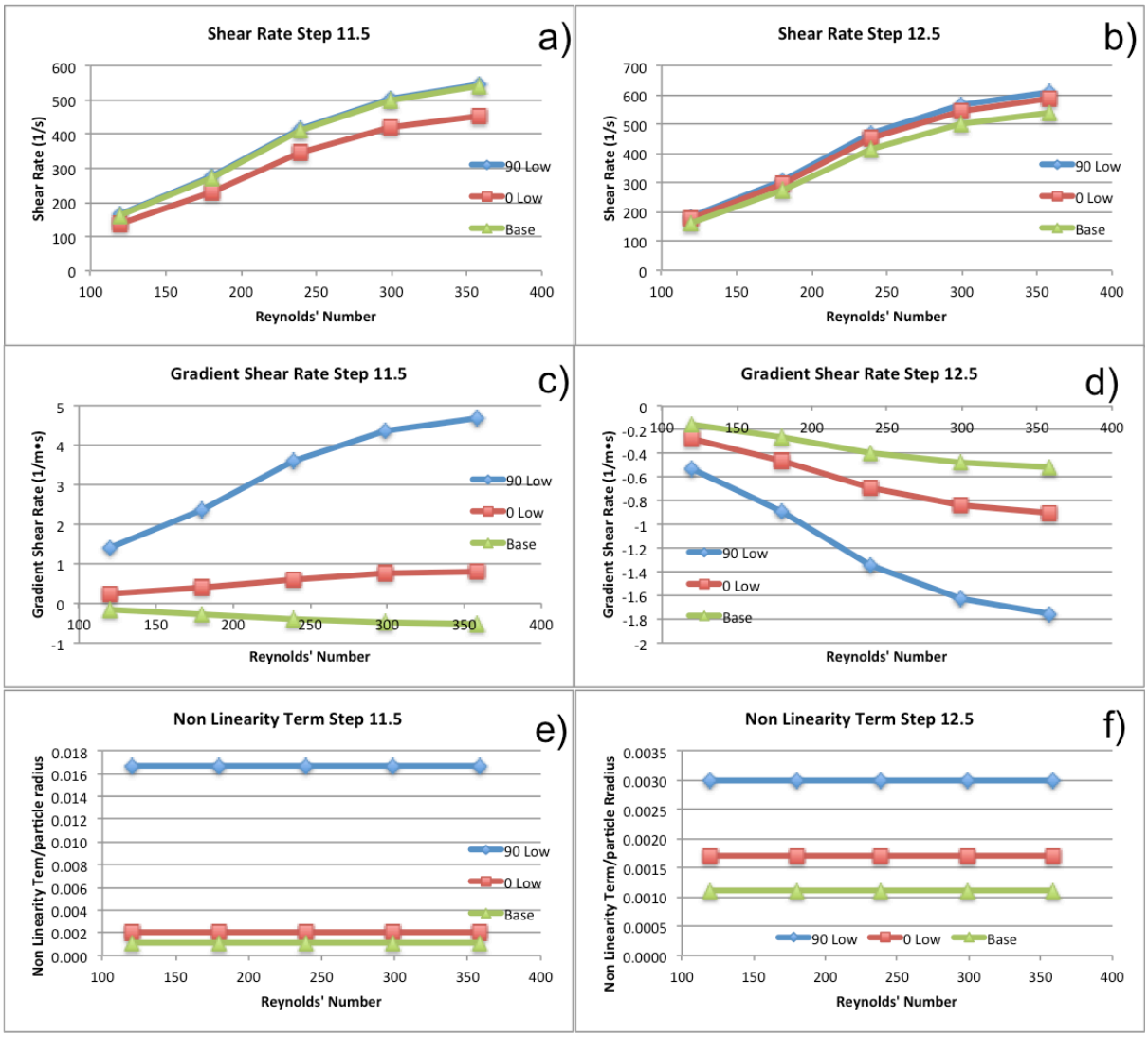


Figure 55: A summary of the cases from the Low cases. Blue is 90º, Red is 0º, and Green is no pattern. The horizontal axis is Reynolds' number while the vertical axes are the respective titles. a) Shear Rate Step 11.5 b) Shear Rate Step 12.5, c) Gradient Shear Rate Step 11.5, d) Gradient Shear Rate Step 12.5, e) Non Linearity Term Step 11.5, f) Non Linearity Term Step 12.5

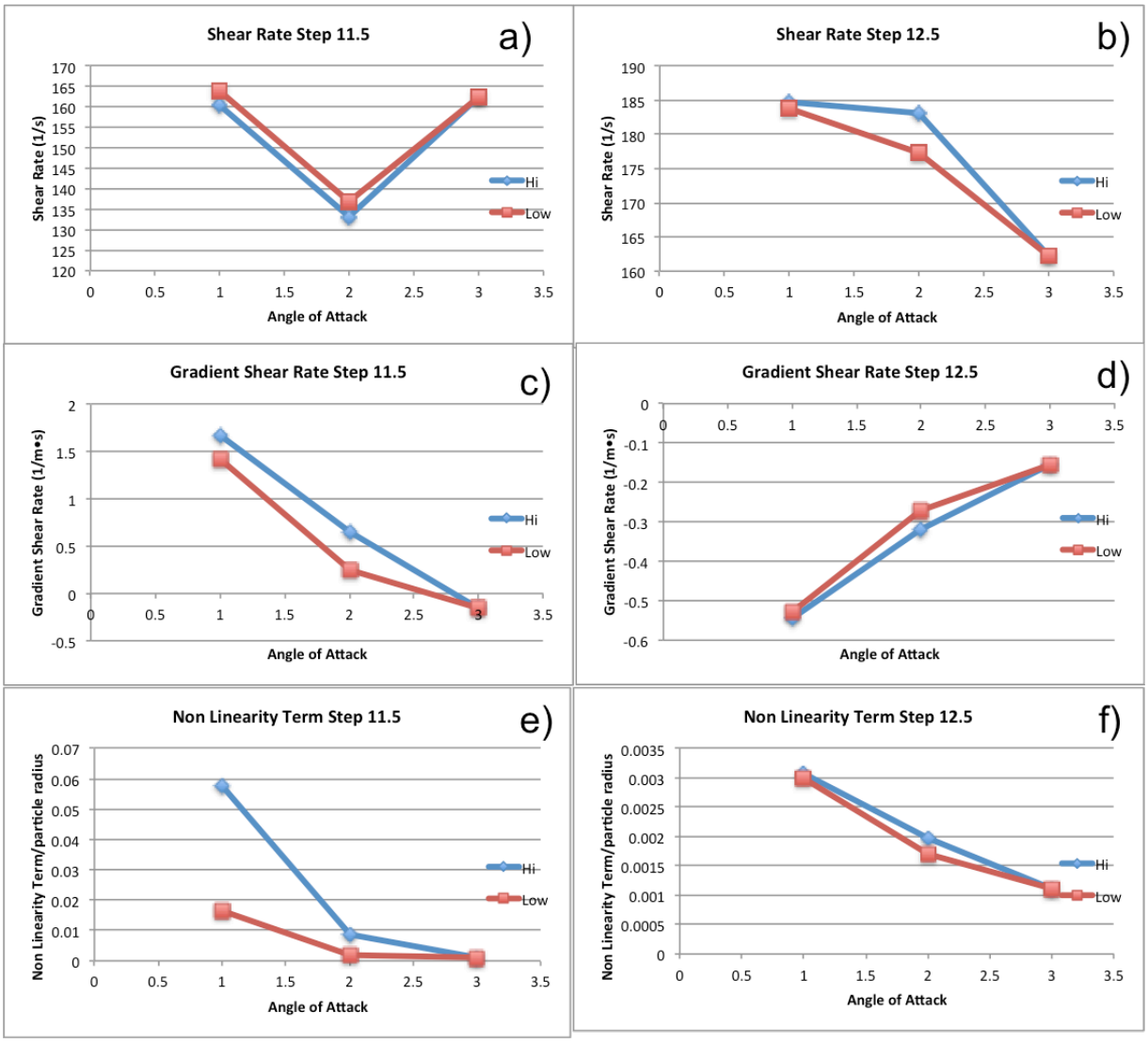


Figure 56: A summary of the comparison between high and low patterns as a function of angle of attack. a-b) shows the shear rate, c-d) shows the gradient of the shear rate, and e-f) show the non-linearity term.

In addition to angle of attack, the pattern height and the Re were varied both during the experiments and the simulations. A full set of simulation results summary can be seen in the appendix. Figure 57 compares the experimental results from [4] with the simulation results from our experiment. Our magnitude of our calculated strain rate transverse gradient over the Merlon appears to correlate well with the observed trends in for lower fouling in regards to the angle of attack as well as the cross flow velocity (Re). Meanwhile the nonlinearity parameter over the crenel describes the effect from pattern height and angle of attack on fouling reduction.

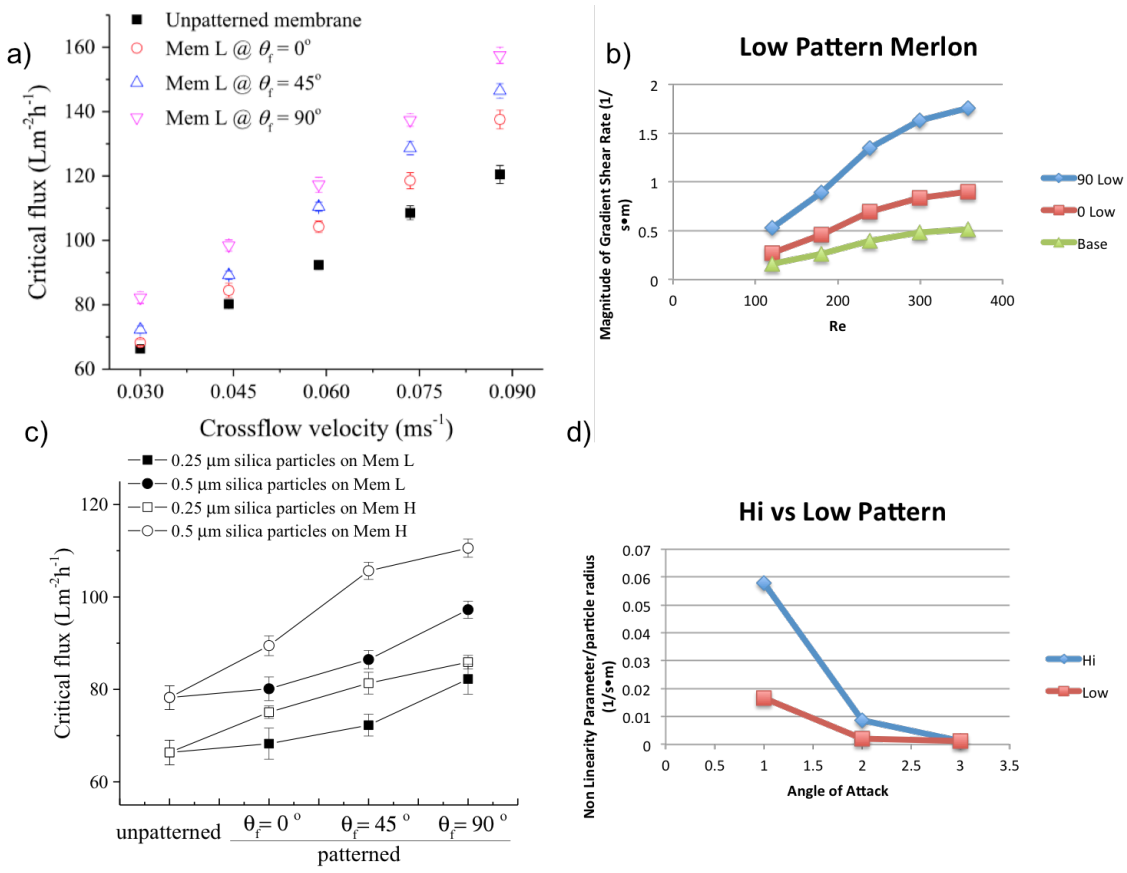


Figure 57) Side by side comparison of critical flux with the gradient of the shear rate and the non-linearity parameter as a function of the same variables. d) 1 is 90 degrees, 2 is 0 degrees, 3 is unpatterned. (56a and 56c are from [4]).

Chapter 4
Milk Experiments

Milk

There were two types of milk used: skim milk and dry milk. Dry milk was Kroger brand dry milk purchased at Kings Super. Its official product title was "Instant Non-Fat Dry Milk: fortified with vitamins A & D". It was used in a few experiments but there was a fundamental problem. It would heavily fall out of solution during normal running procedures. The dry milk's ingredient list on the box is the following: nonfat dry milk, Vitamin A, palmitate, Vitamin D3. The dry milk grams ratio is 125mg sodium to 12g sugars to 8g proteins to 23g of weight. The percentage of daily intake (DI) of vitamins and minerals is as follows: 30% DI value of calcium, 2%DI vitamin C, 10% DI vitamin A and 25% DI Vitamin D. The liquid milk was purchased at Safeway and was Safeway brand fat free milk. Its ingredient list is: nonfat milk, Vitamin A, palmitate, Vitamin D3. The liquid milk ratio is 135mg sodium, 13g carbohydrates of which 12g are sugars, and 9g proteins. That is for 240mL of liquid milk. The vitamins and mineral ratio are vitamin A 10%, calcium 30%, vitamin C 4%, and vitamin D 25%.

Liquid milk was used in almost all experiments. In addition, 0.2g of sodium azide was added as an antimicrobial component for each L of overall solution. We used 4L of milk for our experiment since it provided the biggest basin for minimal changes in composition as the water and sugars were removed. Milk experiments were initially done with cold milk but the fouling rate was extraordinarily high. Since industry performs their filtration at 48ºC and the bulk of literature also filters at this rate due to the temperature's effect on the casein micelles we also decided to perform our experiments at this temperature. In addition the same literature recommended that after the milk reach 48ºC to wait for one hour keeping it at that temperature

for the solution to reach a stable equilibrium. We follow this protocol in all experiments with hot milk.

Membranes

There were two primary types of membranes used. The first is the Koch HFK328 membrane with a molecular weight cut off (MWCO) of 5kg/mol (kDal). The second one is a PW membrane with a MWCO of 20kg/mol. The HFK328 is the membrane used in the milk industry. Relevant to our protocols are its pH tolerance for cleaning, pressure range and temperature range. Of those, only the pH tolerance is close to our cleaning protocols with a tolerance for cleaning of 1.8 to 11pH. The remainder of details for the HFK328 membrane can be found here.[12]

The patterned and pristine membranes are kept in the original storage bags that they were purchased in. This solution is designed to keep them hydrated but I do not know what is in the solution. For preparation of membranes for all experiments the membranes are first cut into the appropriate shape and then soaked overnight in DI water to hydrate the membranes. In the morning the membranes are switched to a solution that is 20% Isopropanol and 80% DI water (18MΩ) and cooked in an oven for 30 minutes at 50ºC. To get the membranes were acquired as rolls and so have a tendency even after the pretreatment to curl up. To rectify this long enough to securely close the flow cell without leaking an ethanol spray bottle was used to flatten the membrane (with ethanol, not the bottle itself).

Experimental Set Up

Our experimental setup was designed to run in triplicate. From the CFD section you have an idea of what the flow cell looked like and the heavily mixed flow that was inside as a result of

[12] http://www.kochmembrane.com/PDFs/Data-Sheets/Spiral/UF/hfk-328-food-dairy-datasheet.aspx

the geometry. It consists of a parallel set of plates with three compartments for membranes operating in a traditional filtration system with ports for flow feed, retentate and permeate. The flow feed begins from one pump and splits into three new channels using a cross pipe fixture. From the retentate exits come half of a differential pressure transducer and then a backpressure regulator, which makes the pressure above the membrane with the other side held at atmospheric pressure. On the permeate side comes the other half of the differential pressure transducer. Then the retentate leads to 3 Sartorius balances. Each permeate has its own balance. Next to the pump is the hot water bath that keeps the feed solution warm.

The Sartorius balances are digital balances with a ±0.1g tolerance. A program called Winwedge, which records the weight for each scale in a text file at a repeated interval, controls the scales. A scale that we use for weighing the ingredients and for the milk when requiring a high time resolution has a tolerance of ±0.0001g. Our hot water bath is a Cole Parmer 12107-70. It is a several gallon container with a digital temperature control. We use Ping-Pong balls to help insulate the bath when the top is open in order to put the flasks in for heating. A wide variety of pumps were used throughout this experiment. In the end a pump capable of providing 0.1L/s to each of the lines was used while still allowing the pressure to get up to 80psi under those cross flow conditions. For other people attempting to duplicate the work, I would recommend a different pump. One that could reach cross flow velocities of 3L/s. For our pump we attempted to use a pulsation dampener but did not use one in our experiments. Labview was used to record the data from the differential pressure transducers.

Types of Filtration Experiments

There are two types of filtration experiments: constant TMP and constant flux. Constant flux involves a constant large pressure up top above the membrane and the pressure varies below

the membrane to create a TMP that provides the correct amount of flux. The fixed amount of flux is controlled by a pump on the permeate side that pumps permeate out at a fixed rate controlled by the scientist. Typically that pressure build up is due to a back up of flux. Constant pressure involves keeping a fixed pressure above the membrane while exposing the permeate feed to containers that are exposed to atmospheric pressure. This keeps the pressure across the membrane constant while allowing the flux to vary. Our lab uses both types of experiments. However, due to the small size of our membranes (our flow cell had a membrane area of 9.6cm^2), constant flux experiments are more difficult to do. As such we chose for most of our experiments to be constant pressure.

The next break down in types of experiment is a stepping method or a constant value method. For example, with constant flux you can use a constant flux value and record and run for a given period of time until the pressure stabilizes. Then you raise the flux rate and wait for the pressure to stabilize. The flux is stepped up and down. The problem with this type of experiment is that milk is a continuously fouling substance at all conditions. This means a true equilibrium is never achieved and that fouling is a function of time. In a simpler solution, fouling is a function of TMP, cross flow and flux. That is why our constant run conditions are used instead. In this type of experiment you keep the pressure constant and watch as the flux varies as a function of time. This provides a clearer image on the time-based nature of fouling along with ideas for pseudo equilibrium states.

Experimental procedures

I will run through the broad steps of the experiment before tackling the individual steps in greater detail. The experiment begins with the membrane preparation. Without proper hydration of the membranes than there will be little to no flux through the membrane. After the

membranes are securely in place with no leaks and with permeate abilities in the filtration system then we begin set ups for recording. First stage is the hot water compaction. Second stage is the first hot milk filtration. Third stage is cleaning. Fourth stage is DI water recovery. Fifth stage is second hot milk filtration. Sixth stage is cleaning. Seventh stage is second DI water recovery. Eighth stage is third hot milk filtration. After the 8th stage we remove one membrane for examination under SEM and for testing. Then we do a final clean stage. The two cleaned membranes are also stored for examination under SEM and for testing. Then a final cleaning of the whole system takes place to make sure no foulants remain in the flow system.

The hot water compaction begins by heating water up to 48ºC in a hot water bath set to 50ºC. Typically I will heat them up together and so it takes around 1 hour to do so. After the water is at the correct temperature, the scales are turned on and the Windwedge program is pulled up for each scale along with a text file to record the data. At this point the Labview program that records the differential transducer data is turned on and set to begin recording. Then the scales are set to start recording. Next the pump is turned on and set in our case to 60Hz, which provided a stable flow to the system at our desired cross flow rate. Lastly fine tune adjustments made to the pressure through the use of the backpressure regulators are completed. The desired TMP pressure will be equal to the highest TMP that is planned to be used in the experiments. For us, the desired TMP was 40psi. The hot water compaction stage ran for two hours for us to level out, but should be run until there is no change in pure water permeance with time. In the compaction stage the scales record the weight of permeate every minute and the Labview records the pressure every 5 seconds.

The hot milk filtrations are the most time consuming stages of the experimental procedure. Under our systems the milk takes two hours to prep and the filtration lasts four hours

and then cleaning must be done before allowing the milk to settle. So before starting the milk has to be heated to 48ºC and held at that temperature for one hour before filtration can begin. During this time, you should weigh and label six small vials to be used in the permeate collection during the first 20 minutes of the milk filtration. It is also helpful during this time to prepare the base for cleaning or if you are doing other cleaning protocols the solutions needed. Unlike water permeation, due to the small size of our membranes our time scale here will be different. Labview will still select pressure data every five seconds, but the weight collection time is different. Due to our membranes our collection time set for the digital scales was 5 to 10 minutes depending upon the permeate rate of the first 20 minutes of collection that were recorded by hand. During the first 20 minutes a different scale that was significantly more sensitive was used (the ±0.0001g). In order to measure the three different permeates we used two sets of three small vials. Before the experiment begins each tube is in a vial marked 1, 2, or 3 in the front row. When the pump is turned on a timer begins, set for 1 minute. After a minute the tubing is switched from the front row to the back row (approximately 2-3 seconds time). Then the sensitive scale issued to record the weight of those vials and then place them back in the front row. At the second minute since the experiment began the tubes are switched back to the first row of vials and the second row is weighed. This process repeats until twenty minutes have passed. This hand monitoring has a two-fold purpose. The first is as mentioned, in order to figure out the time scale needed for the less sensitive scale to be large enough. The second is because the most fouling changes occur in the first 5 to 10 minutes due to a different fouling mechanism and we wanted to be able to resolve that feature. After the first 20 minutes are done it can be switched the digital system and left alone.

The cleaning phase used to be longer and more tedious involving and acid and base cleaning step; however, I already covered in the background why that is not done and why the new cleaning procedures only involved water flushes and hot base cleaning. The first step of cleaning is a purging the milk from the pump chamber (only for pumps that use chambers and not tubing) and also from the flow cell. This takes roughly 8L of DI water for our pump system combination but will depend upon your system and pump specifics. Next, 4L of DI water (18MΩ) is heated to 48ºC and run through the system for 20 minutes. The breakdown for that time consists of 5 minute run, 2 minute sit, 5 minute run, 4 minute sit, 1 minute run. Then the base which, is composed of NaOH in a pH between 10 and 11 (I aim for 10.5), has been heated to 48ºC is then run through the system following the same run instructions as for water. After this you must be careful, because you must collect the entire base to neutralize. When purging the base, the retentate must be collected in a large flask for neutralization. For our system this required a 4L container to catch the base and 4L water reservoir. After that, I run the water on recycle for 20 minutes following the same protocols. I then neutralize the base feed, and the purge based. Once the water is done, I test the pH level. If neutral then it is fine. If it is basic I neutralize it and do another water flush and recycle system. For records, nitric acid was used for neutralizing the base.

The last type of phase is a collection of the recovered pure water permeance again. Depending upon goals, you can run a pure water permeance before cleaning, but we avoid doing that because of the risk of further compacting the gel layer, but also because there are the side effects of cleaning it while running which will lead to unstable results during filtration. As such, we run the permeate after the cleaning. This can be done directly after the cleaning or in the

morning before the next hot milk filtration while waiting for compaction to occur again. This segment occurs just as in the initial compaction.

Preliminary Milk Experiments

I am going to start off with HFK328 unpatterned membrane experiments that we performed. These were performed with room temperature liquid milk. This one was done at constant flux system of experiments (Figure 58). During this experiment our pump was a tube based pump whose cross flow rate was around 0.01L/s and a maximum psi of 40. What this means is that in Figure 58 that both 0.2mL/min and 0.1mL/min were too high compared to the actual production of permeate at a TMP of 40 psi. This is why we stopped doing constant flux based experiments although we eventually did get a new pump head and new tubing to do even smaller flow rates. We, however, decided to finish our current line of inquiry at the time before pursuing our old inquiry.

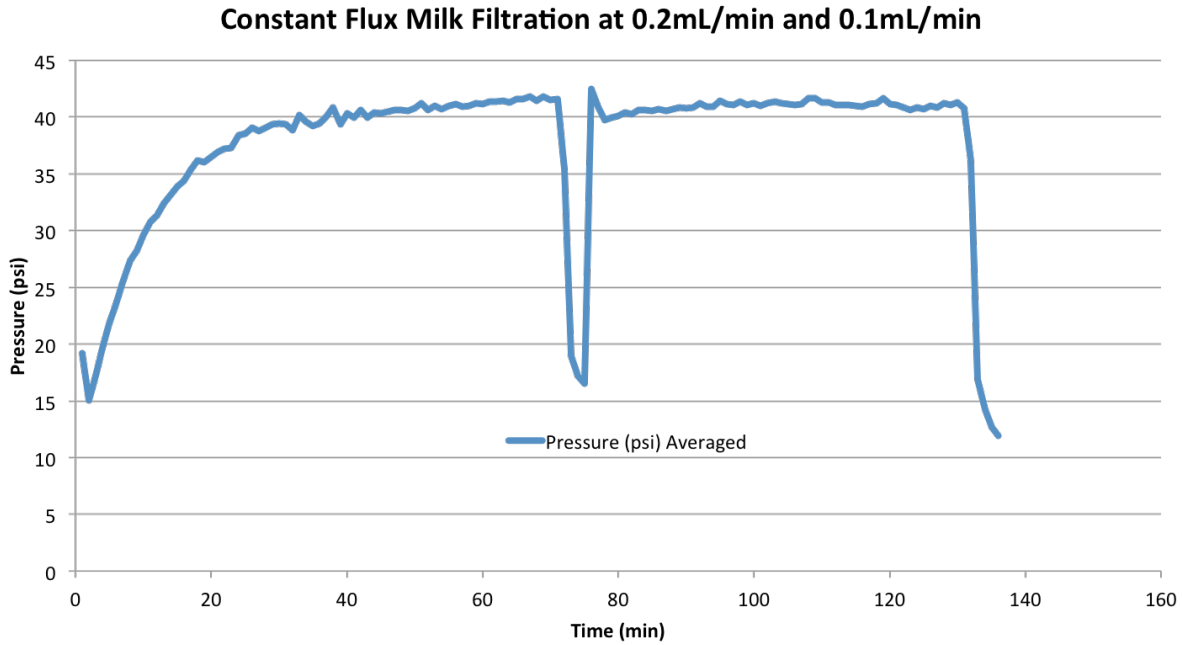


Figure 58: Constant flux experiment done with pure Safeway milk at store concentration at room temperature using our first pump that had a maximum pressure of 40psi. Done on HFK328 membranes.

The next set of experiments was also done on HFK with cold milk, but used a newer pump that performed only slightly better. Here I measured a different set of parameters using a constant pressure system. It was always disheartening from the beginning that our membranes never did operate within their established parameters (Figure 59). But we endured and I gathered some more data. It was figures 60, 61 and most importantly 62 that sealed the deal on using hot water and hot milk for flux experiments. Figure 60's flux vs. TMP step tells us some very interesting behavior. Because an ideal fluid would not have a decrease in flux with an increase in pressure like we observed. Part of this may be due to continuous fouling, but most likely it was due to the cold temperature and the extreme fouling that would take place. The gel formation layer would get near the size of the whole domain. Figure 61 shows the slight deviation in permeance linearity showing a critical fouling point was reached, although permeance is a constant as a function of pressure. Figure 62 shows that the water permeance is much more effective when used warm and contributed greatly to our decision to do warm water and milk experiments.

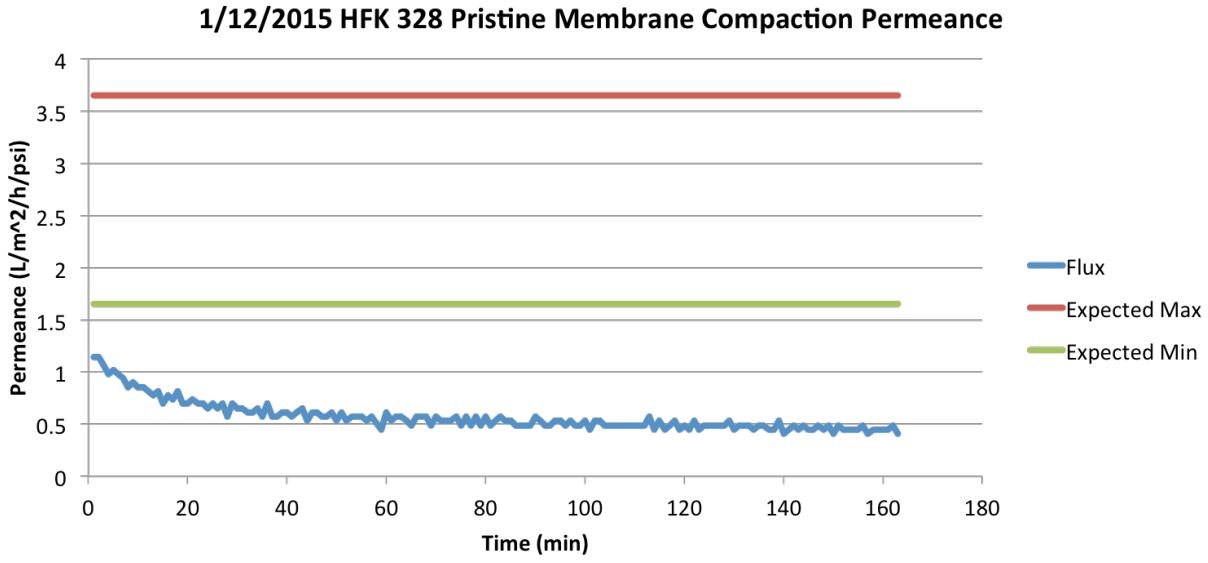


Figure 59: Pure water permeance done at room temperature and 40 psi. The expected min and max are the statistics from the membrane for using room temp (21 ºC) water and well independent of pressure because permeance is supposed to be independent of pressure. The figure illustrates how bad the permeate really was with room temperature fluids with these membranes.

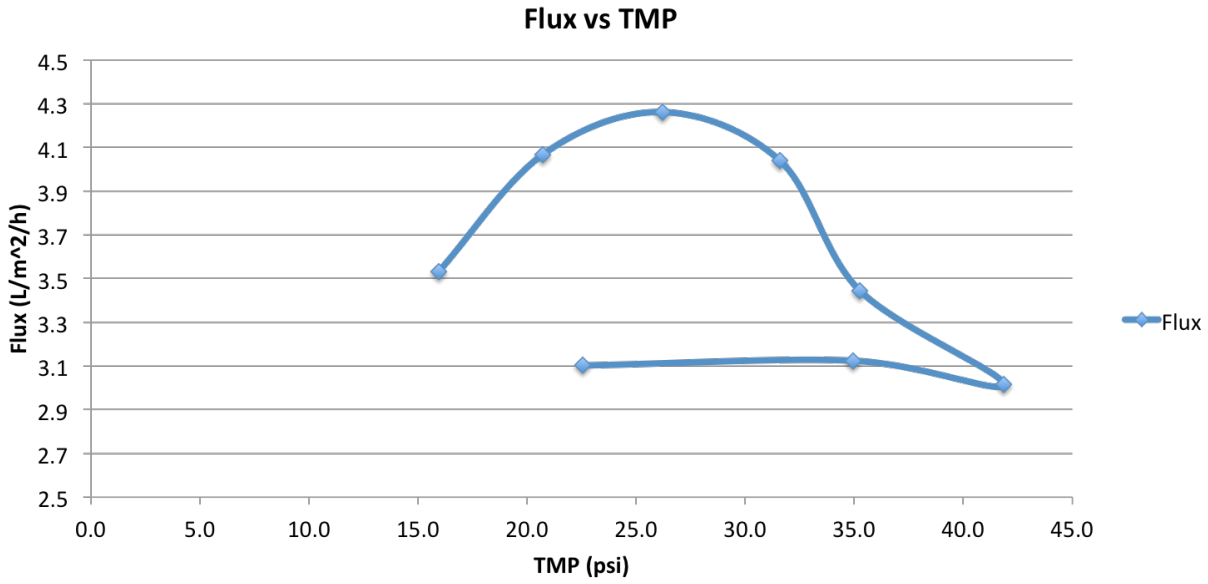


Figure 60: For this experiment I stepped up pressure and took the lowest flux after 30 minutes and used that as the point. Thus making a flux vs. TMP graph. Given the 30-minute run times it was unable to go to completion but did provide some details about complex fluids. The order of the flow was top left and then to the right.

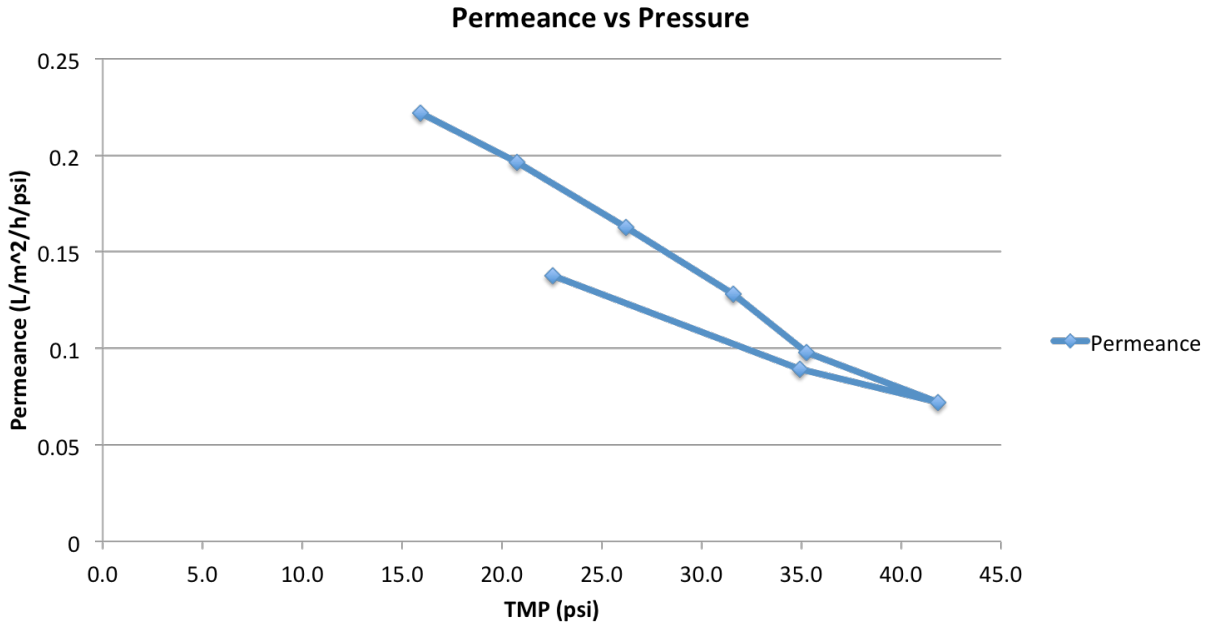


Figure 61:Permeance and Pressure from the same experiment in figure 59

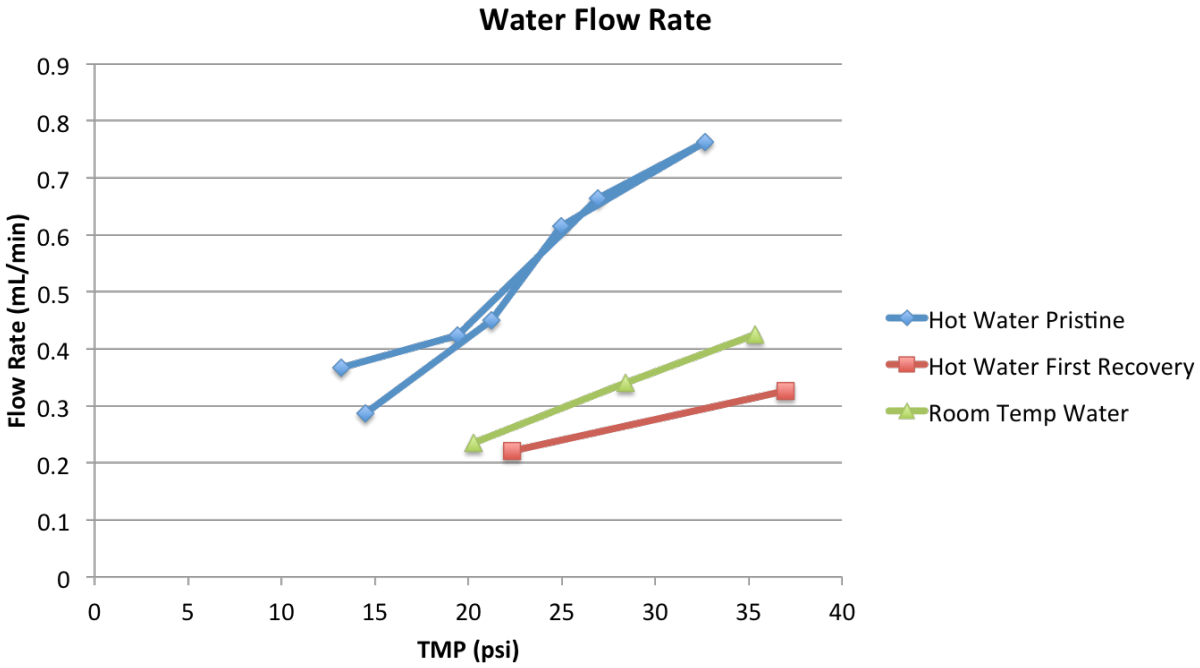


Figure 62: Using a combination of hot water and hot milk to test recovery rates and the effect of heated milk on fouling.

Moving onto PW membrane experiments we get an important set of results concerning permeance recovery. In fact it is our only complete set of data involving a patterned and non-patterned membrane. The downside is that we used acid cleaning so the results are less conclusive than they could have been given the literature I already cited. Table 5 summarizes the results from the completed experiments. Given that there is over 100% flux recovery I don't view the results as to reliable.

	HFK Pristine			HFK Patterned		
	First	Second	Third	First	Second	Third
Permeance 1	0.0346	0.0436	0.0418	0.0498	0.0427	0.0462
Permeance 2	0.0311	0.0458	0.04	0.0478	0.0386	0.0369
Premeance 3	0.0446	0.0455	0.0399	0.0455	0.0366	0.0367
Average Permeance	0.0368	0.045	0.0406	0.0477	0.0393	0.0399
% Recovered 2nd-1st	125.06			82.32		
% Recovered 3rd-2nd		90.27			101.40	
% Recovered Average	107.67			91.86		

Table 5: The values were inconclusive since recovery is increased due to the acid step in proportion to the amount that it is fouled.

What I feel is more meaningful is this dry milk filtration experiment (figure 63). This experiment ran overnight so we were not there during the dramatic drop. However, when we came in during the morning, the majority of the milk powder had fallen out of solution. We did not have a working waterproof stir bar. As such we went back to using wet milk. I was unable to determine the % of dry milk that fell out of solution. However, I find the result that we had a relatively slow decrease for several hours before it began to get worse at a significantly increasing rate. I think it would be worthwhile trying to study and determine what happened here and why would even a decrease in concentration cause the problem. My hypothesis is that while milk fell out of solution in the flask in the 48ºC bath that it also fell out onto the membrane and throughout the flow cell system. And this build up led to the sudden drop in permeance.

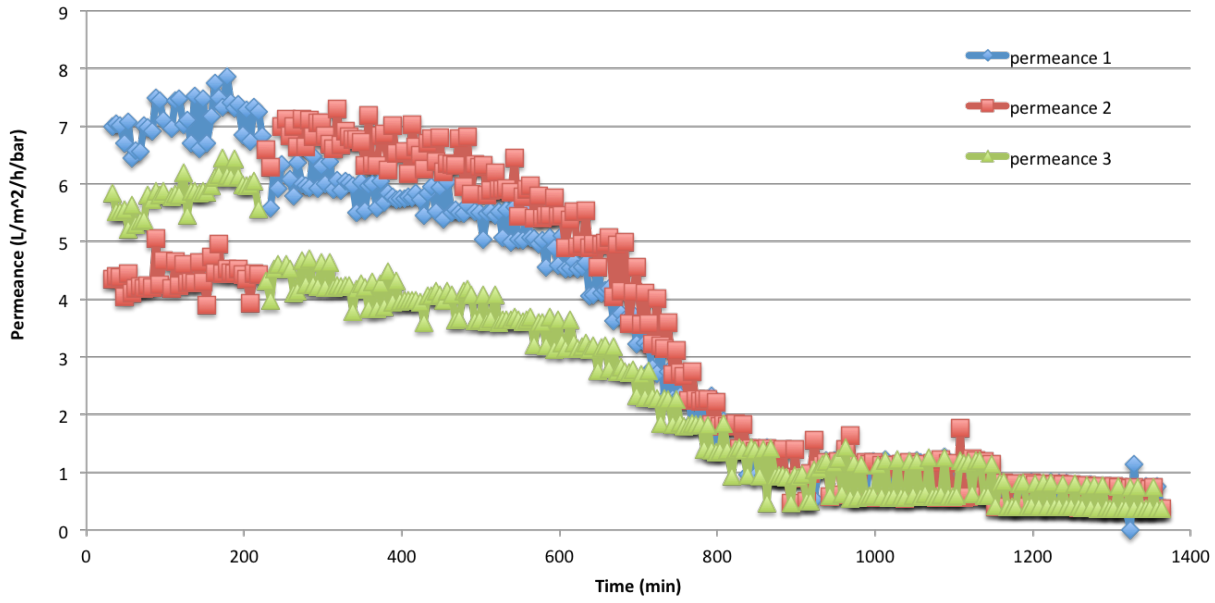


Figure 63: The dry milk filtration experiment. There was a 4% protein concentration and the milk solution was heated to 48ºC and kept there for an hour before filtration as the protocol

dictates. The color change was due to a recording error where the recording file missed a point due to the mouse being elsewhere there on the text. As a result the colors switched.

The experiment from figure 64 is a duplicate experiment of the experiment that gave us figure 62. The only difference is that the experiment for figure 64 used wet milk. There were two main differences; the first is that the standard deviation for the wet milk is significantly lower than the deviation for dry milk. This deviation might have to do with the back flow regulators since eyes adjust those. But it could also just be standard variation. Next is the duration of run. Further work would need to do an overnight experiment to see if wet milk had the same drop.

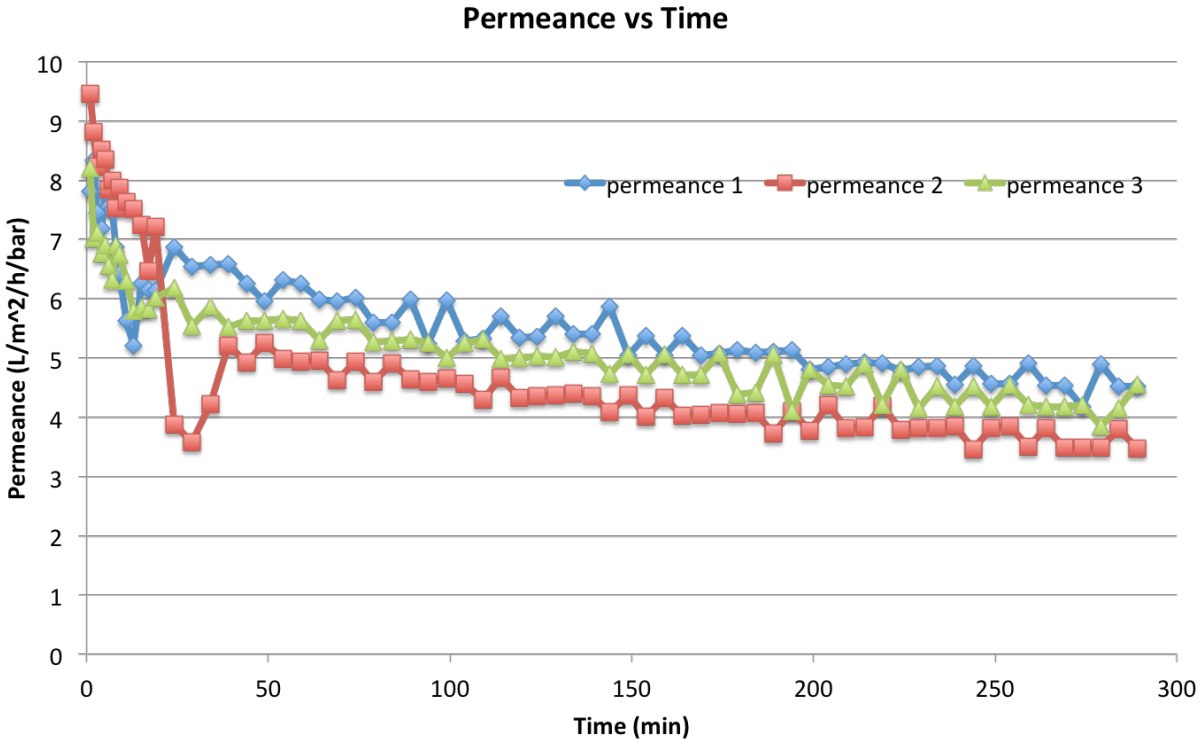


Figure 64: This is a diluted down concentration of wet milk to match the 4% protein concentration from the dry milk experiment. It was run under the same conditions.

Chapter 5
Conclusion

The CFD experiments provide insight into the fluid mechanical workings of the fouling reduction due to the implementation of submicron scale patterns on membrane surfaces. Findings indicate that a combination of the fluid flow over the merlon and crenel contributes to the fouling reduction and that different fluid dynamic effects are behind each domain. Over the crenel Ingber's modified shear induced particle migration model matches the experimental and simulation results [40]. Over the merlon Eckstein's work is the closest but there is a fundamental lack of models where the magnitude of the gradient of the shear rate plays a roll that is not neglected [37]. Future work would include the creation of such a model to better describe the fluid processes going on. Meanwhile, the bulk of literature focuses on the shear rate's role in the fouling process but our simulation models return results indicating that it doesn't play a significant role in fouling reductions with patterns which is novel and significant in its own right. Furthermore, different forces and models control the different experimental phenomena (angle of attack, pattern height, cross flow velocity (Re), and permeate rate). Permeate rate was shown that while linear in effect, the change it induces is so small that it is negligible in effect without particles being introduced into the system for the purpose of balancing the back mass diffusion coefficient. Cross flow velocity was shown to have no correlation with respect to the non linearity term but follows the magnitude of the shear gradient fairly well. Pattern height fails to be described by shear rate or its gradient alone, but matches Ingber's non-linearity factor fairly well. Angle of attack is described both by the non-linearity factor and the gradient of the shear rate. These results provide a pathway for optimization of patterned membrane design, as there is now shown to be a derived simulation result that can, perhaps, be modeled to search for a maximum fouling reduction. Further experimental and simulation work can focus on fine-tuning

the relationship and identifying if there are any other fluid dynamic quantities that will affect fouling mitigation. Additionally, this work also stimulates the development of more ambitious hybrid modeling of the multi-scale, mass transfer in the near-surface domain of the membrane.

Unfortunately our conclusions regarding the milk are far from conclusive. To many experiments under different conditions made it difficult to draw anything from the results. In addition our one full set of experiments while complete and nothing went wrong, its results were brought into question because of the acid cleaning step, which can explain the recovery reaching above 100%. What we did validate is that there is a time dependent fouling feature and that something we might be able to compare with future experiments is the onset of that significant decrease in flux.

That concludes my conclusion on my last three and a half years of research. I feel that so much of my time was dedicated to trial and error, especially regarding the CFD work because there was no parallel basis from which to drawn nor guidelines from which to work. Furthermore, exploring new territory where none had gone before proved to be a challenge for quantifying results but at the same time, the very things that made it inefficient were also the largest causes for growing and why it was real research and not just another lab experiment. In regards to the milk, because of our limited supply of patterned membranes we were overly cautious in our experimental use of them and as such only had the one completed sets of results as we kept changing the laboratory procedure as we kept improving the methods to try to get more accurate results and more conclusive evidence. I believe our current experimental model is sufficient to the task and only regret the lack of time to finish the experiments of the patterned HFK 328 membrane.

Works Cited

1. Bodzek, M.K., Krystyna, *Comparison of various membrane types and module configurations in the treatment of natural water by means of low-pressure membrane methods.* Separation and Purification Technology, 1998. **14**(1-3): p. 69-78.

2. Brans, G., et al., *Membrane fractionation of milk: state of the art and challenges.* Journal of Membrane Science, 2004. **243**(1-2): p. 263-272.

3. Maruf, S.H., et al., *Influence of sub-micron surface patterns on the deposition of model proteins during active filtration.* Journal of Membrane Science, 2013. **444**: p. 420-428.

4. Maruf, S.H., et al., *Critical flux of surface-patterned ultrafiltration membranes during cross-flow filtration of colloidal particles.* Journal of Membrane Science, 2014. **471**: p. 65-71.

5. Bacchin, P., P. Aimar, and R. Field, *Critical and sustainable fluxes: Theory, experiments and applications.* Journal of Membrane Science, 2006. **281**(1-2): p. 42-69.

6. Cohen, R.D.P., R. F., *Colloidal Fouling of Reverse Osmosis Membranes.* Journal of Colloid and Interface Science, 1985. **114**(1): p. 194-207.

7. Belfort, G.D., Robert; Zydney, Andrew, *The behavior of suspensions and macromolecular solutions in crossflow microfiltration.* 1994.

8. Behrens, T., *OpenFoam's basic solvers for linear systems of equations.* February 18, 2009.

9. Jimenezlopez, A., et al., *Role of milk constituents on critical conditions and deposit structure in skimmilk microfiltration (0.1μm).* Separation and Purification Technology, 2008. **61**(1): p. 33-43.

10. Wemsy Diagne, N., M. Rabiller-Baudry, and L. Paugam, *On the actual cleanability of polyethersulfone membrane fouled by proteins at critical or limiting flux.* Journal of Membrane Science, 2013. **425-426**: p. 40-47.

11. Berg, T.H.A., et al., *Investigation of Consecutive Fouling and Cleaning Cycles of Ultrafiltration Membranes Used for Whey Processing.* International Journal of Food Engineering, 2014. **10**(3): p. 367-381.

12. Youravong, W., M.J. Lewis, and A.S. Grandison, *Critical Flux in Ultrafiltration of Skimmed Milk.* Food and Bioproducts Processing, 2003. **81**(4): p. 303-308.

13. James, B.J., Y. Jing, and X. Dong Chen, *Membrane fouling during filtration of milk––a microstructural study.* Journal of Food Engineering, 2003. **60**(4): p. 431-437.

14. Metsämuuronen, S. and M. Nyström, *Critical flux in cross-flow ultrafiltration of protein solutions.* Desalination, 2005. **175**(1): p. 37-47.

15. Rabiller-Baudry, M., et al., *Limiting flux in skimmed milk ultrafiltration: impact of electrostatic repulsion due to casein micelles.* Desalination, 2005. **175**(1): p. 49-59.

16. Paugam, L., et al., *Cleaning of skim milk PES ultrafiltration membrane: On the real effect of nitric acid step.* Journal of Membrane Science, 2013. **428**: p. 275-280.

17. Maruf, S.H., et al., *Use of nanoimprinted surface patterns to mitigate colloidal deposition on ultrafiltration membranes.* Journal of Membrane Science, 2013. **428**: p. 598-607.

18. Pan, F.A., Andreas, *Steady Flow in Rectangular Cavities.* Journal of Fluid Mechanics, 1967. **28**(4): p. 643-655.

19. Bessonov, A., et al., *Design of Patterned Surfaces with Selective Wetting Using Nanoimprint Lithography.* Macromolecular Chemistry and Physics, 2010. **211**(24): p. 2636-2641.

20. Won, Y.-J., et al., *Factors affecting pattern fidelity and performance of a patterned membrane.* Journal of Membrane Science, 2014. **462**: p. 1-8.

21. Maruf, S.H., et al., *Influence of nanoimprint lithography on membrane structure and performance.* Polymer, 2015. **69**: p. 129-137.

22. Maruf, S.H., et al., *Fabrication and characterization of a surface-patterned thin film composite membrane.* Journal of Membrane Science, 2014. **452**: p. 11-19.

23. Howe, K.C., Mark, *Fouling of Microfiltration and Ultrafiltration Membranes by Natural Waters.* Environmental Science & Technology, 2002. **36**(16): p. 3571-3576.

24. Ko, M.P., John, *Determination of Osmotic Pressure and Fouling Resistances and their Effects on Performance of Ultrafiltration Membranes.* Journal of Membrane Science, 1992. **74**: p. 141-157.

25. Le-Clech, P., V. Chen, and T.A.G. Fane, *Fouling in membrane bioreactors used in wastewater treatment.* Journal of Membrane Science, 2006. **284**(1-2): p. 17-53.

26. Rana, D.M., T., *Surface Modifications for antifouling membranes.* Chemical Reviews, 2010. **110**(4): p. 2448-2471.

27. Choo, K.-H.L., Chung-Hak, *Membrane Fouling Mechanisms in the Membrane Coupled Anaerobic Bioreactor.* 1996. **30**.

28. Zhu, X.E., Menachem, *Colloidal Fouling of Reverse Osmosis Membranes: Measurements and Fouling Mechanisms.* Environmental Science & Technology, 1997. **31**: p. 3654-3662.

29. van der Ber, G.B.S., C A, *Diffusional phenomena in membrane separation processes.* Journal of Membrane Science, 1992. **73**: p. 103-118.

30. Luo, J., et al., *Threshold flux for shear-enhanced nanofiltration: Experimental observation in dairy wastewater treatment.* Journal of Membrane Science, 2012. **409-410**: p. 276-284.

31. Choi, D.-C., et al., *Three-dimensional hydraulic modeling of particle deposition on the patterned isopore membrane in crossflow microfiltration.* Journal of Membrane Science, 2015. **492**: p. 156-163.

32. Jung, S.Y., et al., *Particle deposition on the patterned membrane surface: Simulation and experiments.* Desalination, 2015. **370**: p. 17-24.

33. Lee, Y.K., et al., *Flow analysis and fouling on the patterned membrane surface.* Journal of Membrane Science, 2013. **427**: p. 320-325.

34. Jamshidi Gohari, R., et al., *Effect of surface pattern formation on membrane fouling and its control in phase inversion process.* Journal of Membrane Science, 2013. **446**: p. 326-331.

35. T. Reddy, S., *Surface micropattern resists bacterial contamination transferred by healthcare practitioners.* Journal of Microbiology & Experimentation, 2014. **1**(5).

36. Won, Y.J., et al., *Preparation and application of patterned membranes for wastewater treatment.* Environ Sci Technol, 2012. **46**(20): p. 11021-7.

37. Eckstein, E.B., Douglas; Shapiro,Ascher *Self-diffusion of particles in shear flow of a suspension.* Journal of Fluid Mechanics, 1977. **79**: p. 191-208.

38. B. P. Ho, L.G.L., *Inertial migration of rigid spheres in two-dimensional unidirectional flow.* Journal of Fluid Mechanics, 1974. **65**: p. 365-400.

39. *Membrane Handbook*. Vol. 1. 2001.

40. Ingber, M.S., et al., *An improved constitutive model for concentrated suspensions accounting for shear-induced particle migration rate dependence on particle radius.* International Journal of Multiphase Flow, 2009. **35**(3): p. 270-276.

41. Ingber, M.S., et al., *The analysis of self-diffusion and migration of rough spheres in nonlinear shear flow using a traction-corrected boundary element method.* Journal of Fluid Mechanics, 2008. **598**.

42. Tiwari, P., S.P. Antal, and M.Z. Podowski, *Modeling shear-induced diffusion force in particulate flows.* Computers & Fluids, 2009. **38**(4): p. 727-737.

43. Rusconi, R. and H.A. Stone, *Shear-Induced Diffusion of Platelike Particles in Microchannels.* Physical Review Letters, 2008. **101**(25).

44. Cosden, I.A. and J.R. Lukes, *A hybrid atomistic–continuum model for fluid flow using LAMMPS and OpenFOAM.* Computer Physics Communications, 2013. **184**(8): p. 1958-1965.

A: Flow Cell Write Up and Appendix

Abstract

The flow cell is designed to study the effects of energy dissipation from nano-imprinted (NI) membranes through pressure drop. The second iteration of the design will incorporate constant permeate and constant trans membrane pressure (TMP) experiment capabilities to study the fouling and cleaning in greater detail rather than studying only fluid flow. The importance of the first iteration is to validate the simulation model software that will be used to model the multi scale fluid flow over the NI membranes. The key feature is the ability to place nano-imprinted tiles in with different patterns and orientations to gather experimental data on the effect of the patterned roughness and its geometry on both the nano scale and the bulk flow. The principal variables controlled in the flow cell experiments are: channel height, inlet velocity/mass flow, and imprinted tile patterns. Concerns that influenced the development of the flow cell are: cost, pressure, pressure drop, sensitivity and measuring pressure drop, flow development, lab space constrains, and reasonability to simulate domain.

Flow Cell Design Parameters

Several constraints are relatively simple restrictions on design. The most challenging and complex restraints are computational domain size, sensitivity and measuring pressure drop, and flow development. Of these, flow development proves to be less consequential and undoable due to some of the simple constraints. The simplest constraint, lab space, and limits our flow design to not much more than a foot long.

To meet the first design criteria the flow cell needs to demonstrate the accuracy of the simulations. To do so it must be able to distinguish the difference in pressure drop between a patterned and unpatented surface. The longer the flow cell the larger this difference will be. However, the longer the domain the more computer resources are required. Computational

resources are limited. As such we want to minimize the distance needed using the best pressure sensors we can.

The moody chart connects pressure drop (through a friction factor), Reynolds number, and relative roughness of pipe. The relative roughness is based on random roughness and not periodic roughness. As such we specifically don't know how periodic will change the results one will presume that we need sensitivity a few orders smaller than the pressure drop one would achieve with normal roughness. The smallest pressure drop will occur at the lower end of the Reynolds number and at the smaller channel height. The transition region between laminar and turbulent region is of interest, but will not be examined in the initial work. In particular, whether the transition region between laminar and turbulent shifts going from random roughness to periodic roughness is of interest. Before starting, a few terms need to be defined: hydraulic diameter, Reynolds number, friction factor and relative pipe roughness. Hydraulic diameter is the ratio of the area over the perimeter of an object. For a circle it comes out simply to be the inner diameter of the pipe. In slit flow it can be simplified to $D_H = \frac{4 \times Area}{Perimeter} = \frac{2hw}{h+w} = 2h$ where h<<w that applies to slit flow. The Reynolds number is $\mathrm{Re} = \frac{\mathrm{v}D_H}{\upsilon} = \frac{2\mathrm{vh}}{\upsilon}$ and becomes a function of height and velocity. The relative roughness factor in the moody plot is specifically for round pipes. However, since the width is much greater than the height, for the energy loss due to roughness the distance in the vertical (shorter) dimension will play the primary role. Thus, the relative roughness factor will be defined as the root mean square (RMS) over the height of the channel $R_R = \frac{\varepsilon}{h}$. Root mean square is the square root of the sum of squares of the differences in height above or below the mean height of the bottom over the total number of

items summed. So in our case with regular patterns the RMS is half the pattern height or $RMS = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + ... + x_n^2\right)} = \sqrt{\frac{1}{2}\left((55nm)^2 + (-55nm)^2\right)} = 55nm$. The height of the domain will vary from 1mm to 10 mm. Thus the relative roughness factor will vary from 5.5e-5 to 5.5e-6. Since we are in the laminar region, we don't need the roughness factor yet and it will be applicable once we move into the transition region. In the laminar region the friction factor is proportional only to the Reynolds number $f_D = \frac{64}{\mathrm{Re}}$. The friction factor ($f_D$) is defined $f_D = \frac{2D_H}{\rho v^2 L}\Delta p$. We are interested in the pressure drop per unit length so $\frac{\Delta p}{L} = f_D \rho \frac{v^2}{2D_H}$. Substituting in the terms we desire we arrive at $\frac{\Delta p}{L} = 4\frac{\rho \upsilon^2 \mathrm{Re}}{h^3}$ where ρ is density (1000kg/m^3), υ is kinematic viscosity (1e-6 m^2/s), h is the height (0.001 m to 0.01 m) and Re the Reynolds number (varied from 1 to 2500).

Table 1: Pressure Drop per Unit Length		
	Pressure Drop Per Unit Length ΔP/L (Pa/m)	
Reynold's Number	Height 1: 0.001m	Height 2: 0.01m
1	4.00E+00	4.00E-03
10	4.00E+01	4.00E-02
100	4.00E+02	4.00E-01
1000	4.00E+03	4.00E+00
2500	1.00E+04	1.00E+01

Table 2: Pressure Drop over 10.5 inches		
	Pressure Drop (Pa)	
Reynold's Number	Height 1: 0.001m	Height 2: 0.01m
1	0.90680	0.00091
10	9.06800	0.00907
100	90.68000	0.09068
1000	906.80000	0.90680
2500	2267.00000	2.26700

Right now as part of saving money, we are using a pump that the lab already posses. It's Reynolds's number range for our cell is from 800 to 2000. This gives us a pressure drop of 725 Pa at Re= 800 and a pressure drop of 1800 Pa at Re=2000 for the default 1mm height.

Unfortunately the 10mm height pressure drop is too small to measure and differentiate between patterned and pristine surfaces with conventional differential pressures apparatuses. As such the height selection will be re-evaluated after the technological limitations of differential pressure transducers is obtained. The initial choice was Honeywell sensing differential pressure (digikey part number HSCMRRN001PDAA3-ND) whose uncertainty is ±17 Pa (.25%ffs of 1psi). This part is fine for only the 1mm height and really only useful above a Re of 1000. As such additional parts need to be ascertained. There is another part that is currently not stocked and so requires a large minimum order (digikey part number DC010NDR4-ND) that has a range of 0 to 2480 Pa with an uncertainty of ±1.24 Pa (.05%ffs of 0.36 psi). This one covers the whole initial planned range but won't be able to cover the transition region. Its uncertainty is an order 10 better but minimum purchase quantity is 30, so unless it is restocked another supplier needs to be found.

Another sensitive differential pressure device is (digikey part number DC2R5BDR5-ND) which spans only 0 to 250 Pa, but has an uncertainty of ±0.125 Pa. This one would be most useful for dealing with the third proposed height of 0.003m as well as most of the second height, but could not be used at all to measure the first height.

Table 3: Pressure Drop over 10.5 inches			
	Pressure Drop (Pa)		
Reynold's Number	Height 1: 0.001m	Height 2: 0.002m	Height 3: 0.003m
1	0.90680	0.11335	0.03359
10	9.06800	1.13350	0.33585
100	90.68000	11.33500	3.35852
800	725.44000	90.68000	26.86815
1000	906.80000	113.35000	33.58519
2500	2267.00000	283.37500	83.96296

Both the second and third differential pressure transducers are three times as expensive for an individual unit and are not currently stocked requiring a minimum order of 30 (individual price for second and third are $113.70 and $137.56 respectively compared to $47.30 of the first one). But using these three pressure transducers the pressure drops across the three heights in Table 3 can be ascertained with maximum precision to differentiate between patterned and pristine membranes and tiles.

The maximum pressure measured would be well below metal tolerance levels at 15.1 psi (.4 psi above atmospheric pressure). That is until we add a pressure back regulator and are working on the membrane experiments. However, the membrane permeate experiments will not be conducted with this model of the device so it is not a primary concern. The current pressure used in trans membrane filtrations is 50 psi for top pressure.

Using a pump we have access to Reynolds's numbers of 800 to 2000 at various increments. However, this pump tends to leak graphene into the fluid and so we have two filters that will separate the flow. The first filter is a 5-micron filter and the second filter is a 0.5-micron filter. The flow rate needed for these filters is 10L/min. Another option is that the carbon based O-ring simply needs to be replaced. The company is readily available to replace the O-ring according to laboratory postdoc.

CFD Size Limitations

Initially the goal was to use a hybrid molecular dynamics (MD) and computation fluid dynamics (CFD) simulation to reduce CPU hours required to simulate the domain. The initial plan involved using hybrid code [44]. Their work showed that their hybrid system was efficient at operating at a domain size that was at the largest 12.7nm long and 88.9nm high. This took 140 CPU hours. Unfortunately that domain is smaller than one imprint. To scale our problem we shall simulate 5 full repeatable steps, and use only the time it takes for fluid to pass through them as the residence time the simulation needs to run for convergence. In Cosden's paper they ran to 21 residence times. In addition, we will constrain the MD region (which is the limiting time factor by several orders of magnitude) to 10nm above the surface. The paper used for MD a 12.7nm cube for simulation. Reducing to just surface area for ease of comparison under best case the paper's simulation is 161.29 nm^2. For a single step the required surface area is 10540 nm^2.

$$(417nm \times 10nm) + (10nm \times 10nm) + (110nm \times 10nm) + \left[(417nm \times 10nm) - 2 \times (10nm \times 10nm)\right] + (110nm \times 10nm) + (10nm \times 10nm)$$

We will be using five steps for our back of envelope calculation. For five steps the surface area is 52700 nm^2, which is 326 times larger than the domain simulated in the paper. The time step for the MD simulation is $2*10^{-16}$ s. The velocities will vary of course, but for a low Reynolds number (Re=10) the velocity is 0.01 m/s. $v = 0.01$ $l = 417nm \times 2 \times 5 = 4170nm$

$t_{residence} = \frac{l}{v} = \frac{4.17e^{-6}m}{0.01m/s} = 4.17e-4s$. This means that our residence time is 12 orders of magnitude larger than our time steps. This alone is prohibitively expensive. So let say we don't have to run as long and only run as many time steps as the paper does. In that case let us now

apply the calculations to an inch that we would have to simulate. We cannot just simulate part of a domain because it has to be coupled for the whole length in order to have an accurate idea of the pressure drop. Let us say for an inch how many CPU hours it would take. There is 2.54e7 nm in an inch and our 5 steps have a length of 4170nm. So our domain size would increase by a factor of 6090. Combined with the previous domain increase, our domain would be almost 2 million times larger than their domain (1985340). That would mean 31 CPU Millennia (instead of hours). And that is just for one inch for 100,000 time step iterations, strictly not doable.

In order to make a more accomplishable simulation a suggested approach that would have to be developed is to couple two CFD domains together. OpenFOAM needs rectangular mesh and has a restriction that no face may be shared by more than 2 cells. In order to grasp the detail of the fluid flow nm size mesh is required, but once again that places an enormous computation strain on the domain. For 1 inch by 1mm height that is 2.54e13nm^2 which is 1.37e8 times more nodes than the other simulations I have run. That corresponds to 312 CPU Millennium to run to convergence. So just a small grid size through the whole domain is also unrealistic. This leads us to the motivation for having the coupled CFD grids. This would dramatically reduce computational cost.

Now let us theorize that our domain of 1nm spacing stretches only 10nm above the topmost pattern (so for each period it is on average 60nm high) then we have about 50,000 points per period unit. There are about 30,000 period units in an inch so 1.5e9 points in the domain. This following the same scaling procedure gives us 18 CPU Years per inch, which is still a little on large side, but more doable.

Let us reduce to a 10nm layer again. So from above that is 10540 points per period and 30,000 periods in an inch. 3.16e8 points. This is getting much more realistic as I have run 9e6

points for 20 nodes and finish in a few days (17280 CPU Hours). So that simulation domain is only about 35 times larger and so would take a few months on the super computer. Now it is at a doable scale for the department of defense but still to large for us.

Now I will move onto the next possibility, 10nm spacing (the domain would be 30nm above the raised surface so on average 7 points per 10nm of length. This gives us 17.8 million node points. That is only twice now what my simulation was and so is now in our running length and capabilities. And two weeks is a reasonable run time so under this domain setting with 60 nodes instead of 20, we would be able to have a maximum flow cell length of 7 inches. So 7 inches is our goal maximum length between two measuring points.

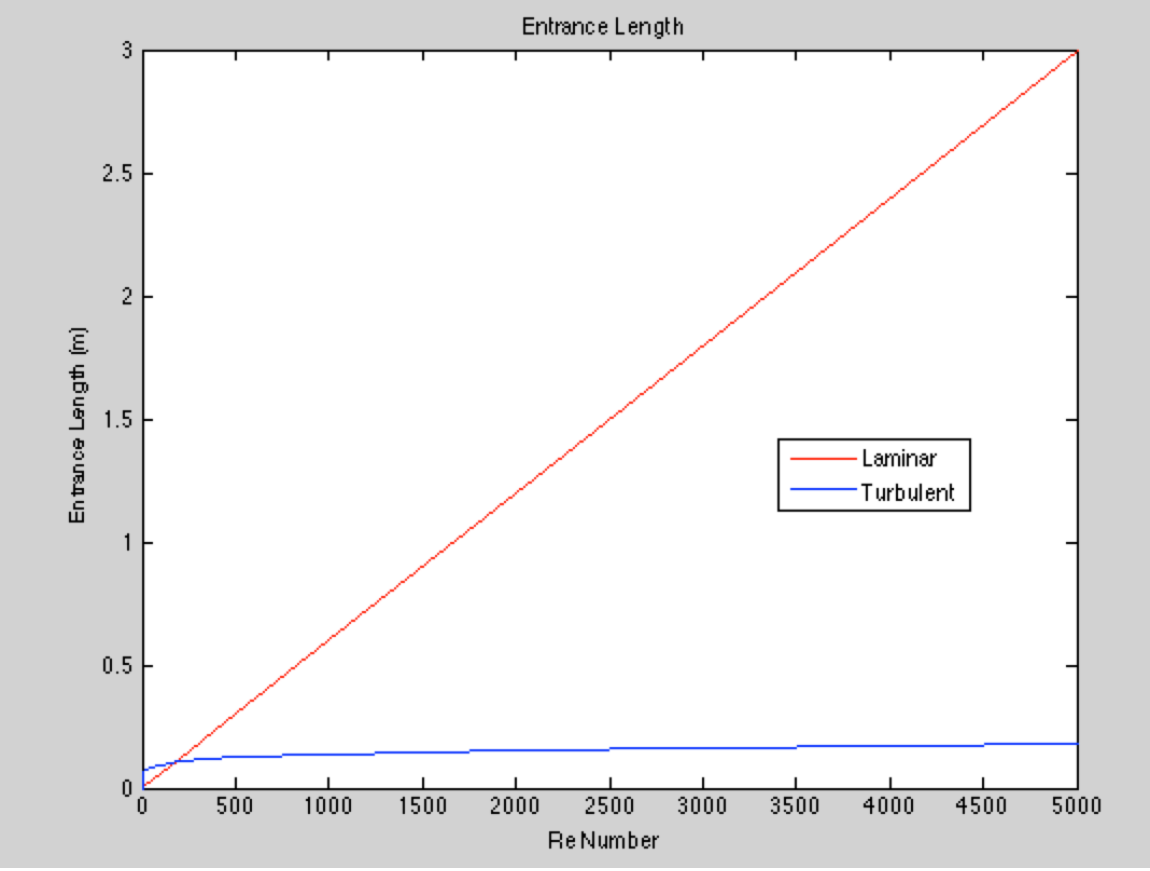
Entrance Length

Entrance length is the minimum length required in order to fully develop the flow. In early designs this was a key criteria, however, this criteria was eventually discarded for several reasons. The first is that it is impractically too long. The domain ended up having to be several feet. This is because of the extreme conditions the flow cell will run at. The key reason is since we are interested in pressure drop and will be doing several control group experiments without patterns, we will have a base to normalize and use as a reference for the pressure change, so it doesn't have to be fully developed for us to make use of it. As such I will only go briefly over references and how I figured out the fully developed length.

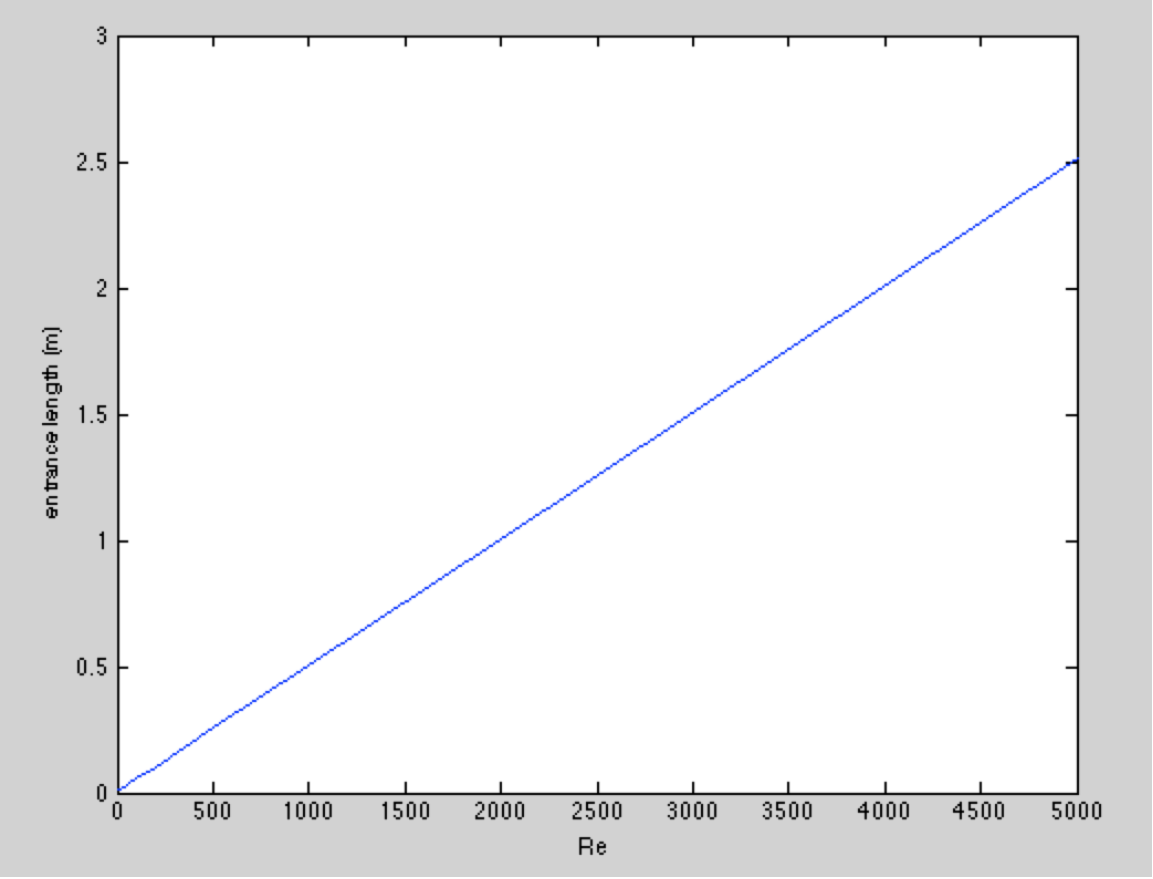
Ideally we would like to be able to have an oscillating mode that requires the entrance and exit region to be symmetric about the center. This feature will also most likely have to wait till the second iteration because our pumps are unable to do such a thing. Smooth transitions and curves help minimize entrance length development and so are used. Furthermore the entrance

length needed for a fully developed flow changes with channel height and Reynolds's number I've plotted the entrance length over our experimental domain.

Next is the entrance region to allow a fully developed flow to occur. Since we want to have an oscillating mode that requires that the entrance region be symmetric about the center around the exit and entrance domain. Smooth transitions and curves help minimize entrance length transitions so those are used. In addition, since the length needed for fully developed flow changes with channel height and Reynolds's number I've varied the flow for the pipes and plotted Reynolds's number versus entrance length for both turbulent and laminar flow. Interestingly enough, it is the laminar flow that has the longest entrance length. The entrance length is defined as $x_l = 0.06\,\mathrm{Re}\,D$ for laminar flow and $x_t = 4.4\,\mathrm{Re}^{1/6}\,D$ for turbulent flow.



As such the circular pipe approximation doesn't serve us very well. After all, we are in an extremely narrow channel and it shouldn't take that great of a distance to develop. So looking at values from experimental approximations from the book "fundamentals of fluid dynamics". For rectangular duct flow with an aspect ratio of less than 2 (which is what our system is) the entrance length is $x_l = (0.25 + 0.03\mathrm{Re})D_h$ where D_h is the hydraulic diameter that is used in calculating the Reynolds number with the hydraulic diameter. The worst-case scenario the hydraulic diameter would be 0.0167m (the case of our 2 in wide channel and 1cm high). Using this number we get a different graph.



While this has a slightly better slope, it would still require over a meter for worst-case scenario. This violates our space constraints. Thus why we figured out a way around the fully

developed flow condition. Interestingly enough, when running a simple simulation the entrance length was less than two inches. Thus two inches was chosen to be used for the entrance length.

General Design

The flow cell consists of four main parts: top plate, bottom plate, spacer plate, O-rings. The flow cell is designed to go from 1mm height spacing to 10 mm. This design height has been changed now to 1mm to 3mm height. If it were possible we would do 1mm, 1.5 mm, and 2mm heights but the O-rings wouldn't allow for that. In addition the flow cell needs to be able to handle 15.1 psi, which is only slightly above atmosphere. For later experiments though it would need to be able to handle 50psi. We need the surfaces inside to be as smooth as possible in order to best minimize noise from the system, but most importantly to try to keep cell height as uniform as possible because pressure drop goes as $1/h^3$. Thus height affects the pressure drop the most (see equation #___).

The height is controlled through the use of spacers. The default channel height is 1mm. This is built into the top plate (see appendix ___). Spacer plates are hollow plates that sit between O-rings and change the cell height. Actual height will be determined and recorded by micrometer. There is a trade off between number of heights we can reach and the chance for increased leaking due to multiple plates and O-rings. We are constrained by budget from making as many plates as we want. Overall the decision is to make a 1mm and either a .5mm or 2mm plate if it is permissible with the O-rings.

The bottom plate is not very special at all. It is a one-inch (might be later reduced due lower pressure constraints than initially thought) thick plate designed to be the structural support base. Its purpose is to serve as the anchoring point for the screws that secure the system to prevent leaking. Further design modification may serve to allow for nesting of O-rings but I

think it will be unnecessary. There are 26 boltholes for securing all the plates together without leaking. There will be imprinted tiles attached to both the top and bottom of the plate using an adhesive the releases with heat. The adhesive in question still needs to be identified.

The top plate is the most intricate and detailed piece of the whole set up. The top plate is the most detailed because it contains the most features. It will have tiles of printed pattern placed in it, the holes for securing the bolts, and the flow entrance and exit regions. The flow entrance region is the most challenging part since it requires flow to be smooth when it reaches the pressure transducers. The top plate has two pressure transducer tubing holes each 5.25 inches from the center of the flow cell. Each hole has 2mm outer diameter tubing attached and welded in that is used to connect to the differential pressure transducer. The entrance region consists of the pipe fitting attachment located in the vertical center and then that domain expands into a large region that is the full width of the cell (2 inches) and then encounters a frit which is chosen to be placed there to normalize the flow so that it is uniform upon entering the flow cell proper (see appendix). Then there is a gradual slope to the bulk of the system where it flattens out. This is designed to minimize flow disturbance and any turbulent tendencies in the flow.

Overall the flow cell's toughest criteria are the need for minute tolerances and small channel height increments. They have been addressed and are the top priority in the machining of the device. Another possible option is instead of the plates being hollow we have thin and shallow bottom plates with carved out space that go onto the original bottom plate. But this design has not been chosen because of its dramatic increase in cost.

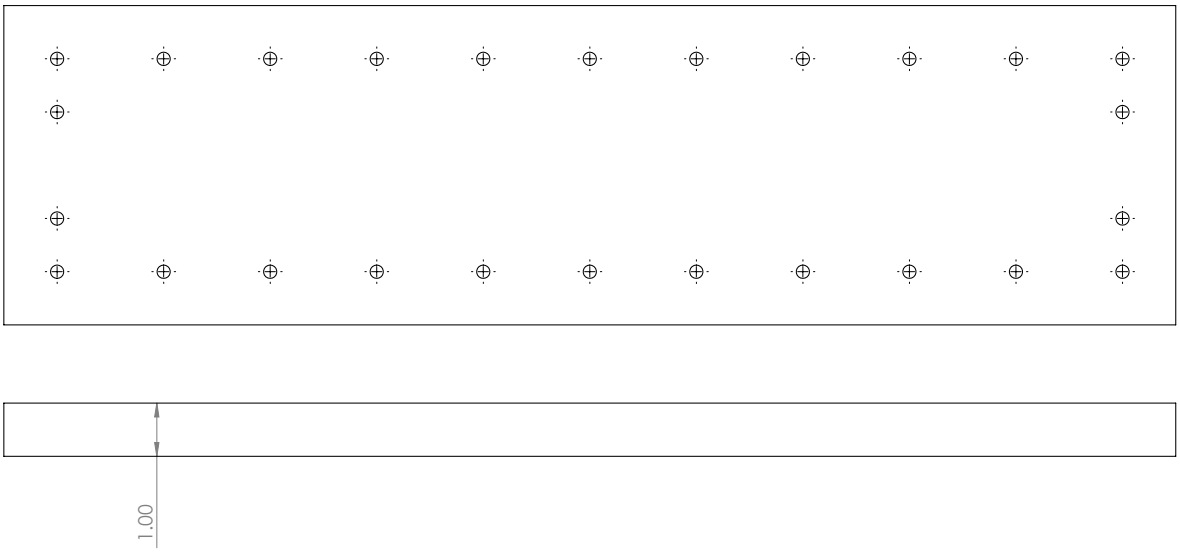
References

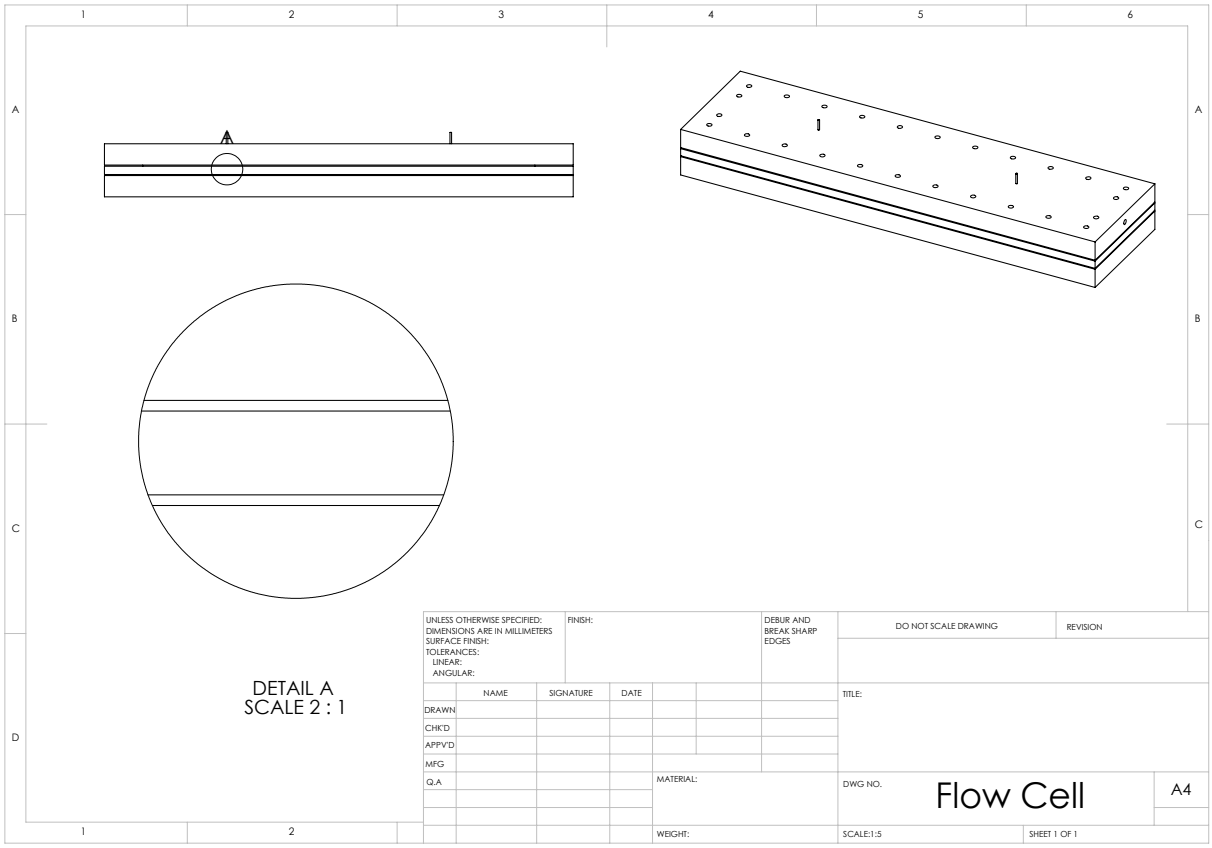
Cosden, I. A. and J. R. Lukes (2013). "A hybrid atomistic–continuum model for fluid flow using LAMMPS and OpenFOAM." Computer Physics Communications **184**(8): 1958-1965.

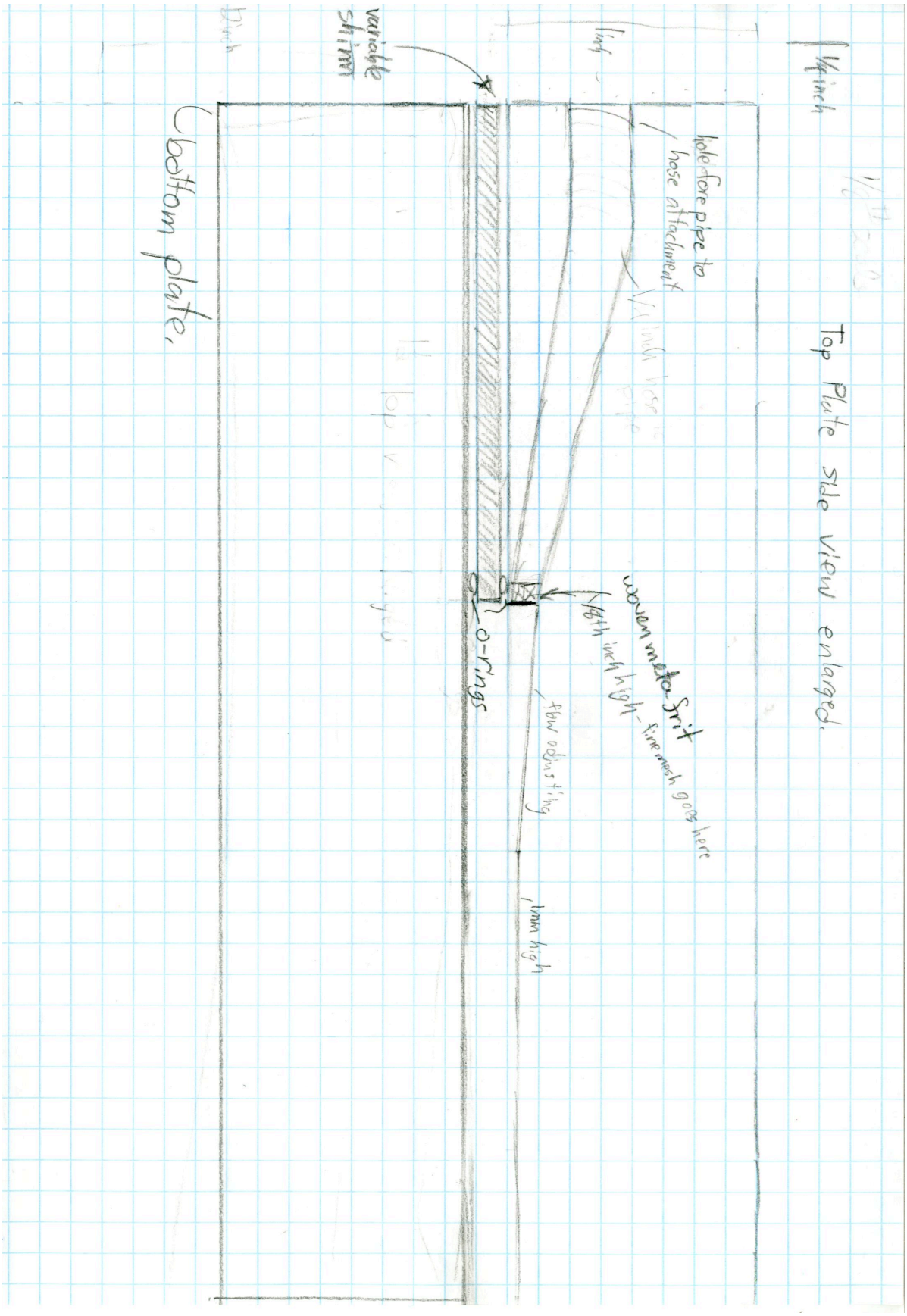
http://books.google.com/books?id=YCSSolzuu9IC&pg=PA486&lpg=PA486&dq=entrance+length+rectangular+duct&source=bl&ots=LMC06HUIRM&sig=Piq6T1vPIADMWB-hKgbyKi6EEBU&hl=en&sa=X&ei=YFc8U_P7B4SiyAHOxYDwBA&ved=0CDEQ6AEwAQ#v=onepage&q=entrance%20length%20rectangular%20duct&f=false

Fundamentals of Fluid Mechanics. Joesph A. Schetz and Allen E. Fuhs.

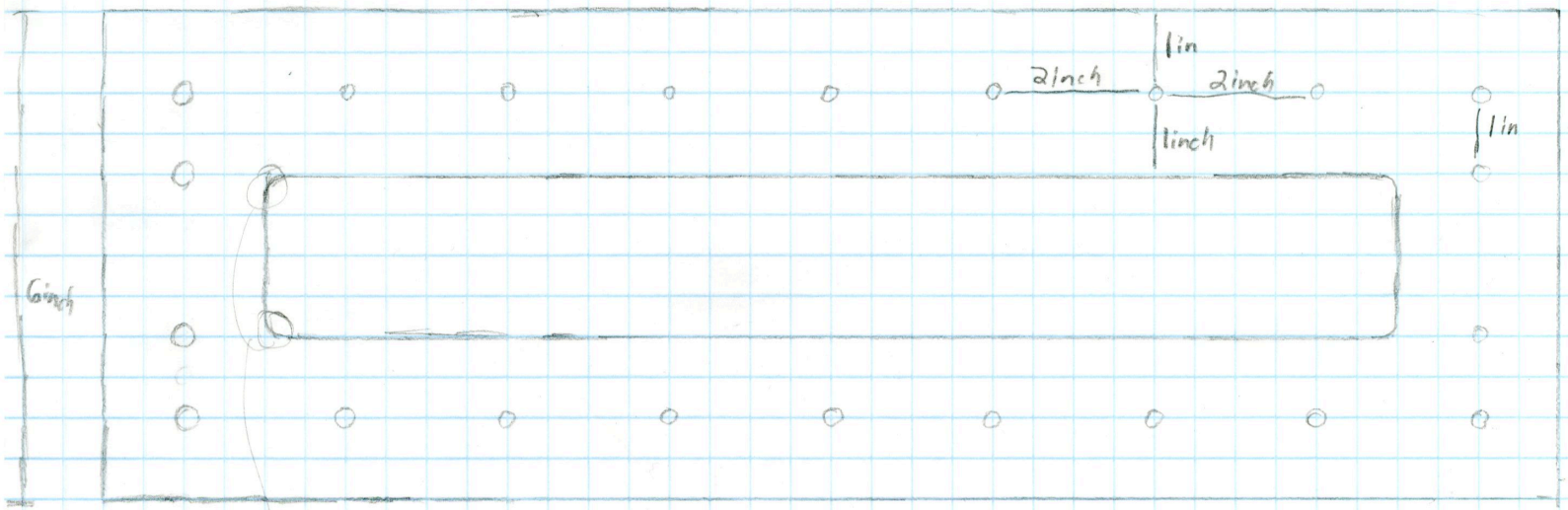
Drawings and Designs



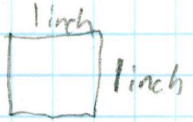

DETAIL A
SCALE 2 : 1
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBUR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
DWG NO.
Flow Cell
A4
WEIGHT:
SCALE:1:5
SHEET 1 OF 1


Top Plate side view enlarged.
1/4 inch
hole for pipe to base attachment
variable shim
bottom plate
woven mesh
1/8th inch high
fine mesh goes here
o-rings
flow adjusting
1mm high

filler plates combined with o-rings. holes are for tightning mechanism. Tolerance into paper is the highest. depths of plates are 1mm, 5mm, 10mm.



If possible round downwards as well & make corners as small as possible within reason. flow will be going over the ledges.

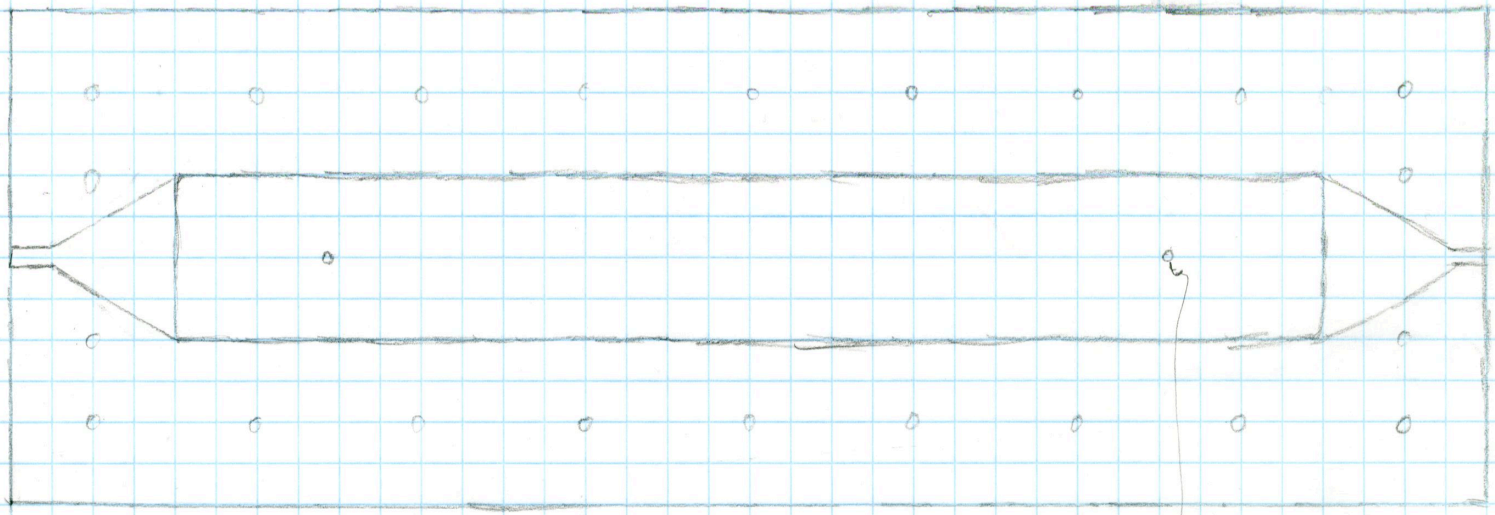


-needs to be able to run in a water bath & not rust

18 inch

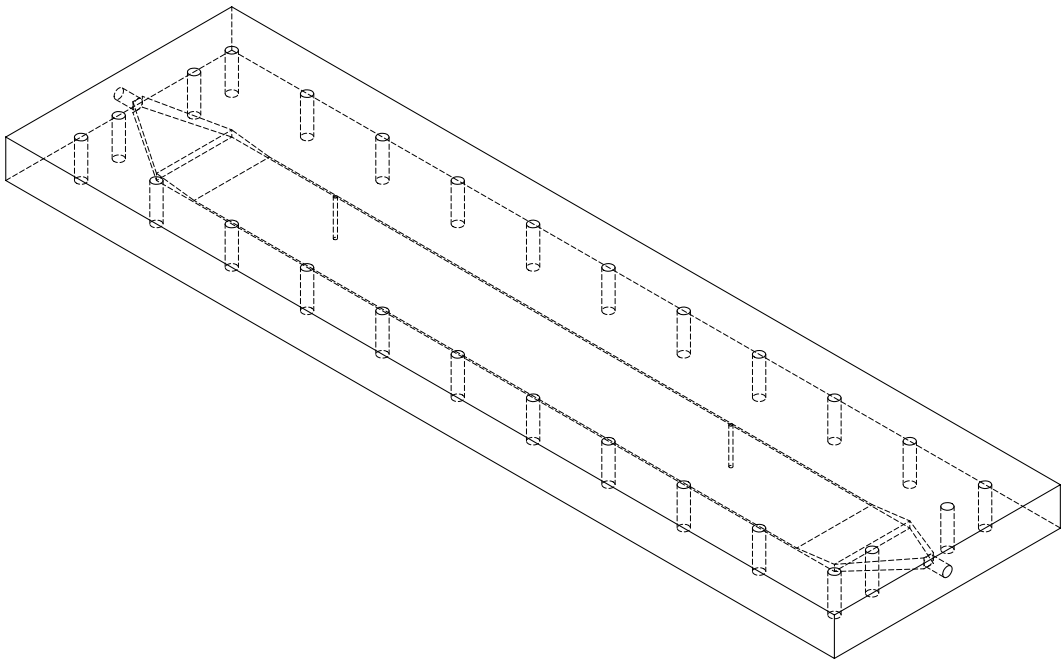
Top plate, main top of enlargement
Rounded corner
Main plate

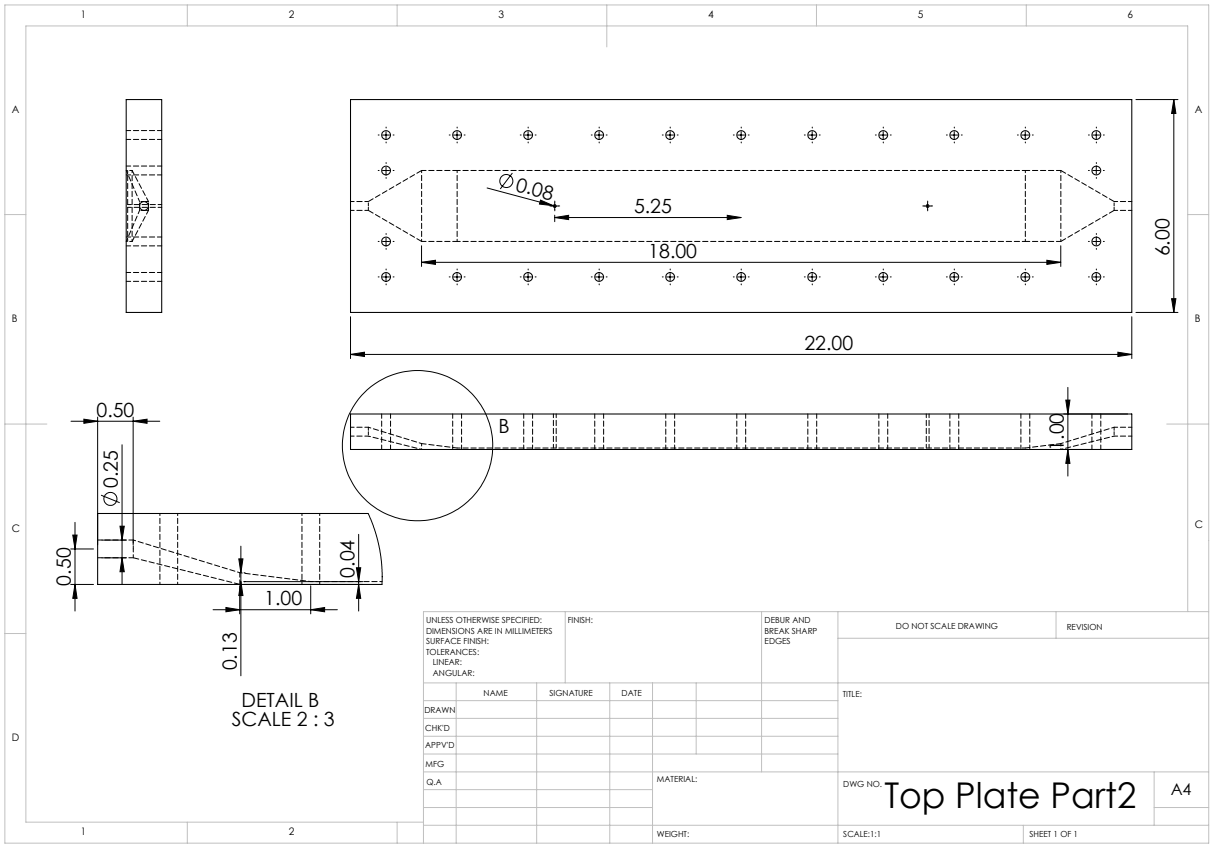
top plate - with transparent view

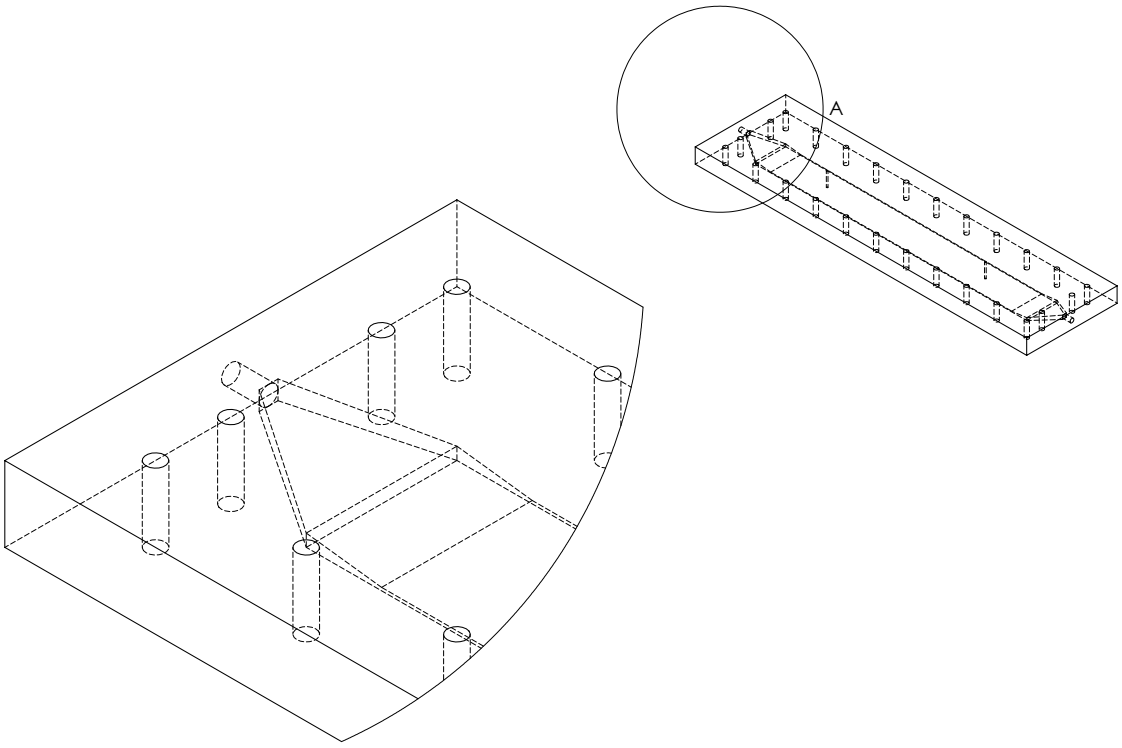


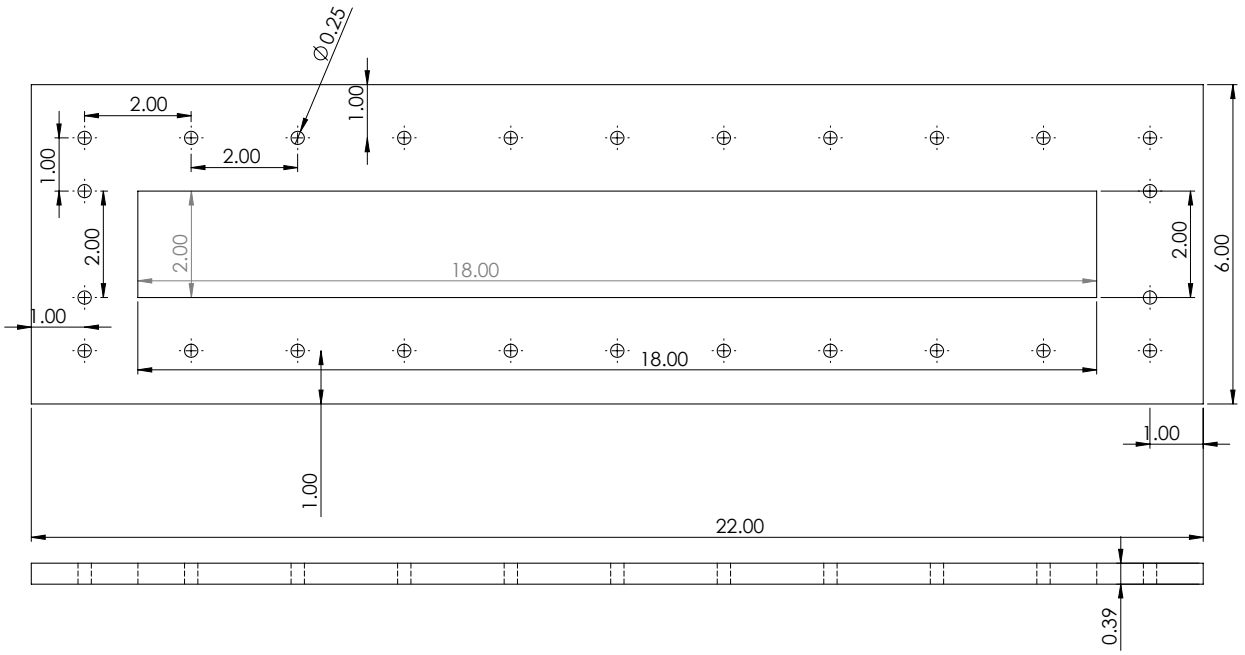
Pressure tube. goes to opening does not extend into cavity sauteed to prevent leaking exends out the top.

1in




Ø0.08
5.25
18.00
6.00
22.00
B
0.50
Ø0.25
0.50
0.04
1.00
0.13
DETAIL B
SCALE 2 : 3
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBUR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
WEIGHT:
TITLE:
DWG NO.
Top Plate Part2
A4
SCALE:1:1
SHEET 1 OF 1


A


Ø0.25
2.00
2.00
1.00
1.00
2.00
2.00
18.00
2.00
6.00
1.00
18.00
1.00
1.00
22.00
0.39

Matlab Code

WeightPost.m

```
function [ W1, W2, W3]=WeightPost(Weight)
a=length(Weight);
b=floor(a/3);
W1=zeros(b,1);
W2=zeros(b,1);
W3=zeros(b,1);
for i=1:b
    W1(i)=Weight(i*3-2);
    W2(i)=Weight(i*3-1);
    W3(i)=Weight(i*3);
end
R1=zeros(b-1,1);
R2=zeros(b-1,1);
R3=zeros(b-1,1);
for i=1:b-1
    R1(i)=W1(i+1)-W1(i);
    R2(i)=W2(i+1)-W2(i);
    R3(i)=W3(i+1)-W3(i);
end
T=[1:b-1]';
plot(T,R1,'r',T,R2,'b',T,R3,'g');legend('1st Scale', '2nd Scale', '3rd
Scale');xlabel('time (min)');ylabel('Rate (mL/min)');title('Rate vs. Time');
end
```

Averagern.m

```
function Ave=Averagern(P,n)
%P=input pressure column values
%n=number of presures to average into a single value
a=floor(length(P)/n);
Ave=zeros(a,1);
for i=1:a
    b=0;
    for j=1:n
        b=(P((i-1)*n+j))+b;
    end
    Ave(i)=b/n;
end
end
```

remover.m

```
function out=remover(input)
% Function removes spaces in weight matrix, but leaves a one in the first
% spot and not the first data.  Needs to be manually copied over.
a=length(input);
b=a/2+.5;
out=ones([1,b])';
for i=1:a
    if mod(2,i)==0

    else
        out(floor(i/2)+1)=input(i);
    end
end
end
```

Remote Access and Visualization

```
new terminal window:
ssh -X -l jome9631 login.rc.colorado.edu
```
module load slurm
sinteractive --qos=janus-debug
cd (case)
module load viz/paraview-4.0.0
paraview

(make sure that X11 is closed before running)

Milk Post Processing Steps Guide

1. Copy the labview and text file into folder labeled with date and experiment name
2. Depending upon experiment type, save appropriate template in said folder with date and name and descriptor of experiment
3. open text file, press select all and copy selected.
4. Paste in tab labeled "raw weight"
5. Open labview file, press select all (mac and PC is command a), copy (command c) and paste into tab labeled "raw pressure" (command v)
6. Click save on the excell sheet
7. Open Matlab
8. under home tab on matlab, there is a green arrow pointed down, fifth button from the left, labeled "Import Data". Click it
9. Now select the name of the excell document you just saved. You will have to change and go to the folder in which it is saved. If you don't remember that, there is a search bar on the top right you can type the name of the excell file into to find it. Then click open
10. Click on the tab at the bottom labeled "Raw Weight"
11. In VarName1 type Weight
12. on the top right there is a green checkmark labeled "Import Selection" click it
13. click on tab "Raw Pressure"
14. In "VarName1" through "VarName3" type "P1" to "P3"
15. Click on that "import selection" again.
16. Enter the following to create the average pressure. Since pressure is recorded every 5 seconds and weight is recorded every minute, we average 12 pressures to average over a minute. If a different weight measuring time is used, divide that time in seconds by 5 and use that number as a replacement for 12.
    a. P1A=Averagern(P1,12);
    b. P2A=Averagern(P2,12);
    c. P3A=Averagern(P3,12);
17. Then in the workspace double click on P1A and then copy those values and paste them in the excell spreadseet in the pressure 1 column of the "final" page.
18. Repeat for each pressure
19. copy and paste " W=remover(Weight);" into the command prompt
20. then open both W and Weight
21. Copy the first value from Weight and replace the 1 in W with it.
22. then copy and paste " [W1, W2, W3]=WeightPost(W);" into the prompt
23. the graph will show you what the rates are and will help you figure out if there is any data irregularities.
24. Open and Copy and paste the W1, W2, W3, files data into their respective slot in the excell sheet
25. you may need to extend formula and graph area to view all data results.

Sartorius Scale

1) Port --> Settings
    a) select com for scale (it depends each time, just choose one, you can change this later to the actual com)
    b) 1200 Band Rate
    c) Odd Parit
    d) 7 Data Bit
    e) 1 Stop Bit
    f) None- Flow Control
2) Port Analyze
    a) cursor in output
    b) ASCII Chart value 27 (<--)
    c) followed by capital P
    d) send
    e) quit
3) Define --> Serial output strings
    a) Interval (ms) ___ place time (be careful it is in ms not seconds)
    b) coursor to Timer Controlled Output string
    c) ASCII Chart value 27 (<--)
    d) followed by capital P
4) Define --> Hot Keys and Hot Key Action
    a) Hot Key 1
    b) Hot Key Action (Enable Timer)
    c) Hot Key Stroke (press F8 while cursor is in box- the actual F8 button, not F followed by 8)
    d) Hot Key 2
    e) Hot Key Action (Disable Timer)
    f) Hot Key Stroke (Press F9 while cursor is in box)
    g) click OK
5) Define --> Input Data Record Structure
    a) start of Record Event
        i) any character received
    b) End of Record Event
        i) Carriage Return or CrLf recieved
    c) continue
    d) Each data record contains a single data field
    e) continue
    f) Input Filter
        i) Change to "Numeric Data Only"
    g) Click in Field Postable Keystrokes
    h) click Keystroke List
    i) Select "Enter" and click okay
    j) click okay
6) Save

OpenFOAM Solution and Algorithm Control Guide with Additional Tricks and Recommendations

Reference Websites
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/TimBehrens/tibeh-report-fin.pdf
http://www.ara.bme.hu/~hernadi/OpenFOAM/
http://www.dicat.unige.it/guerrero/of2014a/14tipsandtricks.pdf
http://www.openfoamworkshop.org/6th_OpenFOAM_Workshop_2011/Program/Training/deVilliers_slides.pdf
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/TimBehrens/tibeh-slides.pdf
http://www.dicat.unige.it/guerrero/of2014a/5meshing.pdf

Different combinations for both u and p will work and not work. Some will work for p and not for u. etc... or will only work if paired in a certain way.

Linear system solvers

1. PBiCG = preconditioned bi-conjugate gradient solver for asymmetric matrices using a run time selectable preconditioner
    a. if over 1024 processors are used it is better than GAMG
2. PCG = preconditioned conjugate gradient solver for symmetric matrices using a run time selectable preconditioner
    a. if over 1024 processors are used it is better than GAMG
3. GAMG = generalized geometric-algebraic multi-grid solver
    a. Often optimal choice for pressure equation.
    b. works well for solving the pressure equation up to 1024 processors at which point the Krylov type solvers (PBiCG and PCG) tend to do better.
    c. not sure of use on U.
    d. Uses principle of generating a quick solution on a course mesh and mapping it onto a finer mesh to obtain accurate results.
    e. faceAreaPair (apparently superior to algebraicPair) agglomerator
    f. does this require similar face areas?
    g. Overview of GAMG running through its loops
        i. get the finest level interfaces from the mesh (mesh spacing?)
        ii. start agglomeration from the given faceWeights (don't really understand this part but it finds faces and then groups cells and faces together)
        iii. nCoarsestCells ends the agglomeration after the agglomeration reaches the cell size listed as the most coarse with nCoarsestCells. still not really sure how that works. Also maxLevels which places a restriction on how many of loops i and ii can be done (how may groups can be merged). It is hard coded to stop at 50.

   h. mergeLevels =1; the amount of blocks combined at each step. Best to do 1, but fore really simple mesh could use 2.
   i. the solver is running a V-cycle at which the coarsest level matrix is solved directly by specifiying "directSolve_Coarsest true" or using the iterative ICCG/BICCG by default. The number of sweeps used by the selected smoother when solving at different levels of mesh desnity are specified by nPreSweeps, nPostSweeps, and nFinestSweeps. nPreSweeps is used when V cycle is moving in coarser direction, nPostSweeps is used when algorithm is refining, and nFinestSweeps is when solution is at its finest level. Defaults are in the GAMG solver in GAMGSolver.C and can be overwritten by definittion in fvSolution.
   j. Defaults are:
      i. cacheAgglomeration_(false)
      ii. nPreSweeps_(0)
      iii. nPostSweeps_(2)
      iv. nFinestSweeps_(2)
      v. scaleCorrection_(matrix.symmetric())
      vi. directSolveCoarsest_(false)
   k. when solving using multiphase problems there may be some problems running in parallel. The problem is mainly related to nCoarsestCells keyword, so he usually has to set a high value of cells (on the order of 1000)
4. smoothSolver - solver using a smoother for both symmetric and asymmetric matrices and uses a run time selected smoother
   a. doesn't work with the other tutorial. ?? on how it works. Worked with U so long as paired with symGaussSeidel and P had PCG and DIC.
5. diagonalSolver = diagonal solver for both symmetric and asymmetric matrices.
   a. Doesn't appear to actually exist in the usable solver options (at least for asymmetric) .
6. BICCG = diagonal incomplete LU preconditioned BICG solver. Only there for compatibility with old versions. Should use PBiCG instead
7. ICC = incomplete Cholesky preconditioned conjugate gradients solver. also for back compatibility and PCG should be used instead now.
8.

Preconditioner

1. diagonalPreconditioner = Diagonal preconditioner for both symmetric and asymmetric matrices. This preconditioner actually does not help with faster propagation through the grid, but it is very easy and can be a good first step. The reciprocal of the diagonal is calculated and stored for reuse because on most systems multiplications are faster than divisions
   a. not sure if that is the right command call.
2. DIC = diagonal incomplete-Cholesky
   a. simplified diagonal- based incomplete Cholesky preconditioner for symmetric matrices (symmetric equivalent of DILU). The reciprocal of the preconditioned diagonal is calculated and stored. In lduMatrix folder it is listed as DICPreconditioner

3. DILU = diagonal incomplete LU
    a. Simplified diagonal based incomplete LU preconditioner for asymmetric matrices. The reciprocal of the preconditioned diagonal is calculated and stored.
4. GAMG = geometric agglomerated algebraic multigrid=generalised geometric-algebraic multi-grid
5. FDIC = Faster version of the DIC preconditioner for symmetric matrices in which the reciprocal of the preconditioned diagonal and the upper coefficients divided by the diagonal are calculated and stored. DIC will run fine though when FDIC doesn't, so keep that in mind.
6. noPreconditioner = Null preconditioneer for both symmetric and asymmetric matrices. No idea how or why it is called.

Smoother

1. DIC = simplified diagonal-based incomplete Cholesky smoother for symmetric matrices
2. DICGaussSeidel = combined DIC/GaussSeidel smoother for symmetric matrices in which DIC smoothing is followed by GaussSeidel to ensure that any "spikes" created by the DIC sweeps are smoothed out
3. DILU = simplified diagonal-based incomplete LU smoother for asymmetric matrices. DILU smoothers are good smoothers for linear multigrid methods
4. GaussSeidel = the Gauss Seidel method is a technique used to solve a linear system of equations. The method is an improved version of the Jacobi method. It is defined on matrices with non-zero diagonals, but convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and positive definite.
5. DILUGaussSeidel = Combined DILU/GaussSeidel smoother for asymmetric matrices in which DILU smoothing is followed by GaussSeidel to ensure that any "spikes" created by the DILU sweeps are smoothed-out.
6. symGaussSeidel
7. nonBlockingGaussSeidel
8.

Algorithms

1. PISO
2. SIMPLE

Numerical Schemes

1. d2dt2Schemes
    a. steadyState
    b. Euler
2. ddtSchemes
    a. backward
        i. second order implicit scheme (unbounded)
    b. bounded
    c. CoEuler
    d. CrankNicolson (number)

        i. second order implicit scheme (bounded)
    e. Euler
        i. first order implicit scheme (bounded)
    f. localEuler
    g. SLTS
    h. steadyState
3. gradSchemes
    a. fourth
    b. gauss
    c. LeastSquares
    d. limited
4. divSchemes
    a. gauss (requires an interpolation scheme)
5. laplacianSchemes
    a. gauss (requires also interpolation scheme and a snGradScheme)
6. interpolationSchemes
    a. biLinearFit
    b. CenteredFit
    c. clippedLinear
    d. CoBlended
    e. cubic
    f. cubicUpwindFit
    g. downwind
    h. fixedBlended
    i. harmonic
    j. llimiterBlended
    k. linear
    l. linearFit
    m. linearPureUpwindFit
    n. linearUpwind
    o. localBlended
    p. localMax
    q. localMin
    r. LUST
    s. midPoint
    t. outletStabilised
    u. pointLinear
    v. PureUpwindFit
    w. quadraticFit
    x. quadraticLinearFit
    y. quadraticLinearPureUpwindFit
    z. quadraticLinearUpwindFit
    aa. quadraticUpwindFit
    bb. reverseLinear
    cc. skewCorrected
    dd. UpwindFit

ee. weighted
ff. Centered Schemes
    i. linear = linear interpolation (central differencing)
    ii. cubicCorrection
    iii. midPoint = linear interpolation with symmetric weighting
gg. Upwinded convection schemes
    i. upwind = upwind differencing
    ii. linearUpwind = linear upwind differencing
    iii. skewLinear = linear with skewness correction
    iv. filteredLinear2 = linear with filtering for high frequency ringing
hh. TVD schemes
    i. limitedLinear
    ii. vanLeer
    iii. MUSCL
    iv. limitedCubic
ii. NVD schemes
    i. SFCD = self-filtered central differencing
    ii. Gamma = Gamma differencing

7. snGradSchemes
    a. CenteredFit
    b. corrected
    c. faceCorrected
    d. limited = requires also a number between 0 and 1 where 0 corresponds to uncorrected, 1 corresponds to corrected, .5 non orthogonal correction is less than or equal to the orthogonal part.
    e. linearFit
    f. orthogonal
    g. quadraticFit
    h. uncorrected
8. fluxRequired: is a yes or no answer.
9. convectionSchemes
    a. bounded
    b. gauss
    c. multivariateGauss

"Iterative methods for sparse linear systems" by Saad is available for free (first edition) and might be useful.

lduMatrix class is a class in which the coefficients are stored as three different arrays. (u) upper triangle, (l) lower triangle, and (d) the diagonal of the matrix. Found in directory source/OpenFOAM/matrices/lduMatrix

A preconditioned iterative solver solves the system $M^{-1}Ax = M^{-1}b$ with M being the preconditioner. The purpose of the preconditioner is to make sure that convergence for the preconditioned system is much faster than for the original one. This leads to M usually being an easily invertible approximation to A. The preconditioner leads to a faster propagation of information through the computational mesh.

Krylov Subspace solvers: PBiCG and PCG are Krylov subspace solvers. the order-r Krylov subspace generated by the n x n matrix A and the vector of n-dimension b is the linear subspace spanned by the images of b under the first r powers of A starting from the Identity matrix as A0. Basically $K_r(A,b) = span\{b, Ab, A^2b, ..., A^{r-1}b\}$ and each vector defining the subspace is quickly and easily made up by multipling a matrix b the previous vector. so b, then Ab, then A (Ab) then A (AAb) etc... The vectors tend to become almost linearly dependent very quickly; methods involving Krylov subspace frequently rely on orthogonalization schemes such as Lanczos iteration for Hermitian matrices or Arnoldi iteration for general matrices. This technique is known as Krylov subspace methods.

Although preconditioners can considerably reduce the number of iterations they do not normally reduce the mesh dependency of the number of iterations.

Numerical Schemes can be found in src/finiteVolume/finiteVolume/

missing the previously used bounded schemes

Interpolation schemes are uses for interpolations of values from cell centers to face centers. Convection specific schemes calculate the interpolation based on the flux of the flow velocity. Also which schemes can be used with the V vector ending?

checkMesh -allGeometry -allToplogy

1. all topolgical errors must be repaired
2. you can run with mesh quality errors such as skewness, aspect ratio, minimum face area, and non-orthogonality, but they will severely tamper the solution accuracy and eventually can make the solver blow up
3. checkMesh will also write a set of the faulty cells, faces, and points to the directory "constant/polyMesh/sets"
4. use of "foamToVTK -set_type name_of_sets" where set_type is the type of sets: faceSet, cellSet, pointSet, surfaceFields. Name_of_sets is the name of the set in the sets directory such as: highAspectRatioCells, nonOrthoFaces, wrongOrientedFaces, skewFaces, unusedPoints.
5. foamToVTK will create the VTK folder that can be viewed with paraView to visualize the failed sets

renumberMesh = renumbers the mesh to minimize its bandwidth helping to make the solver run faster for the beginning time steps at least. Very useful with large meshes such as I run.
Remember mesh quality is extremely important to get good results in an openFoam case

Initial Conditions

1. a good initial condition can improve the stability and convergence. Unphysical boundary conditions can slow down convergence or cause divergence
2. when possible, start off by using "potentialFoam" to get an initial solution as it is computationally inexpensive and provides a good initial condition.
3. if you get a decent solution from a course mesh, you can use mapFields to map the solution onto a finer mesh to do a higher resolution simulation.
4. if you are running a turbulence model you can initialize the velocity and pressure fields from a solution obtained from a laminar case
5. initial conditions should be physically realistic

Important Tricks and miscellaneous things

1. if "runTimeModifiable" is set to use in the controlDict than the controlDict, fvSchemes and fvSolution can be edited and updated while the simulation is running.
2. Never execute production runs or a final solution using first order schemes. They are too diffusive, which means they will under predict the forces and smear the gradients. You can start using a first order scheme and then switch to a higher order scheme (start robustly and end with accuracy).
    a. it will be important to organize the discretization schemes by order
3. time step continuity errors should be small (negative or positive), if it increases in time something is wrong.
4. If after checking the mesh quality, the non-orthogonality is higher than 80 it is prudent if possible to redo the mesh and improve the quality
5. If running LES mesh quality needs to be even better and non-orthogonality really needs to be less than 60 and as low as possible.
6. do not try to push too much the numerical scheme on highly non-orthognal meshes. You already know that the quality is low, so this highly influence the accuracy and stability of the solution
7. Generally start with first order and looser convergence criterion and then as the simulation runs switch to second order and tighter convergence criterion.

Convective terms:

1. for grad and div schemes. robust but highly diffusive
    a. cellLimited Gauss linear 1.0 (for grad)
    b. Gauss upwind
    c. Gauss linear
2. accurate and stable
    a. cellMDLimited Gauss linear 0.5 (grad scheme)
    b. Gauss linearUpwind grad(U)

diffusive terms

1. accurate schemes for orthogonal meshes
    a. laplacian --- Gauss linear corrected

   b. snGradSchemes --- corrected
2. accurate scheme for orthogonal meshes with non-orthogonal corrections
   a. laplacian --- Gauss linear limited 1
   b. snGradSchemes limited 1
3. less accurate numerical scheme valid on non-orthogonal meshes with non orthogonal corrections
   a. laplacianSchemes --- Gauss linear limited 0.5
   b. SnGradSchemes ---- limited 0.5
4.

In the event of Non-orthogonality more than 80 it is highly recommended to not waste time simulating and get a better mesh. But if it must run then it is highly recommend to use this

```
gradSchemes { default         faceLimited leastSquares 1.0;}
divSchemes { div(phi,U)       Gauss linearUpwind grad(U);}
laplacianSchemes { default    Gauss linear limited 0.333;}
snGradSchemes { default       limited 0.333;}
```

In the event of Non-orthogonality between 70 and 80

```
gradSchemes { default         cellLimited leastSquares 1.0;}
divSchemes { div(phi,U)       Gauss linearUpwind grad(u);}
laplacianSchemes { default    Gauss linear limited 0.5;}
snGradSchemes { default       limited 0.5;}
nNonOrthogonalCorrectors 4;
```

In the event of Non-orthogonality between 60 and 70

```
gradSchemes { default         cellMDLimited Gauss linear 0.5;}
divSchemes { div(phi,U)       Gauss linearUpwind grad(u);}
laplacianSchemes { default    Gauss linear limited 0.777;}
snGradSchemes { default       limited 0.777;}
nNonOrthogonalCorrectors 2;
```

In the event of Non-orthogonality between 40 and 60

```
gradSchemes { default         cellMDLimited Gauss linear 0.5;}
divSchemes { div(phi,U)       Gauss linearUpwind grad(u);}
laplacianSchemes { default    Gauss linear limited 1;}
snGradSchemes { default       limited 1;}
nNonOrthogonalCorrectors 1;
```

gradSchemes: defines the way we compute the gradients. Gradients can be computed using the Gauss Method or the Least Squares Method. In practice, the least squares method is more accurate but tends to be more oscillatory on tetrahedral meshes. Gradient limiters avoid over and under shoots on the gradient computations. there are four available: (listed from most diffusive to least diffusive) faceLimited, faceMDLimited, cellLimited, cellMDLimited.

Under-relaxation factors: because of the non-linearity of the equations being solved, it is necessary to control the change of phi. In general, under-relaxation factors are there to suppress oscillations. small under-relaxation factors will significantly slow down convergence. They can even slow down convergence to the extent that it looks converged before it is. The recommendation is to always use under-relaxation factors that are as high as possible without resulting in oscillations or divergence. To find the optimal under-relaxation factor for each case will usually take trial and error. it is recommended to start with the default values and slowly decrease the value. a good number is usually 0.3 for Pressure and 0.7 for Velocity

For pressure a good starting point is usually the bellow with the tolerances later tightened to 1e-6, and 0

```
p
{
        solver                  GAMG;
        tolerance               1e-5;
        relTol                  0.01
        smoother                GaussSeidel;
        nPreSweeps              0;
        nPostSweeps             2;
        cacheAgglomeration      on;
        agglomeerator           faceAreaPair;
        nCellsInCoarsestLevel   100;
        mergeLevels             1;
}
```

if not using GAMG starts often with this and increases tolerance to eventual finally 1e-6 and 0.

```
p
{
 solver             PCG;
 preconditioner     DIC;
 tolerance          1e-4;
 relTol             0.01;
}
```

For the velocity equation this is usually a good place to start

```
U
{
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-8;
        relTol          0;
}
```

when using the pimple solver, you have the option to limit your time step to a maximum CFL (courant) number so the solver automatically chooses the time steps.

If you are using the piso solver you need to give the time step size and cannot have it adjust automatically.

To do so, by setting the keyword "nOuterCorrectors" equal to 1 in the pimple solver is equivalent to using the piso solver and then the Courant number can be specified.

The pimple solver is a solver specially formulated for large time-steps. So in order to increase the stability you will need to add more corrector steps (nOuterCorrectors and nCorrectors)

A smaller time step may be needed in the first iterations to maintain solver stability. First time steps also may take longer to converge so do not be alarmed.

If you are interested in the initial solution, start using a high order discretization scheme, a tight convergence and the right flow properties.

If you use the first order Euler scheme, try to use a Courant number less than 1.0 and preferably in the order of 0.5 in order to keep temporal diffusion to a minimum.

First order schemes are robust and second order schemes are accurate

low residuals do not automatically mean a correct solution, nor do high residuals indicate a wrong solution. Higher order discretization schemes will often have higher final residuals than first order, but are still almost always better than the first order solution.

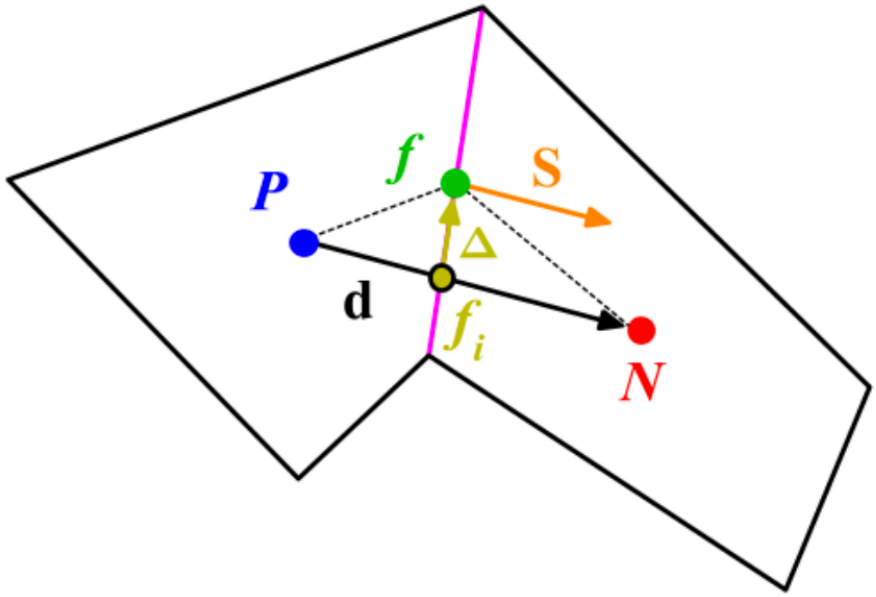
OpenFoam works best with hexahedral meshes. Change in cell size should be smooth. In boundary layers quad, hex, prism/wedge cells are preferred over triangles, tetrahedral or pyramids.

functionObject can be used to monitor the simulations.
You can use the utility "execFlowFunctionObjects" which will calculate the quantities from saved solutions. functionObject is added and defined in the "controlDict" file.

Meshing Information

1. Mesh Orthogonality = the angle that the surface vector between two cells makes with the line going between the two cell centers that share that face.
    a. Plays a very big roll in mesh quality and fvSchemes choices (see above)
    b. Affects the gradient of the face center
    c. Adds diffusion to the solution
2. Mesh Skewness = the distance between the center of the surface of the face and the point on the line (which connecting the two cell centers that share that face)

intersecting with the face surface (see the delta symbol on the attached picture)



   a. Affects the interpolation of the cell centered quantities to the face center
   b. Adds diffusion to the solution
3. Mesh Aspect Ratio = the ratio between the longest and shortest side of a cell.
   a. Large AR are okay if gradients in the long direction are small.
   b. High AR smear gradients.
4. Mesh Smoothness = is the expansion or change rate of cell size between adjacent cells. in general you do not want any length to increase by more than 20% of its neighbor's size. (l2/l1<1.2)
   a. Adds diffusion to the solution

Will want to test out "renumberMesh" and "refineWallLayer"

OpenFOAM Compile Solver

Openfoam Compile Solver
By: John Mersch IV

Steps to Compile Solver:

Making the New Solver

1. First Copy a solver's folder to a location for editing
2. Edit the solver files the SolverName.C and in the Make subfolder the file script called "files".
    a. In the "files" script
        i. first at the top line put the new solver name as "solverName.C" [if your new solver is called Geroge.C then it would go right there]
        ii. Second the file will read "EXE = $(FOAM_APPBIN)/icoFoam" (or the solver you coppied is that last word). you will change it to "EXE = $(FOAM_USER_APPBIN)/SolverName" (where SolverName is the same name used in the first line
    b. in the SolverName.C script you first need to change the name of the document and then make your changes to the solver. Be sure to be working off the newest version.

The next phase is generating the directory, if you already have a $WM_PROJECT_USER_DIR then skip the next step.

1. mkdir -p $WM_PROJECT_USER_DIR/applications/solvers
2. cp -r (pathway to the folder of the solver files you just made) $WM_PROJECT_USER_DIR/applications/solvers/SolverName where SolverName is the name of your new solver

Now Proceed to compiling steps

1. cd $WM_PROJECT_USER_DIR/applications/solvers/SolverName
2. remove the old dependency file (the file ending is ".dep" and has the name of the old solver)
3. delete the old binaries subdirectory "rm -rf linuxGccDP0pt" (it is in the make directory. you need to ender that directory with "cd make" or just manually delete the file
4. "wmake" to compile
5. "ls $FOAM_USER_APPBIN" to check to see that the new solver has compiled.

VirtualBox Instructions

Download and install VirtualBox
Download the operating system you want to use
Go through instructed set up
Once Ubuntu is installed you will need to then install Guest Additions under the Devices tab.
Allow it to run.
Then click on the folder share button on the bottom right (the image is of a folder)
click on the other button and then browse for your desired folder you want to link with the Virtual machine
then under the terminal type "sudo passwd" and type the new root password twice
then

1. Open a terminal.
2. Enter "su -" (without quotes), hit Enter.
3. Enter the root password, hit Enter.
4. Type in "usermod -a -G vboxsf username", without quotes, replacing username with the user you want to add to the group, hit Enter.
5. You may have to reboot to remount the share, the easiest way to do this is probably to reboot the VM.

and now you can access the folder that is shared.

Git Guide Summary

git Git Guide Summary

1. **Initializing GIT**
    a. Type "git init" in the terminal
2. "git status"
    a. current status of projects
3. "git add"
    a. using plane "git add" will track a file, while "git add <file> …." Will add … as a comment onto the <file> for you to see when you use "git status"
    b. when it is tracked it will be tracking the changes to the file
    c.
4. Staging Area
5. "git rm –cached <file>…" to unstage
6. "git commit –m "…" to record comments on changes
7. can use the wild card function '*.txt' for the files
8. "git log" gives all the changes and commands in order from most recent to oldest
9. remote git repository
10. "git remote add <name or remote space> <url>"
11. "git push –u origin master" adding the "-u" tells it to remember the place so next time we can just run "git push" and "master" is the default local branch name
12. "git pull origin master" pulls back the files that were pushed and allows us to see the changes.
13. "get diff" allows us to see what's different from our last commit, and "get diff HEAD" acts as a pointer and does "get diff" on the most recent commit
14. "git reset <filepath/filename>" unstages a file
15. "git checkout -- <file> removes all changes since the last commit for the file.
16. For when working on debugs or a feature, its often best to create a copy to work on. This is done by "git branch <branch name>" and once perfected can be merged with the master branch
17. Can create and visit a branch all at once by "git checkout –b <branch name>"
18. You can switch branches using the "git checkout <branch>" command
19. "git rm '.txt'" will remove all the files and stage the removal of files in the current directory
20. "git rm '<file name>'" will remove that given file
21. "git rm –r folder_of_cats" will remove all the files and folders in the given directory. It might be "git rm –r <directory name>"
22. go back to the master directory. Then "git merge <branch name>" merges the branch with the master section (might also work for higher order branching)
23. after merging to delete a branch "git branch –d <branch name>"

1) Facts about Git
    a) Git saves an entire copy of everything on each computer using it which is great in the event of corruption
    b) You can comment while not connected to the internet and update it when you can connect to the internet

   c) Very difficult to lose data so long as your remember to push after commenting
   d) Three main stages the files can reside in: committed, modified, staged
      i) Committed means that the data is safely stored in your local database
      ii) Modified means you have changed the file but have not committed it to your database
      iii) Staged means you have marked a modified file in its current version to go into your next commit snapshot
   e) The git directory is where Git stores the metadata and object database for the project
   f) The working directory is a single checkout of one version of the project. They files are placed on the disk for use or modify.
      i) I will want to be careful that when I use these I don't accidently add all the run files to the git repository or it will be bogged down
   g) The staging area is where you place files to be committed together
2) Git Workflow
   a) You modify files in your working directory
   b) You stage the files, adding snapshots of them to your staging area.
   c) You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory
3) Help options
   a) "git help <verb>"
   b) "git <verb> --help"
   c) "man git-<verb>"
4)

Setting up Git

1) Setting up your identity
   a) "git config --global user.name "<your name>"
   b) "git config --global user.email <your email>"
   c) "git config --global core.editor <txt file editor>"
   d) "git config --global merge.tool vimdiff"
   e) "git config --list" lists current settings
2) Initializing a Repository in and Existing Directory
   a) "git add <files>" can do whole folders
   b) "git commit –m '<project description/initial project version>'"
3) Cloning an Existing Repository
   a) "git clone <url>"
4) How it works
   a) So while you edit files on the desktop it keeps track of changes to files and the like. So its really odd in that sense. Not sure quite how well it will pair with Globius, and Janus, but we shall see.
5) It is important that after editing a file you use "git add" again so the newest version from the directory is staged for comment
6) To ignore a file "cat > .gitignore" followed by what you want it to ignore. Examples are "*~" go ignore any back up of files

a) There is a further section on ignore, but it looks like you'd need a better cleaned desktop anyway

7) "git diff" shows all the changes that you made with break ups for need to be commit and staged vs. unstagged. Space bar to scroll, and q to end the examination and get back to normal terminal prompt
8) to commit files can use "git commit" which allows you to then type stuff with # lines being ignored.
9) "git commit -a" gives takes all the tracked files that have been modified and stages and commits them. Can also do "git commit –a –m '<description>'" as well.

git remote add origin
https://JM4Boulder@bitbucket.org/JM4Boulder/jm4_firstreposity.git

24. "git remote add <name or remote space> <url>"
25. "git push –u origin master" adding the "-u" tells it to remember the place so next time we can just run "git push" and "master" is the default local branch name

git add <folder>*
git commit -m '<message>'
git push -u origin master

git init (only need be done once)
cd repos/jm4_firstreposity
git add <folder>
git commit -a -m '<message>'
git push

Example of Successful Running

```
rl1-1-223-93-dhcp:jm4_firstreposity Dragon$ git add Case022
rl1-1-223-93-dhcp:jm4_firstreposity Dragon$ git commit –a –m "Adding
Case 22 to the git Repository"
[master f47070d] Adding Case 22 to the git Repository
 66 files changed, 6980930 insertions(+)
 create mode 100644 Case022/.DS_Store
 create mode 100644 Case022/0/.DS_Store
 create mode 100644 Case022/0/U
 create mode 100644 Case022/0/U~
 create mode 100755 Case022/0/p
 create mode 100755 Case022/0/p~
 create mode 100644 Case022/2.99994e-06/U
 create mode 100644 Case022/2.99994e-06/p
 create mode 100644 Case022/2.99994e-06/phi
 create mode 100644 Case022/2.99994e-06/uniform/time
 create mode 100644 Case022/A0_26Case90Degrees.geo
 create mode 100644 Case022/Parallel.sh
 create mode 100644 Case022/Parallel2.sh
```

```
 create mode 100644 Case022/RunInstructions.txt
 create mode 100644 Case022/constant/.DS_Store
 create mode 100644 Case022/constant/polyMesh/.DS_Store
 create mode 100644 Case022/constant/polyMesh/boundary
 create mode 100644 Case022/constant/polyMesh/cellZones
 create mode 100644 Case022/constant/polyMesh/faceZones
 create mode 100644 Case022/constant/polyMesh/faces
 create mode 100644 Case022/constant/polyMesh/neighbour
 create mode 100644 Case022/constant/polyMesh/owner
 create mode 100644 Case022/constant/polyMesh/pointZones
 create mode 100644 Case022/constant/polyMesh/points
 create mode 100644 Case022/constant/polyMesh/sets/inside
 create mode 100755 Case022/constant/transportProperties
 create mode 100755 Case022/constant/transportProperties~
 create mode 100644 Case022/dynamicCode/.DS_Store
 create mode 100644 Case022/dynamicCode/CoutteBC/Make/SHA1Digest
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/dependencies
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/dependencyFi
les
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/dontIncludeD
eps
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/files
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/filesMacros
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/includeDeps
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/localObjectF
iles
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/objectFiles
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/options
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/darwinIntel64Gcc46DPOpt/sourceFiles
 create mode 100644 Case022/dynamicCode/CoutteBC/Make/files
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/dependencies
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/dependencyFiles
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/dontIncludeDeps
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/files
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/filesMacros
```

```
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/fixedValueFvPatchFie
ldTemplate.o
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/includeDeps
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/localObjectFiles
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/objectFiles
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/options
 create mode 100644
Case022/dynamicCode/CoutteBC/Make/linux64GccDPOpt/sourceFiles
 create mode 100644 Case022/dynamicCode/CoutteBC/Make/options
 create mode 100644
Case022/dynamicCode/CoutteBC/fixedValueFvPatchFieldTemplate.C
 create mode 100644
Case022/dynamicCode/CoutteBC/fixedValueFvPatchFieldTemplate.H
 create mode 100644
Case022/dynamicCode/CoutteBC/fixedValueFvPatchFieldTemplate.dep
 create mode 120000
Case022/dynamicCode/CoutteBC/lnInclude/fixedValueFvPatchFieldTemplate.
C
 create mode 120000
Case022/dynamicCode/CoutteBC/lnInclude/fixedValueFvPatchFieldTemplate.
H
 create mode 100644 Case022/dynamicCode/CoutteBC/lnInclude/uptodate
 create mode 100644
Case022/dynamicCode/platforms/linux64GccDPOpt/lib/libCoutteBC_feee7e64
a26080b770b89a0cc77aa23034bd0fdb.so
 create mode 100755 Case022/system/controlDict
 create mode 100644 Case022/system/decomposeParDict
 create mode 100644 Case022/system/decomposeParDict~
 create mode 100755 Case022/system/fvSchemes
 create mode 100755 Case022/system/fvSchemes~
 create mode 100755 Case022/system/fvSolution
 create mode 100755 Case022/system/fvSolution~
rl1-1-223-93-dhcp:jm4_firstreposity Dragon$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further
information.
```

```
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of
Git)

Counting objects: 60, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (49/49), done.
Writing objects: 100% (59/59), 43.69 MiB | 1.34 MiB/s, done.
Total 59 (delta 5), reused 0 (delta 0)
To https://JM4Boulder@bitbucket.org/JM4Boulder/jm4_firstreposity.git
   3ce6fae..f47070d  master -> master


Steps for Adding to Git repository
   1. git add <file/folder>
   2. git commit –a –m "comment"
   3. git push
```

To put something on the GIT Repository

1. place files and folders in the jm4_firstreposity folder in the repos folder in the user folders so
    a. /repos/jm4_firstreposity
2. then go to the jm4_firstreposity folder in the terminal
3. git add <folder(s) of interset>
4. git commit -a -m "commit"
5. git push

GMSH Janus Instillation Instructions

How to install GMSH on Janus for your use:
so when logging in, the first thing is to check to see what compiling nodes are being used.
after doing (not quite sure how to check the compiling nodes—will make a note to ask later)

but log on using janus-compile1 to 4 instead and then run or afterwards do ssh janus-compile#

```
module load intel/intel-13.0.0
module load cmake/cmake-2.8.10.2
mkdir GMSH
wget http://geuz.org/gmsh/src/gmsh-2.8.3-source.tgz
tar -xzf gmsh-2.8.3-source.tgz
cd gmsh-2.8.3-source
cmake -DCMAKE_INSTALL_PREFIX=~/GMSH
make -j
make install
cd ../GMSH/bin/
```

and to view commands
./gmsh

Derivation of Re

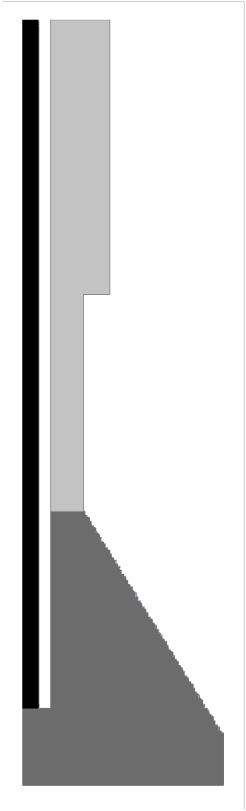


Figure A1: The impinging jet flow cell is divided into three regions. The black region is the inlet where the first Reynolds' Number is derived. The middle gray region is where the bulk Reynolds's Number is calculated. The light gray region is the exit zone and the Reynolds's Number was not calculated.

$$\mathrm{Re}_1 = \frac{QD_H}{\upsilon A} = \frac{\left(1.5\cdot 10^{-6} m^3/s\right)\left(0.0168m\right)}{\left(1\cdot 10^{-6} m^2/s\right)\left(2.2\cdot 10^{-4} m^2\right)} = 114.54$$

For the second region the domain the hydraulic diameter and cross-area are broken down as follows $\overline{A} = V/h$ and $D_H = \frac{\int_0^h \left(D_o(y) - D_i(y)\right)dy}{h}$ where $\overline{A}$ is the average cross sectional area, V is the volume and h is the height of domain. For the hydraulic diameter the annular duct formation was used and then the hydraulic diameter was integrated and normalized to provide an effective hydraulic diameter.

$$\mathrm{Re}_2 = \frac{Q\frac{\int_0^h \left(D_o(y) - D_i(y)\right)dy}{h}}{\upsilon\frac{V}{h}} = \frac{Q\int_0^h \left(D_o(y) - D_i(y)\right)dy}{\upsilon V}$$

The integral is solved by use of the trapezoid summation approximation, which is an exact solution. The volume portion is calculated using the cylinder volume formula and the conical frustum formula.

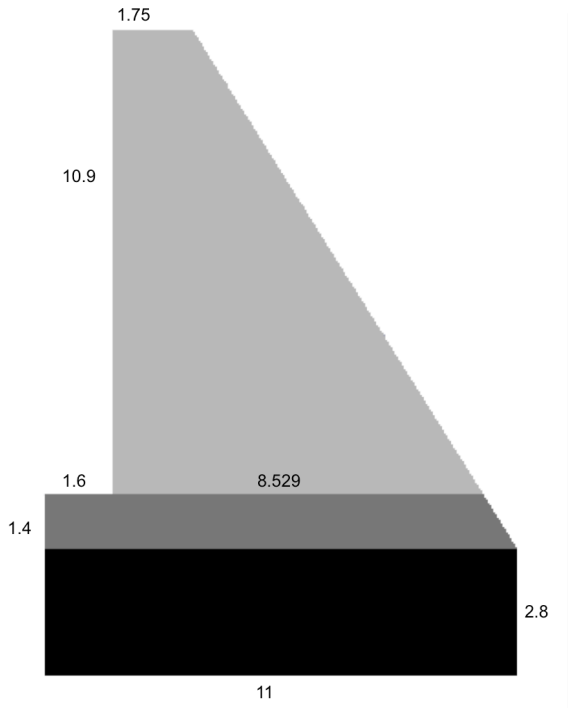


Figure A2: A breakdown of domain along with lengths necessary for trapezoidal summation for integral solution.

$$\mathrm{Re}_2 = \frac{Q\int_0^h \left(D_o(y) - D_i(y)\right)dy}{\upsilon V} = \frac{\left(1.5\cdot 10^{-6} m^3/s\right)\left(0.0102 m^2\right)}{\left(1\cdot 10^{-6} m^2/s\right)\left(0.00209 m^3\right)} = 7.3$$