Modeling and Numerical Simulations of Active and Passive Forces Using Immersed Boundary Method

A Master's Thesis

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

by

Xin Lai

December 11, 2019

Approved

Professor Roger Y. Lui
Thesis Advisor

Abstract

This thesis uses the Immersed Boundary Method (IBM) to simulate the movement of a human heart. The IBM was developed by Charles Peskin in the 70's to solve Fluid-Structure Interaction models (FSI). The heart is embedded inside a fluid (blood) which moves according to the Navier-Stokes equation. The Navier-Stokes equation is solved by the Spectral Method. Forces on the heart muscle can be divided into two kinds: Active Force and Passive Force. Passive includes the effect of curvature (Peskin's model), spring model, and the torsional spring (or beam) model. Active force is modeled by the 3-element Hill model, which was used in the 30's to model skeletal muscle. We performed simulations with different combinations of these four forces. Numerical simulations are performed using MATLAB. We downloaded Peskin's code from the Internet and modified the Force.m file to include the above four forces. This thesis only considers heart muscle movement in the organ (macroscopic) scale.

Acknowledgments

First, I would like to thank my extremely patient advisor and mentor, Professor Roger Lui, for his guidance and support since the beginning of this work. I would also like to thank the Mathematical Sciences Department of Worcester Polytechnic Institute for helping my research. For my background in the areas of mathematics necessary to complete this work, I thank the professors of WPI, in particular Roger Lui, Sarah Olson, Francis Medina, and Padraig O Cathain.

Second, I would like to thank my friends and some PHD students for their support and understanding, without whom I would have much less motivation to succeed. In particular, I express my sincere gratitude to all those who have listened and contributed ideas, including Yuchen Dong, Yulong Jiang, Liang Wang and Sam Walcott.

Finally, I would like to thank my parents for their love, support, and effort to understand my study. Without them I would not be where I am today.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Left Ventricle

The heart chambers are separated by a wall of tissue called the septum. Each side has a small heart chamber called the atrium, which leads into a large pumping chamber called a ventricle. The heart has four heart chambers. The atria are the upper two chambers, and the ventricles are the lower two chambers.

The heart is about the size of the clenched fist. It lies in the front and middle of the chest, behind and slightly to the left of the breastbone. It is an organ that pumps blood to all parts of your body to provide it with the oxygen and nutrients in needs to function.

The left ventricle is one of the four chambers of the heart. It is located in the left portion of the heart below the left atrium, separated by the mitral valve. The wall of the chamber is about 9 to 12 mm thick, and it is 2 to 3 times size larger than right ventricle. In order to adapt to the function of the left ventricle, the chamber has a conical shape with a bottom upward and a sharp left to the front. The left ventricular cavity is divided into two parts, the inflow channel and the outflow channel, by the anterior flap of the mitral valve. As the heart contracts, blood eventually flows back into the left atrium, and then through the mitral valve, whereupon it next enters the left ventricle. Blood is pumped through the chambers, aided by four heart valves. The valves open and close to let the blood flow in only one direction. The physiological load on the ventricles requiring pumping of blood throughout the body and lungs is much greater than the pressure generated by the atria to fill the ventricles. Further, the left ventricle has thicker walls than the right because it needs to pump blood to most of the body while the right ventricle fills only the lungs. The normal blood flow is a cycle that flows like this; body-heart-lungs-heart-body.

1.2 Navier-Stokes Equations

Navier–Stokes equations are useful because they describe the physics of many phenomena of scientific and engineering interest. They may be used to model the weather, ocean currents, water flow in a pipe and air flow around a wing. The Navier–Stokes equations, in their full and simplified forms, help with the design of aircraft and cars, the study of blood flow, the design of power stations, the analysis of pollution, and many other things. Coupled with Maxwell's equations, they can be used to model and study magnetohydrodynamics.

Navier-Stokes equations is one of the most important explanation to calculate fluid spread and velocity influenced by the force.

In physics, the Navier–Stokes equations, named after Claude-Louis Navier and George Gabriel Stokes, describe the motion of viscous fluid substances.

These balance equations arise from applying Isaac Newton's second law to fluid motion, together with the assumption that the stress in the fluid is the sum of a diffusing viscous term (proportional to the gradient of velocity) and a pressure term.

The Navier–Stokes equations are also of great interest in a purely mathematical sense. Despite their wide range of practical uses, it has not yet been proven whether solutions always exist in three dimensions and, if they do exist, whether they are smooth

The Navier–Stokes equations are strictly a statement of the balance of momentum. To fully describe fluid flow, more information is needed, how much depending on the assumptions made. This additional information may include boundary data, conservation of mass, balance of energy, and/or an equation of state.The Naiver-Stokes equations are given as following:

$$\rho\left(\frac{\partial \mathbf{u}(x,t)}{\partial t} + \mathbf{u}(x,t)\cdot\nabla\mathbf{u}(x,t)\right) = -\nabla p(x,t) + \mu\Delta\mathbf{u}(x,t) + \mathbf{f}(x,t)$$

For this case, in order to keep the fluid incompressible, we assume

$$\nabla\cdot\mathbf{u}(x,t) = 0$$

The Navier–Stokes equations, even when written explicitly for specific fluids, are rather generic in nature and their proper application to specific problems can be very diverse. This is partly because there is an enormous variety of problems that may be modeled, ranging from as simple as the distribution of static pressure to as complicated as multiphase flow driven by surface tension.

In this thesis, the Navier-Stokes equation will be needed to model blood flow in the heart.

1.3 Immersed Boundary Methods

Immersed boundary methods were first introduced by Peskin (1972) [2] to simulate cardiac mechanics and associated blood flow. For this topic, the following materials we cited the knowledge from Jiyuan Tu's book directly [10, 11]. One distinguishing feature of this approach is the ability to perform the entire simulation on a fixed Cartesian grid. A novel procedure was developed by directly imposing the effect of the immersed boundary on the flow; the requirement for the grids to conform to the complex geometrical structure of the heart was thus avoided. Since its inception, numerous modifications and refinements have been proposed, and a number of variants of this approach currently exist.

When applying an immersed boundary method to solve for fluid flow in a complex geometry, we select a (usually) rectangular domain that contains both the fluid region and also the bounding solid. The resulting rectangular domain now contains fluid regions where the fluid velocity is governed by the Navier-Stokes equations and solid regions where the velocity is zero, or given. The domain is resolved by a regular structured grid and some grid points (or control volumes) are in the fluid region and some are in the solid region.

In the 1970s, Peskin developed the immersed-boundary method to simulate flexible membranes in fluid flows. The membrane-fluid interaction is accomplished by distributing membrane forces as local fluid forces and updating membrane configuration according to local flow velocity. Since then, the immersed-boundary method has been widely employed to study various situations, including cell deformation in micropipettes, leukocyte adhesion and movement, multiphase flows, red blood cells deformation and aggregation in shear flows and the behavior of biofilms.

In immersed-boundary method, the key idea is that the membrane force $f(x_m)$at a membrane marker x_m induced by membrane deformation is distributed to the nearby fluid grid points x_f by [3]:

$$f(x_f) = \sum_{x_m} D(x_f - x_m)F(x_m) \tag{1.1}$$

through a discrete delta function $D(x)$, which is chosen to approximate the properties of the Dirac delta function.

Chapter 2

Immersed Boundary Method

The material in this section comes from Peskin's paper 2002 [3]. In this chapter, we will describe some numerical methods which is essential to help us understand and solve the Navier-Stokes equation in 2D.

In the study, we apply the immersed boundary method developed by Charles Peskin. The fluid is represented on the Eulerian grid, and the structure is represented on the Lagrangian coordinate.

Recall the Navier-Stokes equations to solve our fluid motion in two dimensions is

$$\begin{aligned} \rho\left(\frac{\partial \mathbf{u}(x,t)}{\partial t} + \mathbf{u}(x,t)\cdot\nabla\mathbf{u}(x,t)\right) &= -\nabla p(x,t) + \mu\Delta\mathbf{u}(x,t) + \mathbf{f}(x,t) \\ \nabla\cdot\mathbf{u}(x,t) &= 0 \end{aligned}$$

where $\mathbf{u}(x,t) = (u(x,t), v(x,t))$ is the fluid velocity, $p(x,t)$ is the pressure, and $\mathbf{f}(x,t)$ is the force per unit volume (area in 2D) applied to the fluid by the immersed boundary.

2.1 Notation

$$\text{Let} \quad \mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix}$$

The nonlinear term in the Navier-Stokes equation in 2D is,

$$\mathbf{u}\cdot\nabla\mathbf{u} = \begin{pmatrix} u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} \\ u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} \end{pmatrix}$$

We use the notation that D_h^0 is the vector difference operator. Thus, $D_h^0\phi$ is the central difference approximation to the gradient of ϕ, and $D_h^0\cdot u$ is the central difference approximation

to the divergence of u.
Let,

$$D\alpha = \begin{pmatrix} \dfrac{\partial \alpha}{\partial x} \\ \dfrac{\partial \alpha}{\partial y} \end{pmatrix}$$

Therefore, we have

$$\begin{aligned} S(\mathbf{u})\alpha &= \frac{1}{2}\mathbf{u}\cdot\mathbf{D}\alpha + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{D}\cdot(\mathbf{u}\alpha) \\ &= \frac{1}{2}\Big(u\frac{\partial \alpha}{\partial x} + v\frac{\partial \alpha}{\partial y}\Big) + \frac{1}{2}D\cdot\begin{pmatrix} u\alpha \\ v\alpha \end{pmatrix} \\ &= \frac{1}{2}\Big(u\frac{\partial \alpha}{\partial x} + v\frac{\partial \alpha}{\partial y}\Big) + \frac{1}{2}\Big((u\alpha)_x + (v\alpha)_y\Big) \\ &= \frac{1}{2}\Big(u\frac{\partial \alpha}{\partial x} + v\frac{\partial \alpha}{\partial y}\Big) + \frac{1}{2}\Big(u_x\alpha + u\alpha_x + v_y\alpha + v\alpha_y\Big) \\ &= u\frac{\partial \alpha}{\partial x} + v\frac{\partial \alpha}{\partial y} + \frac{1}{2}\Big(u_x + v_y\Big)\alpha \\ &= u\frac{\partial \alpha}{\partial x} + v\frac{\partial \alpha}{\partial y} \end{aligned} \tag{2.1}$$

because $u_x + v_y = 0$, which follows from the incoompressibility condition $\nabla\cdot\mathbf{u} = \mathbf{0}$

2.2 Numerical Method

The temporal discretization that we currently use (Lai and Peskin 2000, McQueen and Peskin 2001) [8, 9] is a second-order Runge–Kutta method, based primarily on the midpoint rule. For a system of ordinary differential equations of the for

$$\frac{dy}{dt} = f(y)$$

such a scheme looks like this:

$$\frac{y^{n+\frac{1}{2}} - y^n}{\Delta t/2} = f(y^n) \tag{2.2}$$

$$\frac{y^{n+1} - y^n}{\Delta t} = f(y^{n+\frac{1}{2}}) \tag{2.3}$$

where the superscript is the time-step index. The salient feature of this scheme is that each time-step involves a 'preliminary substep' to the halftime level followed by a 'final sub-step' from time level n to $n+1$, in which the results of the preliminary substep are

used. The preliminary sub-step involves a first-order accurate scheme (forward Euler in the above example), and the final substep is done by a second-order accurate scheme (here, the midpoint rule). One would think that the accuracy would be limited by the least accurate sub-step, but the magic of Runge–Kutta is that the overall scheme is second-order accurate.

2.3 First-Half Step

The following formula are taken from page 507 of Peskin (2002) [3]. Consider the half step and integrate the Navier–Stokes equations on the Eulerian grid g_h from time level n to time level $n + \frac{1}{2}$.

$$\begin{aligned} \rho^{n+\frac{1}{2}} \left(\frac{\mathbf{u}^{n+\frac{1}{2}} - \mathbf{u}^n}{\Delta t/2} + S_h(\mathbf{u}^n)\mathbf{u}^n \right) + \mathbf{D}_h^0 \tilde{p}^{n+\frac{1}{2}} &= \mu L_h \mathbf{u}^{n+\frac{1}{2}} + \mathbf{f}^{n+\frac{1}{2}} \\ \mathbf{D}_h^0 \cdot \mathbf{u}^{n+\frac{1}{2}} &= 0 \end{aligned} \tag{2.4}$$

From the above definition, since $\mathbf{u^n} = \begin{pmatrix} u^n \\ v^n \end{pmatrix}$

From (2.1),

$$S_h(\mathbf{u}^n)\,\mathbf{u}^n = \begin{pmatrix} u^n u_x^n + v^n u_y^n \\ u^n v_x^n + v^n v_y^n \end{pmatrix}$$

From the first component of (2.4), we have

$$u^{n+\frac{1}{2}} = u^n + \frac{\Delta t/2}{\rho^{n+\frac{1}{2}}} \left(f^{n+\frac{1}{2}} - \rho^{n+\frac{1}{2}} (u^n u_x^n + v^n u_y^n) \right) \tag{2.5}$$

Similarity, we have $v^{n+\frac{1}{2}}$

$$v^{n+\frac{1}{2}} = v^n + \frac{\Delta t/2}{\rho^{n+\frac{1}{2}}} \left(f^{n+\frac{1}{2}} - \rho^{n+\frac{1}{2}} (u^n v_x^n + v^n v_y^n) \right) \tag{2.6}$$

Note that this is the formula without pressure and Laplacian terms.

2.4 Spectral Method

The spectral method comes from Trefethen's book [7]. The Fast Fourier Transform of the function p evaluated at the grid-point (mh_x, nh_y) is

$$\hat{p} = \sum_{j,k} p_{j,k}\, e^{imh_x 2\pi j + inh_y 2\pi k}$$

Using central difference, the second partial derivative of p at the same grid-point is given by

$$\hat{p}_{xx} = \sum_{j,k} \frac{1}{h_x^2}\Big(e^{ih_x 2\pi j} + e^{-ih_x 2\pi j} - 2\Big)e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k}$$

Let $\theta = h_x\, 2\pi j$, then above becomes

$$\hat{p}_{xx} = \sum_{j,k} \frac{1}{h_x^2}(e^{i\theta} + e^{-i\theta} - 2)e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k}$$

Since,

$$\begin{aligned} e^{i\theta} + e^{-i\theta} &= \cos\theta + i\sin\theta + \cos(-\theta) + i\sin(-\theta) \\ &= 2\cos\theta \\ 2\cos\theta - 2 &= 2\Big(1 - 2\sin^2\Big(\frac{\theta}{2}\Big)\Big) - 2 \\ &= -4\sin^2\Big(\frac{\theta}{2}\Big) \end{aligned}$$

Then,

$$\begin{aligned} \hat{p}_{xx} &= \sum_{j,k} \frac{1}{h_x^2}\Big(-4\sin^2\Big(\frac{h_x 2\pi j}{2}\Big)\Big)e^{imh_x 2\pi j + inh_y 2\pi k} p_{j,k} \qquad (2.7) \\ &= -\sum_{j,k} e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k} \frac{\sin^2\Big(\frac{h_x 2\pi j}{2}\Big)}{(\frac{h_x}{2})^2} \\ &= -\sum_{j,k} e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k}\Big(\frac{\sin(dx\ 2\pi j)}{dx}\Big)^2 \qquad (2.8) \end{aligned}$$

where $dx = \dfrac{h_x}{2}$.
For the first derivative,

$$\begin{aligned} \hat{p}_x &= \sum_{j,k} \frac{1}{2h_x}\Big(e^{ih_x 2\pi j} - e^{-ih_x 2\pi j}\Big)e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k} \\ &= \sum_{j,k} \frac{1}{2h_x} 2i\sin(h_x 2\pi j)e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k} \\ &= i\sum_{j,k} \frac{\sin(h_x 2\pi j)}{h_x} e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k} \qquad (2.9) \end{aligned}$$

In Fourier space, (2.4) becomes

$$\begin{aligned} \rho^{n+\frac{1}{2}}\Big(\frac{\hat{\mathbf{u}}^{n+\frac{1}{2}} - \hat{\mathbf{u}}^n}{\Delta t/2} + \widehat{S_h(\mathbf{u}^n)\mathbf{u}^n}\Big) &= -\nabla\hat{p}^{n+\frac{1}{2}} + \hat{\mathbf{f}}^{n+\frac{1}{2}} + \mu L_h \hat{\mathbf{u}}^{n+\frac{1}{2}} \\ \nabla\cdot\hat{\mathbf{u}}^{n+\frac{1}{2}} &= 0 \end{aligned}$$

Rearranging,

$$\begin{aligned}
\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}-\mu L_h\right)\hat{\mathbf{u}}^{n+\frac{1}{2}} &= -\nabla\hat{p}^{n+\frac{1}{2}}+\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left(\hat{\mathbf{u}}^n+\frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\hat{\mathbf{f}}^{n+\frac{1}{2}}-\frac{\Delta t}{2}\,\widehat{S_h(\mathbf{u}^n)\mathbf{u}^n}\right)\\
\nabla\cdot\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}-\mu L_h\right)\hat{\mathbf{u}}^{n+\frac{1}{2}} &= -\Delta\hat{p}^{n+\frac{1}{2}}+\nabla\cdot\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left(\hat{\mathbf{u}}^n+\frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\hat{\mathbf{f}}^{n+\frac{1}{2}}-\frac{\Delta t}{2}\,\widehat{S_h(\mathbf{u}^n)\mathbf{u}^n}\right)\\
\Delta\hat{p}^{n+\frac{1}{2}} &= \frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left((\widehat{rhs_u})_x+(\widehat{rhs_v})_y\right) \qquad (2.10)
\end{aligned}$$

The left hand side is zero because of the incompressibility condition. The right side becomes,

$$\Delta\hat{p}^{n+\frac{1}{2}}=\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left((\widehat{rhs_u})_x+(\widehat{rhs_u})_y\right) \qquad (2.11)$$

where,

$$\begin{pmatrix}rhs_u\\ rhs_v\end{pmatrix}=-\frac{\Delta t}{2}S_h(\mathbf{u})\mathbf{u}^n+\frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\mathbf{f}^{n+\frac{1}{2}} \qquad (2.12)$$

To simplified, we use $\hat{p}$ substituting $\hat{p}^{n+\frac{1}{2}}$ Since, from (2.8), we have

$$\begin{aligned}
\Delta\hat{p} &= \hat{p}_{xx}+\hat{p}_{yy}\\
\Delta\hat{p} &= -\sum_{j,k}\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2+\left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)e^{imh_x2\pi j+inh_y2\pi k}\ p_{j,k}
\end{aligned}$$

From above,

$$\Delta\hat{p}_{j,k}^{n+\frac{1}{2}}=-\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2+\left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)\ p_{j,k} \qquad (2.13)$$

And from (2.9), we have

$$\begin{aligned}
(\widehat{rhs_u})_x &= i\sum_{j,k}\left(\frac{\sin(2\pi k\ dx)}{dx}\right)e^{imh_x2\pi j+inh_y2\pi k}\ rhs_u\\
(\widehat{rhs_v})_y &= i\sum_{j,k}\left(\frac{\sin(2\pi j\ dy)}{dx}\right)e^{imh_x2\pi j+inh_y2\pi k}\ rhs_v
\end{aligned}$$

Equating (2.11) and (2.13), for each distinct $p_{j,k}$ we have

$$\begin{aligned}
-\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2+\left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)\ p_{j,k}^{n+\frac{1}{2}}&\\
=\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left(i\left(\frac{\sin(2\pi j\ dx)}{dx}\right)\ rhs_u+i\left(\frac{\sin(2\pi k\ dy)}{dy}\right)\ rhs_v\right)&
\end{aligned}$$

This formula uses to find $\hat{p}^{n+\frac{1}{2}}$

2.5 Back to the First-Half Step

From the above section, we find the pressure term $\hat{p}_{j,k}$. We now solve for the fluid velocity term $\mathbf{u^{n+\frac{1}{2}}}$. From (2.4),

$$\begin{aligned}
\rho^{n+\frac{1}{2}}\left(\frac{\hat{\mathbf{u}}^{n+\frac{1}{2}}-\hat{\mathbf{u}}^{n}}{\Delta t/2}+\widehat{S_h(\mathbf{u}^n)\mathbf{u}^n}\right) &= -\nabla\hat{p}^{n+\frac{1}{2}}+\hat{\mathbf{f}}^{n+\frac{1}{2}}+\mu L_h\,\hat{\mathbf{u}}^{n+\frac{1}{2}} \\
\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}-\mu L_h\right)\hat{\mathbf{u}}^{n+\frac{1}{2}} &= -\nabla\hat{p}^{n+\frac{1}{2}}+\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left(\hat{\mathbf{u}}^{n}+\frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\hat{\mathbf{f}}^{n+\frac{1}{2}}-\frac{\Delta t}{2}\,\widehat{S_h(\mathbf{u}^n)\mathbf{u}^n}\right) \\
&= -\nabla\hat{p}^{n+\frac{1}{2}}+\frac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\left(\hat{\mathbf{u}}^{n}+\begin{pmatrix}\widehat{rhs_u}\\ \widehat{rhs_v}\end{pmatrix}\right)
\end{aligned}$$

We know that

$$\hat{\mathbf{u}}^{n+\frac{1}{2}}=\begin{pmatrix}\hat{u}^{n+\frac{1}{2}}\\ \hat{v}^{n+\frac{1}{2}}\end{pmatrix} \tag{2.14}$$

Thus, from (2.9) and (2.4), we have

$$\begin{aligned}
\hat{u}^{n+\frac{1}{2}} &= \frac{-\nabla\hat{p}+\dfrac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\,\widehat{rhs_u}}{\left(\dfrac{\rho^{n+\frac{1}{2}}}{\Delta t/2}-\mu L_h\right)} \\
&= \frac{\widehat{rhs_u}-i\left(\dfrac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\right)\left(\dfrac{\sin(2\pi j\,dx)}{dx}\right)\hat{p}^{n+\frac{1}{2}}}{1-\dfrac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\mu L_h}
\end{aligned} \tag{2.15}$$

Similarly, to the second component of $\mathbf{u^{n+\frac{1}{2}}}$, we have

$$\begin{aligned}
\hat{v}^{n+\frac{1}{2}} &= \frac{-\nabla\hat{p}+\dfrac{\rho^{n+\frac{1}{2}}}{\Delta t/2}\,\widehat{rhs_v}}{\left(\dfrac{\rho^{n+\frac{1}{2}}}{\Delta t/2}-\mu L_h\right)} \\
&= \frac{\widehat{rhs_v}-i\left(\dfrac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\right)\left(\dfrac{\sin(2\pi k\,dy)}{dy}\right)\hat{p}^{n+\frac{1}{2}}}{1-\dfrac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\mu L_h}
\end{aligned} \tag{2.16}$$

We now find L_h in (2.15) and (2.16)

From (2.7), we have

$$\begin{aligned}\hat{\mathbf{u}}_{xx}^{n+\frac{1}{2}} &= \sum_{j,k} \frac{1}{h_x^2}\left(-4\sin^2\left(\frac{h_x 2\pi j}{2}\right)\right) e^{imh_x 2\pi j + inh_y 2\pi k}\mathbf{u}_{j,k}^{n+\frac{1}{2}} \\ &= -\sum_{j,k} e^{imh_x 2\pi j + inh_y 2\pi k}\,\frac{\sin^2\left(\frac{h_x 2\pi j}{2}\right)}{\left(\frac{hx}{2}\right)^2}\mathbf{u}_{j,k}^{n+\frac{1}{2}}\end{aligned}$$

Let $dx = \frac{hx}{2}$ and $dy = \frac{hy}{2}$, then

$$L_h = -\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2 - \left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2$$

The denominator of (2.15) and (2.16) is equal to

$$1 - \frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\,\mu L_h = 1 + \frac{\Delta t/2}{\rho^{n+\frac{1}{2}}}\mu\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2 + \left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)$$

Equations (2.15) and (2.16) allow us to find $\hat{u}^{n+\frac{1}{2}}$ and $\hat{v}^{n+\frac{1}{2}}$, then apply the inverse Fourier series to obtain $\begin{pmatrix} u^{n+\frac{1}{2}} \\ v^{n+\frac{1}{2}}\end{pmatrix}$.

2.6 Second-Half Step

Now we proceed directly to the Navier–Stokes equations. We are going to integrate the Navier-Stokes equations from time level n to the time level $n+1$ using (2.3). We use a similar method in above section.

$$\begin{aligned}\rho^{n+\frac{1}{2}}\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + S_h(\mathbf{u}^{n+\frac{1}{2}})\mathbf{u}^{n+\frac{1}{2}}\right) + \mathbf{D}_h^0\, p^{n+\frac{1}{2}} &= \mu L_h\left(\frac{\mathbf{u}^n+\mathbf{u}^{n+1}}{2}\right) + \mathbf{f}^{n+\frac{1}{2}} \quad (2.17)\\ \mathbf{D}_h^0\cdot\mathbf{u}^{n+1} &= 0\end{aligned}$$

Thus, we could find our $S_h(\mathbf{u}^{n+\frac{1}{2}})\mathbf{u}^{n+\frac{1}{2}}$, using the notation in section 2.1,

$$\begin{aligned}S_h(\mathbf{u}^{n+\frac{1}{2}})\,u^{n+\frac{1}{2}} &= \frac{1}{2}\mathbf{u}^{n+\frac{1}{2}}\cdot(D_h^0 u^{n+\frac{1}{2}}) + \frac{1}{2}D_h^0\cdot(\mathbf{u}^{n+\frac{1}{2}}u^{n+\frac{1}{2}}) \\ &= u^{n+\frac{1}{2}}u_x^{n+\frac{1}{2}} + v^{n+\frac{1}{2}}u_y^{n+\frac{1}{2}} \quad (2.18)\end{aligned}$$

$$\begin{aligned}S_h(\mathbf{u}^{n+\frac{1}{2}})\,v^{n+\frac{1}{2}} &= \frac{1}{2}\mathbf{u}^{n+\frac{1}{2}}\cdot(D_h^0 v^{n+\frac{1}{2}}) + \frac{1}{2}D_h^0\cdot(\mathbf{u}^{n+\frac{1}{2}}v^{n+\frac{1}{2}}) \\ &= u^{n+\frac{1}{2}}v_x^{n+\frac{1}{2}} + v^{n+\frac{1}{2}}v_y^{n+\frac{1}{2}} \quad (2.19)\end{aligned}$$

Therefore, we have $S_h(\mathbf{u}^{n+\frac{1}{2}})\mathbf{u}^{n+\frac{1}{2}}$,

$$S_h(\mathbf{u}^{n+\frac{1}{2}})\,\mathbf{u}^{n+\frac{1}{2}} = \begin{pmatrix} u^{n+\frac{1}{2}}u_x^{n+\frac{1}{2}} + v^{n+\frac{1}{2}}u_y^{n+\frac{1}{2}} \\ u^{n+\frac{1}{2}}v_x^{n+\frac{1}{2}} + v^{n+\frac{1}{2}}v_y^{n+\frac{1}{2}} \end{pmatrix} \tag{2.20}$$

From (2.17), we have:

$$\Big(1 - \frac{\Delta t}{\rho^{n+\frac{1}{2}}}\,\frac{\mu L_h}{2}\Big)u^{n+1} = u^n + \frac{\Delta t}{\rho^{n+\frac{1}{2}}}\left(f^{n+\frac{1}{2}} - \mu L_h\frac{u^n}{2} - \rho^{n+\frac{1}{2}}u^{n+\frac{1}{2}}u_x^{n+\frac{1}{2}} - \rho^{n+\frac{1}{2}}v^{n+\frac{1}{2}}u_y^{n+\frac{1}{2}}\right)$$

Note that this is the formula without pressure term.
Apply for the fast Fourier Transform again.

Equation (2.17) in Fourier space is

$$\begin{aligned} \rho^{n+\frac{1}{2}}\left(\frac{\hat{\mathbf{u}}^{n+1}-\hat{\mathbf{u}}^n}{\Delta t} + \widehat{S_h(\mathbf{u}^{n+\frac{1}{2}})\mathbf{u}^{n+\frac{1}{2}}}\right) &= -\nabla\hat{p}^{n+\frac{1}{2}} + \mu L_h\left(\frac{\hat{\mathbf{u}}^n+\hat{\mathbf{u}}^{n+1}}{2}\right) + \hat{\mathbf{f}}^{n+\frac{1}{2}} \\ \mathbf{D}_h^0\cdot\hat{\mathbf{u}}^{n+1} &= 0 \end{aligned}$$

Thus, the (j,k) term in the Fourier series is

$$\begin{aligned} \left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t} - \mu\frac{L_h}{2}\right)\hat{\mathbf{u}}^{n+1} &= -\nabla\hat{p}^{n+\frac{1}{2}} \\ &+ \frac{\rho^{n+\frac{1}{2}}}{\Delta t}\left(\hat{\mathbf{u}}^n + \frac{\Delta t}{\rho^{n+\frac{1}{2}}}\,\mu\frac{L_h}{2}\hat{\mathbf{u}}^n + \frac{\Delta t}{\rho^{n+\frac{1}{2}}}\,\hat{\mathbf{f}}^{n+\frac{1}{2}} - \Delta t\,\widehat{S_h(\mathbf{u}^{n+\frac{1}{2}})\mathbf{u}^{n+\frac{1}{2}}}\right) \end{aligned} \tag{2.21}$$

From $\nabla\cdot\mathbf{u} = \mathbf{0}$, we have

$$\begin{aligned} \nabla\cdot\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t} - \mu\frac{L_h}{2}\right)\hat{\mathbf{u}}^{n+1} &= -\Delta\hat{p}^{n+\frac{1}{2}} + \nabla\cdot\frac{\rho^{n+\frac{1}{2}}}{\Delta t}\left(\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\,\hat{\mathbf{f}}^{n+\frac{1}{2}} - \Delta t\widehat{S_h(\mathbf{u}^{n+\frac{1}{2}})\,\mathbf{u}^{n+\frac{1}{2}}}\right) \\ \Delta\hat{p}^{n+\frac{1}{2}} &= \frac{\rho^{n+\frac{1}{2}}}{\Delta t}\left((\widehat{rhs_u})_x + (\widehat{rhs_v})_y\right) \end{aligned} \tag{2.22}$$

Since, from (2.8), we have

$$\begin{aligned} \Delta\hat{p} &= \hat{p}_{xx} + \hat{p}_{yy} \\ \Delta\hat{p} &= -\sum_{j,k}\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2 + \left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)e^{imh_x 2\pi j + inh_y 2\pi k}\ p_{j,k} \end{aligned}$$

Then

$$\Delta\hat{p}_{j,k}^{n+\frac{1}{2}} = -\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2 + \left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right)p_{j,k}$$

And from (2.9), we have

$$\begin{aligned}(\widehat{rhs_u})_x &= i \sum_{j,k} \left(\frac{\sin(2\pi j\ dx)}{dx}\right) e^{imh_x 2\pi j + inh_y 2\pi k}\ rhs_u \\ (\widehat{rhs_v})_y &= i \sum_{j,k} \left(\frac{\sin(2\pi k\ dy)}{dy}\right) e^{imh_x 2\pi j + inh_y 2\pi k}\ rhs_v\end{aligned}$$

Thus, from (2.22) we have that

$$\begin{aligned}-\left(\left(\frac{\sin(2\pi j\ dx)}{dx}\right)^2 + \left(\frac{\sin(2\pi k\ dy)}{dy}\right)^2\right) p_{j,k} \\ = \frac{\rho^{n+\frac{1}{2}}}{\Delta t}\left(i\left(\frac{\sin(2\pi j\ dx)}{dx}\right) rhs_u + i\left(\frac{\sin(2\pi k\ dy)}{dy}\right) rhs_v\right)\end{aligned}$$

We know that

$$\hat{\mathbf{u}}^{n+1} = \begin{pmatrix}\hat{u}^{n+1} \\ \hat{v}^{n+1}\end{pmatrix} \tag{2.23}$$

Thus, from (2.21), we have $\hat{u}^{n+1}$ as following

$$\begin{aligned}\hat{u}^{n+1} &= \frac{-\nabla\hat{p} + \frac{\rho^{n+\frac{1}{2}}}{\Delta t}\widehat{rhs_u}}{\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t} - \mu\frac{L_h}{2}\right)} \\ &= \frac{\widehat{rhs_u} - i\left(\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\right)\left(\frac{\sin(2\pi j\ dx)}{dx}\right)\hat{p}^{n+\frac{1}{2}}}{1 - \mu\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\ \frac{L_h}{2}}\end{aligned} \tag{2.24}$$

Similarly, for the second component $\mathbf{u}^{n+1}$, we have

$$\begin{aligned}\hat{v}^{n+1} &= \frac{-\nabla\hat{p} + \frac{\rho^{n+\frac{1}{2}}}{\Delta t}\widehat{rhs_v}}{\left(\frac{\rho^{n+\frac{1}{2}}}{\Delta t} - \mu\frac{L_h}{2}\right)} \\ &= \frac{\widehat{rhs_v} - i\left(\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\right)\left(\frac{\sin(2\pi k\ dy)}{dy}\right)\hat{p}^{n+\frac{1}{2}}}{1 - \mu\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\ \frac{L_h}{2}}\end{aligned} \tag{2.25}$$

We now find $\frac{L_h}{2}$ in (2.24) and (2.25)
From (2.7), we have

$$\begin{aligned}\hat{\mathbf{u}}_{xx}^{n+1} &= \sum_{j,k} \frac{1}{h_x^2}\Big(-4\sin^2\Big(\frac{h_x 2\pi j}{2}\Big)\Big) e^{imh_x 2\pi j + inh_y 2\pi k}\mathbf{u}_{j,k}^{n+1} \\ &= -\sum_{j,k} e^{imh_x 2\pi j + inh_y 2\pi k}\,\frac{\sin^2\Big(\frac{h_x 2\pi j}{2}\Big)}{\Big(\frac{h_x}{2}\Big)^2}\,\mathbf{u}_{j,k}^{n+1}\end{aligned}$$

Assume $dx = \frac{hx}{2}$ and $dy = \frac{hy}{2}$, then

$$\frac{L_h}{2} = -\frac{1}{2}\Big(\frac{\sin(2\pi j\ dx)}{dx}\Big)^2 - \frac{1}{2}\Big(\frac{\sin(2\pi k\ dy)}{dy}\Big)^2$$

The denominator of (2.24) and (2.25) is equal to

$$1 - \mu\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\,\frac{L_h}{2} = 1 + 2\mu\,\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\left(\Big(\frac{\sin(2\pi j\ dx)}{dx}\Big)^2 + \Big(\frac{\sin(2\pi k\ dy)}{dy}\Big)^2\right)$$

According to equations (2.24) and (2.25), we can find $\hat{u}^{n+1}$ and $\hat{v}^{n+1}$. Then applying the inverse Fast Fourier Transform, we could obtain the updated velocity $\mathbf{u}^{n+1}$, which means moving the immersed structure at the local fluid velocities thereby enforcing no slip boundary conditions. As for the inverse Fast Fourier Transform, it could be found in Trefethen's textbook [7]. Using the updated velocity to replace the old version. Then we could calculate the next position.

2.7 Modeling the Structure Part

Let θ $(0 \le \theta \le 2\pi)$ be the parametrization of the structure, which we assume to be a closed non-self intersecting closed loop immersed in the fluid. Let $\mathbf{F}(\theta, t)$ be the force acting on the filament. Then $\mathbf{f}$, the force acting on the fluid in the Navier-Stokes equation, is given by

$$\mathbf{f}(x,t) = \int_0^{2\pi} \mathbf{F}(\theta,t)\delta\Big(x - X(\theta,t)\Big)d\theta \tag{2.26}$$

and the movement of the filament is governed by the equation

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{u}\Big(\mathbf{X}(\theta,t),t\Big) \tag{2.27}$$

$$= \int_\Omega \mathbf{u}(x,t)\ \delta\Big(\mathbf{x} - \mathbf{X}(\theta,t)\Big)d\mathbf{x} \tag{2.28}$$

where $\Omega = [0, L] \times [0, L]$ is the domain of the fluid and $\delta(\mathbf{x}) = \phi(x_1)\phi(x_2)$, where $\mathbf{x} = (x_1, x_2)$. In the above, $\phi(r)$ is continuous for all r and $\phi(r) = 0$ for $|r| \geq 2$. Define

$$\begin{aligned}
\phi(r) &= 0, \quad r \leq 2 \\
&= \frac{1}{8}\Big(5 + 2r - \sqrt{-7 - 12r - 4r^2}\Big), \quad -2 \leq r \leq -1 \\
&= \frac{1}{8}\Big(3 + 2r + \sqrt{1 - 4r - 4r^2}\Big), \quad -1 \leq r \leq 0 \\
&= \frac{1}{8}\Big(3 - 2r + \sqrt{1 + 4r - 4r^2}\Big), \quad 0 \leq r \leq 1 \\
&= \frac{1}{8}\Big(5 - 2r - \sqrt{-7 + 12r - 4r^2}\Big), \quad 1 \leq r \leq 2 \\
&= 0, \qquad r \geq 2
\end{aligned}$$

Chapter 3

Active and Passive Forces

In the section, we would like to describe the fiber models implemented in this thesis. As for the Navier-Stokes Equation, it is affected by two kinds of forces. One of the force is the internal force which includes pressure and viscosity. The other is external forces. In this chapter, all kind of forces are the external force in the fluid-structure interaction. Various fiber models give the immersed boundary certain desirable material properties relevant to many scientific applications. We should composite the total fiber model to obtain the entirely force in Navier-Stokes Equations.

In this thesis the following types of fiber models are implemented:

1. Peskin's Model
2. Spring Model
3. Torsional Spring(Beam) Model
4. Muscle-Fluid-Structure Model
5. 3-Element-Hill Model

Once the deformation energy has been calculated in the algorithm, we get

$$E(X(r,t),t) = \sum_{k=0}^{m} E_k(X_{k,1}, X_{k,2}, ..., X_{k,N_k}) \tag{3.1}$$

the corresponding elastic forces can be computed via derivatives of the elastic energy, where the elastic deformation force at point c of fiber model k is calculated as

$$F_{k,c}(X(r,t),t) = -\frac{\partial E(X(r,t),t)}{\partial X_{k,c}} \tag{3.2}$$

Note that X contains the coordinates of all immersed boundary points, M means the number of fiber structures in the system, N_k means the number of immersed boundary points in fiber structure M, and the negative sign is chosen to drive the system towards a minimal energy state. In addition, (3.1) described the compositing of the deformation energy from all respective fiber models. [12]

3.1 Peskin's Model

According to Peskin's Model[3], the force applied by arc $d\theta$ if immersed boundary to fluid. Generally, Force balance in interval (a, b) yields.
Let

$$\begin{aligned} T(\theta,t) &= \text{tension in immersed boundary} && (3.3)\\ \tau &= \frac{\partial \mathbf{X}/\partial\theta}{|\partial \mathbf{X}/\partial\theta|} = \text{unit tangent to immersed boundary} && (3.4)\\ \mathbf{F}(\theta,t)d\theta &= \text{force applied by arc } d\theta \text{ if immersed boundary to fluid} && (3.5) \end{aligned}$$

$$\begin{aligned} -\int_a^b \mathbf{F}d\theta &= \text{force of fluid on boundary} && (3.6)\\ 0 &= T\tau|_{\mathbf{a}}^{\mathbf{b}} - \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{F}\mathbf{d}\theta && (3.7)\\ &= \int_a^b \Big(\frac{\partial}{\partial\theta}(T\tau) - \mathbf{F}\Big)\mathbf{d}\theta && (3.8)\\ & && (3.9) \end{aligned}$$

Since a, b are arbitrary

$$\begin{aligned} \mathbf{F} &= \frac{\partial}{\partial\theta}(T\tau) && (3.10)\\ &= \frac{\partial T}{\partial\theta}\tau + \mathbf{T}\,\frac{\partial\tau}{\partial\theta} && (3.11)\\ &= \frac{\partial T}{\partial\theta}\tau + \mathbf{T}\,\mathbf{Cn} && (3.12) \end{aligned}$$

where C = curvature, and $\mathbf{n}$ = unit normal to boundary.
In generally, T is some function of $\left|\frac{\partial \mathbf{X}}{\partial\theta}\right|$
The special case is

$$T = K\left|\frac{\partial \mathbf{X}}{\partial\theta}\right| \quad (3.13)$$

Therefore,

$$T\tau = K\left|\frac{\partial \mathbf{X}}{\partial \theta}\right| \frac{\partial \mathbf{X}/\partial\theta}{|\partial \mathbf{X}/\partial\theta|} = K\,\frac{\partial \mathbf{X}}{\partial \theta} = K\frac{\partial \mathbf{X}}{\partial \theta} \tag{3.14}$$

So

$$F = \frac{\partial}{\partial \theta}(T\tau) = K\,\frac{\partial^2 \mathbf{X}}{\partial \theta^2} \tag{3.15}$$

3.2 Spring Model

The following model is cited the Battista's paper directly [12]. Springs is a kind of force that resistance to stretching between each consecutive Lagrangian points(nodes) can be achieved by modeling the connections with Hookean springs of resting length R_L and spring constant k_s. Thus, referring to the Hookean springs, it related to Hooke's law which is a law of physics that states that that the force needed to extend or compress a spring by some distance x cales linearly with respect to that distance. That is: $F_s = kx$, where k is a constant factor characteristic of springs: its stiffness, and x is small compared to the total possible deformation of the spring. Since we need to consider the resting the length of springs, then we obtain the energy formula as following:

$$E_{spring} = \frac{1}{2}(||X_L - X_R|| - R_L)^2 \tag{3.16}$$

where X_L and X_R are left and right node coordinates respectively. Applying for (3.2), we have the deformation force is given by a derivative on the elastic energy:

$$F_{spring} = k_x\left(1 - \frac{R_L}{||X_L - X_R||}\right) \cdot \begin{pmatrix} x_L - x_R \\ y_L - y_R \end{pmatrix} \tag{3.17}$$

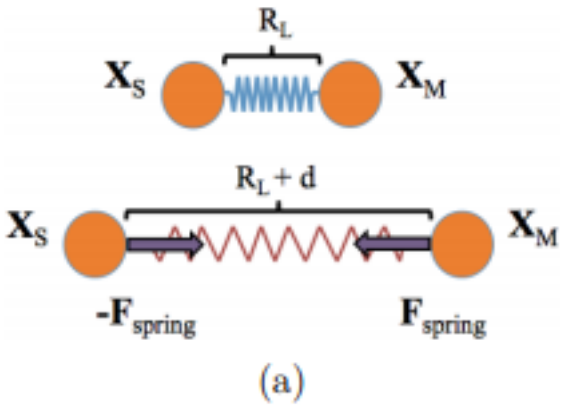
As you can see in the Figure 3.1, when the resting length is exactly equal to R_L, there is no force between two consecutive nodes. While the length greater or less than R_L, the spring provides forces for two consecutive nodes.

3.3 Torsional Spring Model

The following model is cited the Battista's paper directly [12]. Torsional springs is also called as beams, which means resistance to bending three consecutive Lagrangian points (nodes). The model assumes a desired angle θ, a prescribed 'curavture' between the three Lagrangian poins, with corresponding bending stiffness k_B. Then the bending energy is given as

$$E_{bend} = \frac{1}{2}k_B(\hat{z} \cdot (X_R - X_M) \times (X_M - X_L) - C)^2 \tag{3.18}$$

Figure 3.1: Spring Force



where X_R, X_M and X_L are right, middle and left nodes, and $C = d_{LM} d_{MR} \sin\theta$. Note that C is not the standard definition of curvature, but a curvature defined at the desired angle θ and distances between links, d_{LM} and d_{MR}.

The penalty force is designed to drive any deviations in the angle between these links back towards a lower energy state. The corresponding bending force is given by

$$F_{bend} = k_B\Big((x_R - x_M)(y_M - y_L) - (y_R - y_M)(x_M - x_L) - C\Big) \cdot \begin{pmatrix} (y_M - y_L) + (y_R - y_M) \\ -(x_R - x_M) - (x_M - x_L) \end{pmatrix} \tag{3.19}$$

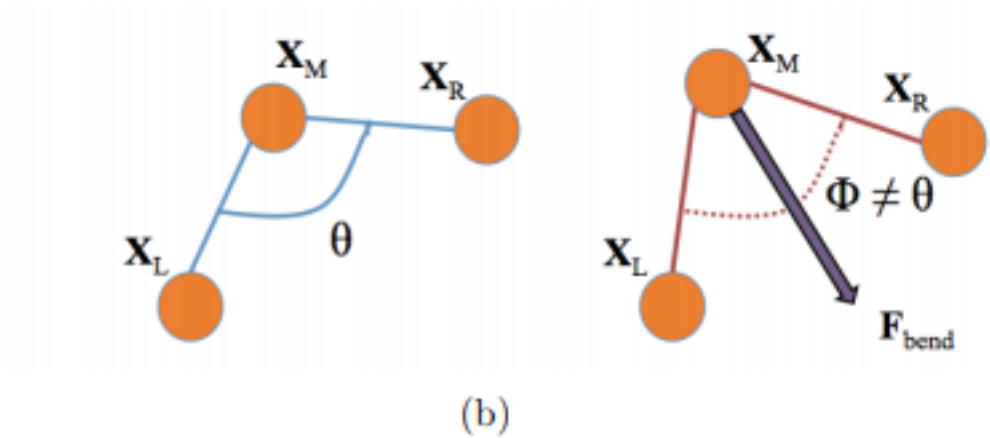
As you can see in the Figure 3.2, there is no force when the angle of the three consecutive nodes is θ. Otherwise, the torsional spring provides the force for middle point of the three consecutive nodes.

3.4 Muscle-Fluid-Structure Model

The following model is cited the Battista's paper directly [12]. According to the Model by N.A Battista [12]. For his model, it described the muscle model combined with a force-velocity and length-tension relationship in muscle without coupling in the underlying cellular processes like calcium signaling, myosin cross-bridge attachment and detachment, or filament compliance.

The muscle force is influenced by the speed of muscle contraction. Traditionally, a Hill

Figure 3.2: Torsional Spring Force



(b)

model is used to describe this relationship and takes the following form [1, 14].

$$V_F = \frac{b(F_{max} - F)}{F + a} \tag{3.20}$$

where V_F is the muscle fiber's shortening velocity, F is the force generated by the fiber, and F_{max} is the maximum load at zero contractile velocity. a and b are the parameters which are related to the internal thermodynamics of muscle.

The muscle force is also related to a function of its length. Initially when the thick filaments begin to bind to the thin filaments, the resulting force increases as the muscle shortens. However, if the muscle is contracted too far, there are fewer myosin heads to attach to the actin filaments, and the resulting force exerted is smaller. Therefore, the maximal muscle tension is located between the two extremes. A simple model of a length-tension relationship as following [4],

$$F_l = F_{l0} \exp\left[-\left(\frac{Q-1}{SK}\right)^2 \right] \tag{3.21}$$

where $Q = \frac{L_F}{L_{F0}}$ is the ratio of the length of the muscle fibers to the length when they reach the maximum tension, F_l is the maximum isometric tension at a given fiber length L_F, F_{l0} is the maximum isometric force when they reach the optimum length of the muscle fibers, and SK is a parameter specific for each muscle.

Follow by Battista's model, combining (3.20) and (3.21) is to take product of their normalized versions, as in [12, 5]. The model as following

$$F_{muscle}(L_F, V_F) = a_f F_{max} F_1(L_F) F_2(V_F) \tag{3.22}$$

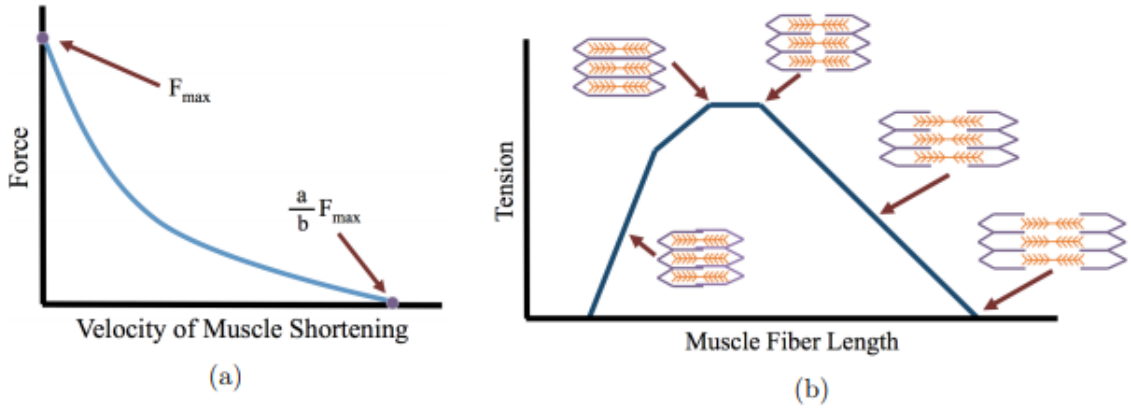
where a_f is the activation strength of the muscle related to the Calcium and F_{max} is the normalized maximum isometric force at the full activation of the muscle fibers at their optimum lengths, and $F_1(L_F)$ and $F_2(V_F)$ are normalized versions of (3.20) and (3.21), given by

$$F_1(L_F) = \exp\left[-\left(\frac{L_F/L_{F0}-1}{SK}\right)^2\right] \tag{3.23}$$

$$F_2(V_F) = \frac{1}{F_{max}}\left[\frac{bF_{max}-aV_F}{b+V_F}\right] \tag{3.24}$$

Since a_f is the activation strength of the muscle related to the Calcium Dynamics, we used the Shannon Model as the activation strength and re-scale the range from 0 to 1 [13].

Figure 3.3: Force-velocity and length-tension curves illustrating the respectively



As you can see in the Figure 3.3, the F_2 decreases with the increasing of velocity and F_muscle related to the actin and myosin cross-bridges at the sarcomere level.

3.5 3-Element-Hill Model

The following model is cited the Battista's paper directly [12]. The 3-Element Hill model activation describes sustained muscle contraction by modeling the actin and myosin cross-bridges, muscle tendon, and connective tissues for a muscle. The model has a contractile element which models the force generated by the actin and myosin cross-bridges at the sarcomere level, and two non-linear spring elements, and one of them is parellel element

and the other one is the series element with contractile element. The series element models related to the muscle tendon, and has a soft tissue response and provides energy storing mechanism. The parallel element is responsible for the passive behavior when the muscle is stretched. For the contractile element, it is the same model we described in the above section 3.4.

Let F_{CE}, F_{SE} and F_{PE} be contractile, series, and parallel elements respectively. In mechanics, two or more springs share the same force in series if they are connected end-to-end. If two or more springs in parallel which means they are connected side-by-side, the force of this case is summation of the force of each side. Thus, we have the formula as following

$$F_{tot} = F_{SE} + F_{PE} \tag{3.25}$$
$$F_{CE} = F_{SE} \tag{3.26}$$

where F_{tot} is the total force produced by muscle contraction. For the muscle shortening, we have

$$L_{tot} = L_{CE} + L_{SE} = L_{PE} \tag{3.27}$$

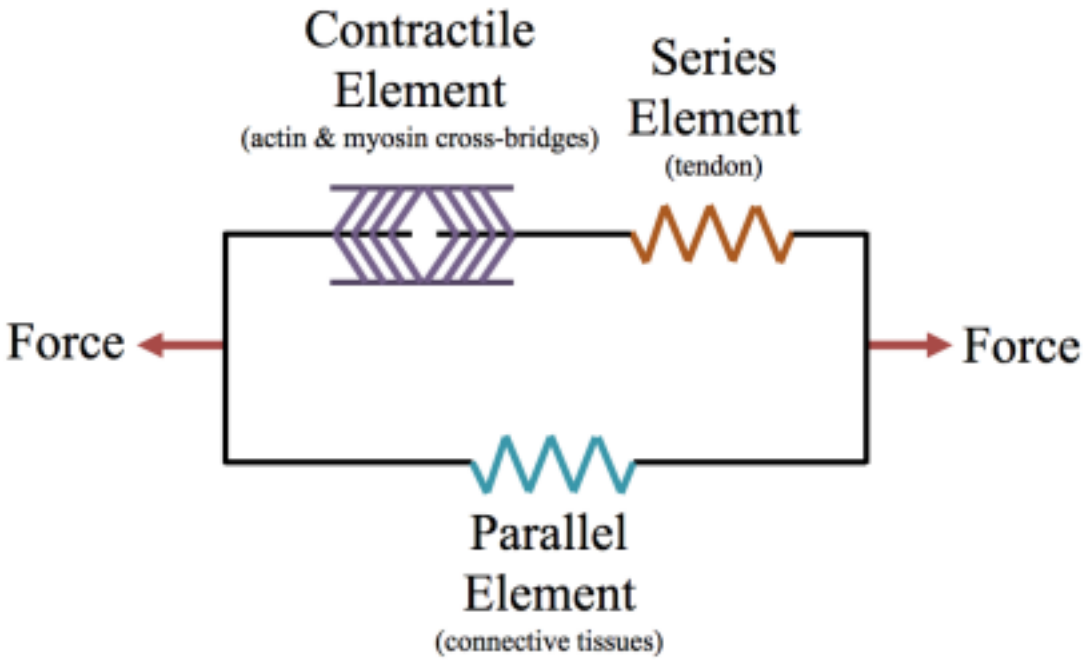
where L_{tot} is the total length of the muscle.

The contractile element is similar with the above section. Thus, we can assume it related to the length-tension and force-velocity relationship of muscle. By equation (3.22), we have

$$F_{CE} = a_f F_{max} F_1(L_{CE}) F_2(V_{CE}) \tag{3.28}$$

where L_{CE} is the length of the muscle fibers and V_{CE} is the contraction speed of the muscle fibers.
From Figure 3.4, this is the picture of the three-element Hill model.

Figure 3.4: 3-Element Hill Model



Chapter 4

Numerical Simulation

In this chapter, we modified the model which comes from Charles S. Peskin [2]. This thesis shows some numerical simulations with tables and results of three cases of different combinations of the following 4 models.

1. Original Peskin's Model
2. Spring Model
3. Torsional Spring Model
4. 3-Element-Hills Model

The combination of the 4 models is based on

$$F = \omega_0 \, F_{peskin} + \omega_1 \, F_{spring} + \omega_2 \, F_{torsionalspring} + \omega_3 \, F_{muscle} \tag{4.1}$$

where ω_i is the weight of each force. The initial velocity $\mathbf{u}$ is set to zero. The initial geometry of the strucutre is a circle centered at $(\frac{L}{2}, \frac{L}{2})$ with radius $\frac{L}{4}$. $[0, L] \times [0, L]$ is the size of the domain.

Firstly, we provide the proper parameters in the following table.

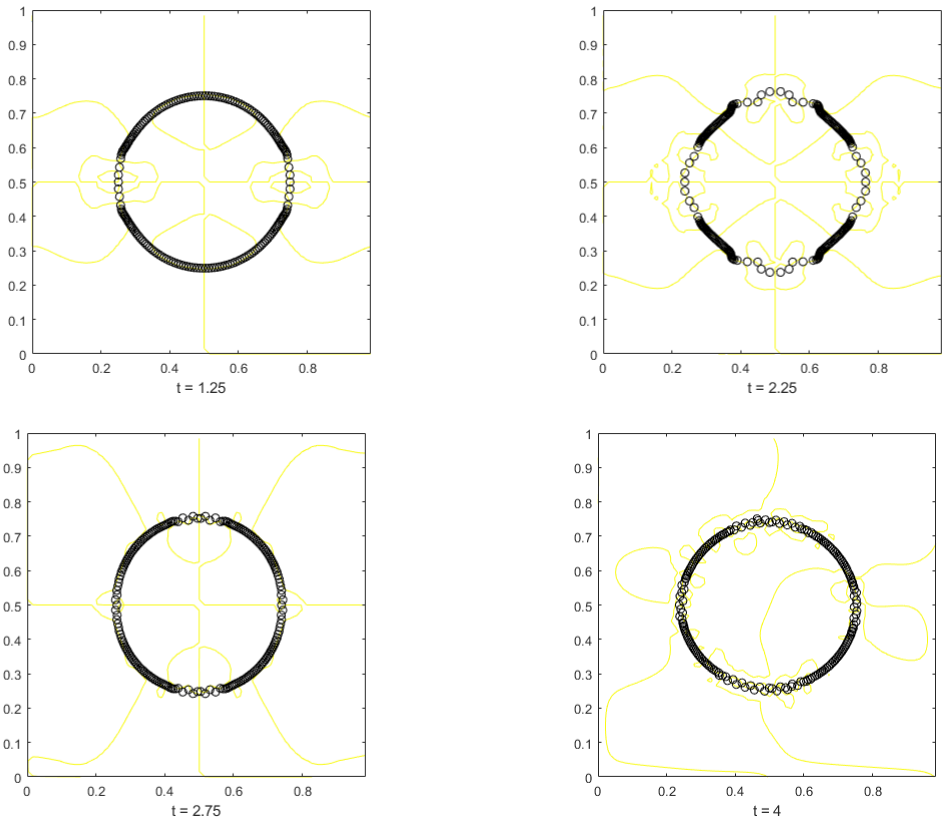
4.1 Parameters

Parameter Values and Their Units			
Measurement	Symbol	Magnitude	Unit
Length	L	1.0	mm
Number of Grid	N	64	Each
Length of each Grid	h	$\frac{1}{64}$	mm
Number of Points	Nb	202	Each
Density	ρ	1.06	kg/m^2
Dynamic Viscosity	μ	0.04	$N * s/m^2$
Time-step	dt	0.01	second
Degree	α	1	No unit
x-Direction Length	Lx	1.0	mm
y-Direction Length	Ly	1.0	mm
Rest of Length	Lr	0.015	mm
Stiffness	K_{spring}	300	N/mm
Degree	α	1	No unit
x-Direction Length	Lx	1.0	mm
y-Direction Length	Ly	1.0	mm
Rest of Length	C	0	mm
Stiffness	K_{beam}	150	N/mm
Degree	α	1	No Unit
x-Direction Length	Lx	1.0	mm
y-Direction Length	Ly	1.0	mm
Stiffness	K_{beam}	5	N/m
Maximum Tension	F_{max}	212	KPa
Hill parameter a	a	0.25	No Unit
Hill parameter b	b	4	No Unit
Muscle constant	Sk	0.3	No Unit
Length for max muscle tension	LF0	0.018	mm

4.2 Case 1

In the first case, we show the simulation which only contains Original Peskin's model and 3-Element Hill model. Thus, suppose $\omega_0 = 1$ and $\omega_3 = 1$, and set ω_1 and ω_2 to 0,

Figure 4.1: Case 1: $\omega_0 = 1, \omega_1 = 0, \omega_2 = 0$ and $\omega_3 = 1$.

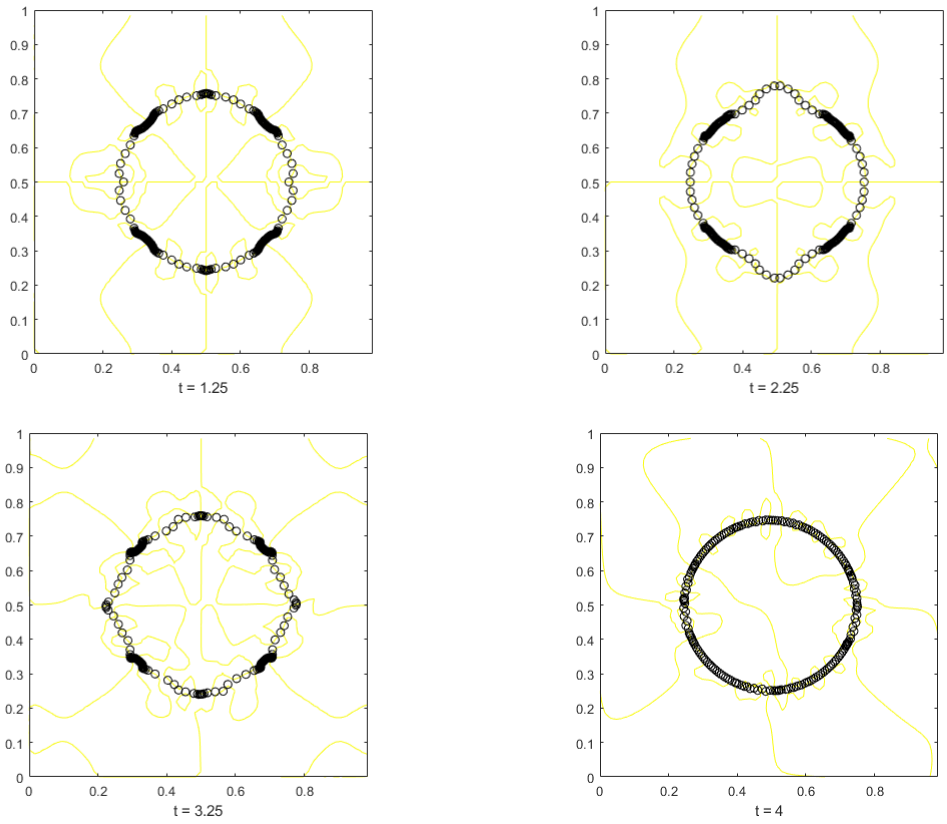


The above Figure (4.1) show the simulation at $t = 1.25, 2.25, 2.75$, and 4 respectively. Since this case does not contain the spring and torsional spring model. The nodes apparently converges to the diagonal at $t = 4$. We will add spring and torsional spring model into the second case, and observe the changes.

4.3 Case 2

In this case, we keep the Original Peskin's model and 3-Element Hill model, and add spring and torsional spring model into the case. Thus, let $\omega_i = 1$, for $i = 0, 1, 2, 3$.

Figure 4.2: Case 2: $\omega_0 = 1, \omega_1 = 1, \omega_2 = 1$ and $\omega_3 = 1$.

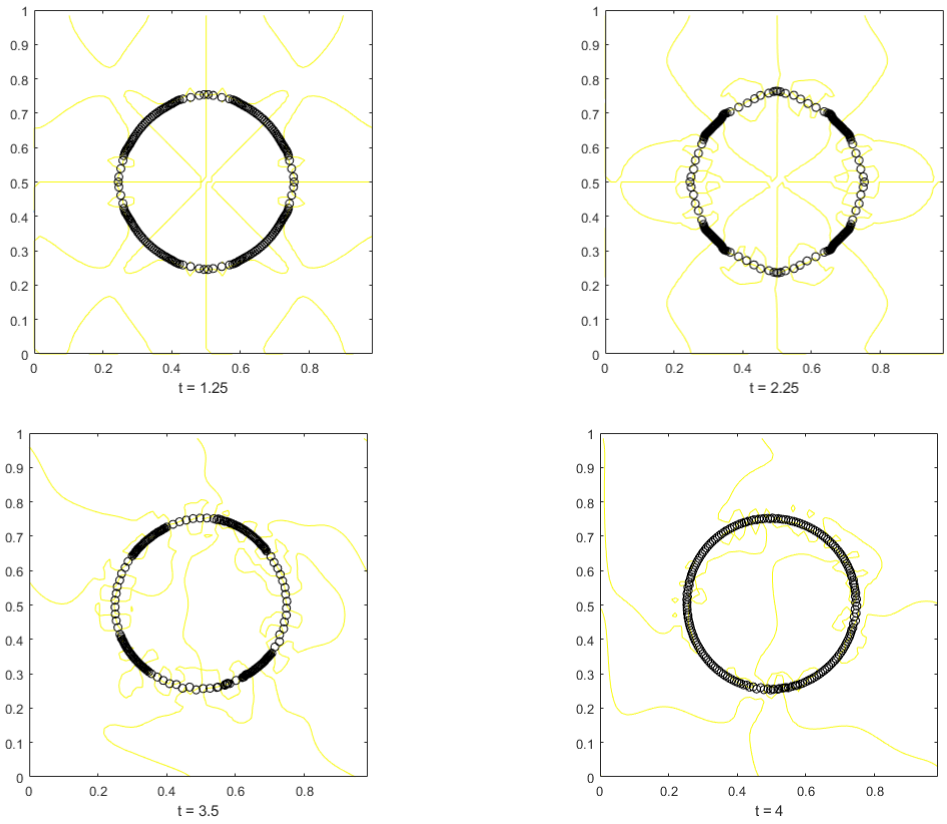


The above Figure (4.2) show the simulation at $t = 1.25, 2.25, 3.25$, and 4 respectively. A comparison of Figure (4.2) and Figure (4.1), it is obvious that the force is more evenly spread between the nodes in Case 2. However, the shape of the structure is not convex enough. Since the original Peskin's model provides the curvature force to the shape, we enlarge the weight of the original Peskin's model in the next case.

4.4 Case 3

In this case, we keep the same weight of spring, torsional spring and 3-Element Hill model as Case 2, and increase the weight of the Original Peskin's model. Let $\omega_0 = 1.5$

Figure 4.3: Case 3: $\omega_0 = 1.5, \omega_1 = 1, \omega_2 = 1$ and $\omega_3 = 1$.



The above Figure (4.3) show the simulation at $t = 1.25, 2.25, 3.5$, and 4 respectively. According to Figure (4.1),Figure (4.2) and Figure (4.3), after increasing of weight of the original Peskin's model, the structure begins doing the periodic movement and the shape and force magnitude of each time step is stable.

Chapter 5

Conclusion

In this thesis, we modified the numerical results of Charles Peskin in [2] to simulate the active and passive forces in the Fluid-Structure Interaction problem of the human heart. In addition to Peskin model, we implemented three other models including the spring model, the torsional spring (or beam) model, and the 3-element Hill model into our simulations. Using Fast Fourier Transform (FFT), we solved the Navier-Stokes equation. FFT allows computational time to be $O(n\,log\,n)$ instead of $O(n^2)$. For numerical implementations, the domain is taken to be $[0,1] \times [0,1]$ and we used the parameter values from Battista's paper [12]. In Chapter 1, we introduced the left ventricle and provided background of the Navier-Stokes equations. In Chapter 2, we described the half-step method to solve the Navier-Stokes equation, and explained the Fast Fourier Transform method [7]. At the end of Chapter 2, we illustrated how to update the velocity $\mathbf{u}(x,t)$ in the Fluid-Structure Interaction Problem. In Chapter 3, we explained details of the above mentioned four models which come from Peskin's Note and Battista's paper [2, 12]. Chapter 4 is numerical results. We implemented three combinations of the model mentioned in Chapter 3. Since to simulate the movement of the left ventricle, we added the Calcium Dynamics from the Shannon model [13]. The best result was obtained if we weighted Peskin model 1.5 times the other three models. Because of the lack of time, we did not find the optimal combination of these four forces to produce simulation that best resembles a beating heart.

Bibliography

[1] A. V. Hill, (1938) The heat of shortening and the dynamic constants of muscle, Proc. R. Soc. Lond. 126: 136–195.

[2] C. Peskin, (2007), Lecture Note, www.math.nyu.edu/faculty/peskin/ib_lecture_notes/index.html

[3] C. Peskin, (2002). The immersed boundary method. Acta Numerica, 11, 479-517. doi:10.1017/S0962492902000077

[4] H. Hatze, (1981) A comprehensive model for human motion simulation and its application to the take-off phase of the long jump, J. Biomech. 14: 135–142.

[5] J. H. Challis, D. G. Kerwin, (1994) Determining individual muscle forces during maximal activity: Model development, parameter determination, and validation, Hum. Movement Sci. 13: 2961.

[6] Jia Yan-Bin , Curvature, http://web.cs.iastate.edu/ cs577/handouts/curvature.pdf

[7] Lloyd N. Trefethen, (2000) Spectral Methods in MATLAB, SIAM, Philadelphia

[8] Lai and C. S. Peskin (2000), An immersed boundary method with formal second order accuracy and reduced numerical viscosity. J. Comput. Phys. 160: 705–719.

[9] McQueen and C. S. Peskin (1983), Computer-assisted design of pivoting-disc prosthetic mitral valves. J. Thorac. Cardiovasc. Surg. 86:126–135.

[10] Jiyuan Tu,Guan-Heng Yeoh and Chaoqun Liu, (2013) Computational Fluid Dynamics :A Practical Approach 2nd edition, https://doi.org/10.1016/C2010-0-67980-6

[11] Jiyuan Tu,Guan-Heng Yeoh and Chaoqun Liu, (2018) Computational Fluid Dynamics :A Practical Approach 3rd edition, https://doi.org/10.1016/C2015-0-06135-4

[12] N. A. Battista, A. J. Baird, L. A. Miller, (2015) A mathematical model and matlab code for muscle-fluidstructure simulations, Integr. Comp. Biol. DOI: 10.1093/ICB/ICV102

[13] Thomas R. Shannon, Fei Wang, José Puglisi, Christopher Weber, and Donald M. Bers,(2004) A Mathematical Treatment of Integrated Ca Dynamics within the Ventricular Myocyte, Biophys J.; 87(5): 3351–3371. Published online 2004 Sep 3. doi: 10.1529/biophysj.104.047449

[14] Y. C. Fung, (1993) *Biomechanics: mechanical properties of living tissues*, Springer-Verlag, New York, USA.

Appendix A

Matlab Code

Code for all Matlab functions used in simulations is listed below.

```
% ib2D.m
% This script is the main program.
clear all
close all

global dt Nb N h rho mu ip im a;
global kp km dtheta K LF0 dt RL;
initialize
init_a

x_master = X(kp,1);
x_slave = X(km,1);
y_master = X(kp,2);
y_slave = X(km,2);

for i = 1:202
    dx = x_master(i) - x_slave(i);
    dy = y_master(i) - y_slave(i);
    RL(i) = sqrt( dx^2 + dy^2 );
end

for i = 1:202
    dx = x_master(i) - x_slave(i);
    dy = y_master(i) - y_slave(i);
    LF0(i) = 0.018;
end

tmp = load('shannonresult');
ti = tmp(:,1);  % time
Cai = tmp(:,2);   % calcium
X_P = X;

```

```
33
34 for clock=1:clockmax
35   XX=X+(dt/2)*interp(u,X);
36   af = Cai(clock*10);
37   ff=spread(Force(XX,X_P, af),XX);
38   [u,uu]=fluid(u,ff);
39   X_PP = X_P;
40   X_P = X;
41   X=X+dt*interp(uu,XX);
42
43   %animation:
44   vorticity=(u(ip,:,2)-u(im,:,2)-u(:,ip,1)+u(:,im,1))/(2*h);
45   contour(xgrid,ygrid,vorticity,values)
46   hold on
47   plot(X(:,1),X(:,2),'ko')
48   axis([0,L,0,L])
49   caxis(valminmax)
50
51   axis equal
52   axis manual
53   drawnow
54   hold off
55 end
```

```
%initialize.m
L=1.0
N=64
h=L/N
ip=[(2:N),1]
im=[N,(1:(N-1))]
Nb=ceil(pi*(L/2)/(h/2))
dtheta=2*pi/Nb
kp=[(2:Nb),1]
km=[Nb,(1:(Nb-1))]
K=1
rho=1.06
mu=0.04
tmax=4;
dt=0.01
clockmax=ceil(tmax/dt)

for k=0:(Nb-1)
  theta=k*dtheta;
  X(k+1,1)=(L/2)+(L/4)*cos(theta);
  X(k+1,2)=(L/2)+(L/4)*sin(theta);
end

u=zeros(N,N,2);
for j1=0:(N-1)
  x=j1*h;
  u(j1+1,:,2)=eps*cos(2*pi*x/L);
end

vorticity=(u(ip,:,2)-u(im,:,2)-u(:,ip,1)+u(:,im,1))/(2*h);
dvorticity=(max(max(vorticity))-min(min(vorticity)))/5;
values= (-10*dvorticity):dvorticity:(10*dvorticity);
valminmax=[min(values),max(values)];
xgrid=zeros(N,N);
ygrid=zeros(N,N);
for j=0:(N-1)
  xgrid(j+1,:)=j*h;
  ygrid(:,j+1)=j*h;
end

set(gcf,'double','on')
contour(xgrid,ygrid,vorticity,values)
hold on
plot(X(:,1),X(:,2),'ko')
axis([0,L,0,L])
caxis(valminmax)
axis equal
axis manual
drawnow
hold off
```

```
function F=Force(X,X_P,af)
global kp km dtheta K LF0 dt Nb RL;



F=K*(X(kp,:)+X(km,:)-2*X)/(dtheta*dtheta);

%F_peskin = mean(F)   % ? 20191201-WL;

k_Spring = 300; % 300;
% F_spring1 = 1/2*k_Spring*(1 - (X(kp,:)+X(km,:))/norm(X(kp,:)-X(km,:)));
% F_spring2 = X(kp,:)-X(km,:);
% F_spring = F_spring1.*F_spring2;
%
% F = F + F_spring;

kk=1:Nb;
x_master = X(kp,1);
x_slave = X(kk,1);
y_master = X(kp,2);
y_slave = X(kk,2);
% Lx = 1;       % 20191201-WL;
% Ly = 1;
% L_r = 2*pi*Lx/4/202;
alpha = 1;

fx = zeros(Nb,1);                % Initialize storage for x-forces
fy = fx;                         % Initialize storage for y-forces

for i = 1:Nb
dx = x_master(i) - x_slave(i);
dy = y_master(i) - y_slave(i);

%     if abs(dx) > Lx/2                      % 20191201-WL;
%         dx = sign(dx)*( Lx - sign(dx)*dx );
%     end
%
%     if abs(dy) > Ly/2
%         dy = sign(dy)*( Ly - sign(dy)*dy );
%     end

sF_x = 0.5*(alpha+1) * k_Spring * ( sqrt( dx^2 + dy^2 ) - RL(i) )^(alpha) * (
    dx / sqrt(dx^2+dy^2) );
sF_y = 0.5*(alpha+1) * k_Spring * ( sqrt( dx^2 + dy^2 ) - RL(i) )^(alpha) * (
    dy / sqrt(dx^2+dy^2) );

% THE FORCE IS THE ONE APPLIED ON SPRING TO HAVE THIS DEFORMATION; 20191201-WL
    ;
 fx(kp(i),1) = fx(kp(i),1) + sF_x;    % Sum total forces for node, i in x-
    direction (this is MASTER node for this spring)
 fy(kp(i),1) = fy(kp(i),1) + sF_y;    % Sum total forces for node, i in y-
```

```
    direction (this is MASTER node for this spring)
48
49  fx(kk(i),1) = fx(kk(i),1) - sF_x;    % Sum total forces for node, i in x-
        direction (this is SLAVE node for this spring)
50  fy(kk(i),1) = fy(kk(i),1) - sF_y;    % Sum total forces for node, i in y-
        direction (this is SLAVE node for this spring)
51
52  end
53
54  F_spring = [fx fy];
55
56  % F_spring_force = mean(F_spring)
57
58
59  %new, beam
60  kk=1:Nb;
61  k_Beam = 150;
62  % C = 0;
63  % C=LF0(1)*LF0(2)*sin(dtheta);
64
65  fx = zeros(Nb,1);                  % Initialize storage for x-forces
66  fy = fx;                           % Initialize storage for y-forces
67
68
69  x_left = X(km,1);
70  x_middle = X(kk,1);
71  x_right = X(kp,1);
72
73  y_left = X(km,2);
74  y_middle = X(kk,2);
75  y_right = X(kp,2);
76
77  for i=1:Nb
78      X_L = x_left(i);           % xPt of 1ST Node Pt. in beam
79      X_M = x_middle(i);           % xPt of 2ND (MIDDLE) Node Pt. in beam
80      X_R = x_right(i);          % xPt of 3RD Node Pt. in beam
81
82      Y_L = y_left(i);           % yPt of 1ST Node Pt. in beam
83      Y_M = y_middle(i);           % yPt of 2ND (MIDDLE) Node Pt. in beam
84      Y_R = y_right(i);          % yPt of 3RD Node Pt. in beam
85
86          % Compute Cross-Product
87      cross_prod = (X_R-X_M)*(Y_M-Y_L) - (Y_R-Y_M)*(X_M-X_L);
88
89      C= sqrt( (X_M - X_L)^2 + (Y_M - Y_L)^2 )* sqrt( (X_M - X_R)^2 + (Y_M - Y_R
            )^2 )*sin(dtheta);
90
91      % FORCES FOR LEFT NODE
92      bF_x_L = -k_Beam * ( cross_prod - C ) * ( Y_R-Y_M );
93      bF_y_L =  k_Beam * ( cross_prod - C ) * ( X_R-X_M );
94
```

```
95      % FORCES FOR MIDDLE NODE
96      bF_x_M =  k_Beam * ( cross_prod - C ) * (  (Y_M-Y_L) + (Y_R-Y_M) );
97      bF_y_M = -k_Beam * ( cross_prod - C ) * (  (X_R-X_M) + (X_M-X_L) );
98
99      % FORCES FOR RIGHT NODE
100     bF_x_R = -k_Beam * ( cross_prod - C ) * ( Y_M-Y_L );
101     bF_y_R =  k_Beam * ( cross_prod - C ) * ( X_M-X_L );
102
103     fx(km(i),1) = fx(km(i),1,1) - bF_x_L;  % Sum total forces for left node,
           in x-direction (this is LEFT node for this beam)
104     fy(km(i),1) = fy(km(i),1,1) - bF_y_L;  % Sum total forces for left node,
           in y-direction (this is LEFT node for this beam)
105
106     fx(kk(i),1) = fx(kk(i),1,1) + bF_x_M;  % Sum total forces for middle node,
            in x-direction (this is MIDDLE node for this beam)
107     fy(kk(i),1) = fy(kk(i),1,1) + bF_y_M;  % Sum total forces for middle node,
            in y-direction (this is MIDDLE node for this beam)
108
109     fx(kp(i),1) = fx(kp(i),1,1) - bF_x_R;  % Sum total forces for right node,
           in x-direction (this is RIGHT node for this beam)
110     fy(kp(i),1) = fy(kp(i),1,1) - bF_y_R;  % Sum total forces for right node,
           in y-direction (this is RIGHT node for this beam)
111
112 end
113
114 F_beam = [fx fy];
115
116 % F_beam_force = mean(F_beam)
117
118
119 %Muscle Mode
120 FMax = 212;
121 SK=0.3;
122 a=0.25;
123 b=4;
124 x_master_P = X_P(kp,1);  % x_master_P = X_P(kp,1);
125 x_slave_P = X_P(kk,1);   % x_slave_P = X_P(km,1);
126 y_master_P = X_P(kp,2);  % y_master_P = X_P(kp,2);
127 y_slave_P = X_P(kk,2);   % y_slave_P = X_P(km,2);
128 kSpr = 5;
129
130 fx = zeros(Nb,1);                % Initialize storage for x-forces
131 fy = fx;                         % Initialize storage for y-forces
132
133 for i= 1:Nb
134     dx = x_master(i) - x_slave(i);
135     dy = y_master(i) - y_slave(i);
136     LF = sqrt( dx^2 + dy^2 );
137     Q=LF/LF0(i);
138     F1 = exp( -( (Q-1)/SK )^2 );
139     P0 = FMax*F1;
```

```
140     dx_P = x_master_P(i) - x_slave_P(i);
141     dy_P = y_master_P(i) - y_slave_P(i);
142     LF_P = sqrt( dx_P^2 + dy_P^2 );
143     v = abs(LF-LF_P)/(dt/2);
144     F2 = (1/P0)*(b*P0-a*v)/(v+b);
145     Fm = af*FMax*abs(F1*F2);
146     mF_x = Fm*(dx/LF);
147     mF_y = Fm*(dy/LF);
148
149
150     % PE force
151     sF_PE_x = kSpr * ( sqrt( dx^2 + dy^2 ) - LF0(i) ) * ( dx / sqrt(dx^2+dy^2)
            );
152     sF_PE_y = kSpr * ( sqrt( dx^2 + dy^2 ) - LF0(i) ) * ( dy / sqrt(dx^2+dy^2)
            );
153
154
155     fx(kp(i),1) = fx(kp(i),1) + mF_x+sF_PE_x;
156     fy(kp(i),1) = fy(kp(i),1) + mF_y+sF_PE_y;
157
158     fx(kk(i),1) = fx(kk(i),1) - mF_x - sF_PE_x;
159     fy(kk(i),1) = fy(kk(i),1) - mF_y - sF_PE_y;
160
161 end
162
163
164 F_Muscle = [fx fy];
165
166 % F_Muscle_force = mean(F_Muscle)
167
168
169 % %Curvature Force, LX is wrong;
170 % k_curve = 1e-08;
171 % x_deri =  (X(kp,1) - X(kk,1))/(dtheta);            %first deri for x-
       direction
172 % y_deri =  (X(kp,2) - X(kk,2))/(dtheta);            %first deri for y-
       direction
173 % % x_deri =  (X(kk,1) - X_P(kk,1))/(dtheta);            %first deri for x-
       direction
174 % % y_deri =  (X(kk,2) - X_P(kk,2))/(dtheta);            %first deri for y-
       direction
175 %
176 % x_second_deri =  (X(kp,1) - 2*X(kk,1) + X(km,1))/(dtheta^2);
       %second deri for x-direction
177 % y_second_deri =  (X(kp,2) - 2*X(kk,2) + X(km,2))/(dtheta^2);
       %second deri for y-direction
178 % % x_second_deri =  (X(kk,1) - 2*X_P(kk,1) + X_PP(kk,1))/((dtheta)^2);
                    %second deri for x-direction
179 % % y_second_deri =  (X(kk,2) - 2*X_P(kk,2) + X_PP(kk,2))/((dtheta)^2);
                    %second deri for y-direction
180 %
```

```
% fx = zeros(Nb,1);                  % Initialize storage for x-forces
% fy = fx;                            % Initialize storage for y-forces
%
% for i = 1:Nb
%     k_curvature = (x_deri(i)*y_second_deri(i) - x_second_deri(i)*y_deri(i))
   /((x_deri(i)^2 + y_deri(i)^2)^(3/2));
%     N_x = -y_deri(i)/(x_deri(i)^2 + y_deri(i)^2)^0.5;
%     N_y =  x_deri(i)/(x_deri(i)^2 + y_deri(i)^2)^0.5;
%     fx(kk(i),1) = k_curve*k_curvature*N_x;
%     fy(kk(i),1) = k_curve*k_curvature*N_y;
%     if isnan(fx(kk(i),1))
%         fx(kk(i),1)  = 0;
%     end
%     if  isnan(fy(kk(i),1))
%         fy(kk(i),1)  = 0;
%     end
% end
%
% F_curvature = [fx fy];

% F_cur_force = mean(F_curvature)


F = F + F_Muscle; % F + F_spring + F_beam +  + F_curvature;
%F = 1.5*F + F_spring + F_beam + F_Muscle;
```