IMMERSED BOUNDARY SIMULATIONS AND TOOLS FOR STUDYING INSECT FLIGHT AND OTHER APPLICATIONS

D. Michael Senter

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2021

Approved by:

Laura A. Miller

M. Gregory Forest

Boyce E. Griffith

Katherine A. Newhall

Brian K. Taylor

ABSTRACT

D. Michael Senter: Immersed boundary simulations and tools for studying insect flight and other applications
(Under the direction of Laura A. Miller)

All organisms must deal with fluid transport and interaction, whether it be internal, such as lungs moving air for the extraction of oxygen, or external, such as the expansion and contraction of a jellyfish bell for locomotion. Most organisms are highly deformable and their elastic deformations can be used to move fluid, move through fluid, and resist fluid forces. A particularly effective numerical method for biological fluid-structure interaction simulations is the immersed boundary (IB) method. An important feature of this method is that the fluid is discretized separately from the boundary interface, meaning that the two meshes do not need to conform with each other. This thesis covers the development of a new software tool for the semi-automated creation of finite difference meshes of complex 2D geometries for use with immersed boundary solvers IB2d and IBAMR, alongside two examples of locomotion - the flight of tiny insects and the metachronal paddling of brine shrimp.

As mentioned, an advantage of the IB method is that complex geometries, e.g., internal or external morphology, can easily be handled without the need to generate matching grids for both the fluid and the structure. Consequently, the difficulty of modeling the structure lies often in discretizing the boundary of the complex geometry (morphology). Both commercial and open source mesh generators for finite element methods have long been established; however, the traditional immersed boundary method is based on a finite difference discretization of the structure. In chapter 2, I present a software library called MeshmerizeMe for obtaining finite difference discretizations of boundaries for direct use in the 2D immersed boundary method. This library provides tools for extracting such boundaries as discrete mesh points from digital images. Several examples of how the method can be applied are given to demonstrate the effectiveness of the software, including passing flow through the veins of insect wings, within lymphatic capillaries, and around starfish using open-source immersed boundary software.

As an example of insect flight, I present a 3D model of clap and fling. Of the smallest insects filmed in flight, most if not all clap their wings together at the end of the upstroke and fling them apart at the beginning of the downstroke. This motion increases the strength of the leading edge vortices generated during the downstroke and augments the lift. At the Reynolds numbers (Re) relevant to flight in these insects (roughly $4 < Re < 40$), the drag produced during the fling is substantial, although this can be reduced through the presence of wing bristles, chordwise wing flexibility, and more complex wingbeat kinematics. It is not clear how flexibility in the spanwise direction of the wings can alter the lift and drag generated. In chapter 3, a hybrid version of the immersed boundary method with finite elements is used to simulate a 3D idealized clap and fling motion across a range of wing flexibilities. I find that spanwise flexibility, in addition to three-dimensional spanwise flow, can reduce the drag forces produced during the fling while maintaining lift, especially at lower Re. While the drag required to fling 2D wings apart may be more than an order of magnitude higher than the force required to translate the wings, this effect is significantly reduced in 3D. Similar to previous studies, dimensionless drag increases dramatically for $Re < 20$, and only moderate increases in lift are observed. Both lift and drag decrease with increasing wing flexibility, but below some threshold, lift decreases much faster. This study highlights the importance of flexibility in both the chordwise and spanwise directions for low Re insect flight. The results also suggest that there is a large aerodynamic cost if insect wings are too flexible.

My second application of locomotion pertains to a 2D model of swimming, specifically the method known as metachronal paddling. This method is used by a variety of organisms to propel themselves through a fluid. This mode of swimming is characterized by an array of appendages that beat out of phase, such as the swimmerets used by long-tailed crustaceans like crayfish and lobster. This form of locomotion is typically observed over a range of Reynolds numbers greater than 1 where the flow is dominated by inertia. The majority of experimental, modeling, and numerical work on metachronal paddling has been conducted on the higher Reynolds number regime (order 100). In this chapter, a simplified numerical model of one of the smaller metachronal swimmers, the brine shrimp, is constructed. Brine shrimp are particularly interesting since they swim at Reynolds numbers on the order of 10 and sprout additional paddling appendages as they grow. The immersed boundary method is used to numerically solve the fluid-structure interaction problem of multiple rigid paddles undergoing cycles of power and return strokes with a constant phase difference and spacing that are

based on brine shrimp parameters. Using a phase difference of 8%, the volumetric flux and efficiency per paddle as a function of the Reynolds number and the spacing between legs is quantified. I find that the time to reach periodic steady state for adult brine shrimp is large (≈ 150 stroke cycles) and decreases with decreasing Reynolds number. Both efficiency and average flux increase with Reynolds number. In terms of leg spacing, the average flux decreases with increased spacing while the efficiency is maximized for intermediate leg spacing.

To my family

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor Laura Miller. She has been an incredible mentor with never ending patience and a wellspring of knowledge. She has created an atmosphere in her lab where I was able to feel welcome and included from my very first weeks at UNC. Aside from her invaluable advice and guidance on math, biology, and science in general, she has always made it clear that she cares about each of her students as a person as well as a scientist. Her leadership style fosters an environment where we can not only be colleagues, but friends. I am grateful to have been part of the Miller lab, and hope to continue as an 'affiliate' into the future, as my predecessors have done.

At this point I would like to thank all of my friends in the Miller lab, current, former, and affiliated, in alphabetical order: Nick Battista, Kirsten Giesbrecht, Christy Hamlet, Alex Hoover, Shannon Jones, Grace McLaughlin, Kemal Ozalp, Virginia Pasour, Julia Samson, Matt Snyder, and Christopher Strickland. Each of you have been an important part of my journey, and I am glad to have been privileged to work with you all on various endeavors. I also would like to thank my undergraduate research assistant, Kristen Armel, for all of her hard work over the years.

I am grateful for all of my committee members and their contributions to my graduate career. I would like to thank Greg Forest for always having an open ear and encouraging me in all of my endeavors, Boyce Griffith and his lab for their incredible IBAMR software and expertise, Katie Newhall for helping me gain solid footing, particularly in my early graduate career, and Brian Taylor for his sense of humor and career advice.

Even though they are not on my committee, I would also like to extend heartful thanks to Sorin Mitran for the hours we have spent in his office discussing numerical methods and asymptotics, and to Bill Kier for fruitful discussions on biomechanics. I would also like to thank George Fishman, Nadav Dym, and Steven Thomas for always letting me bounce ideas off of them at any opportunity. And thank you also to the many graduate students in the math department for uplifting mathematical and sometimes decidedly non-mathematical chatter at tea times, including (but not limited to) Brian

Adam, Marc Besson, Wyatt Bridgman, Gracie Conte, Andrew Ford, Sam Jeralds, Blake Keeler, Jason Pearson, Carold Sadek, Brittany Shepherd and Ben Vadala-Roth.

I have been fortunate to work with an amazing team at my SAS internship. They have broadened my horizons and made the last year of my program more enjoyable. Thank you Ariel Chien, Lincoln Groves, James Harroun, Jackie Johnson, Rachel Nisbet, and Lynn Letukas for this experience.

Of course I would not have made it this far without the many people who have helped my educational career before I ever came to UNC. Thank you Bernhard Grosser for not giving up on me despite my rather dreadful translation skills in ancient Greek, and thank you Joachim Ritter for encouraging me to reach for the stars. Thank you Matthias Gröschel for making Gymnasium fun and the great conversations we've had about life and our scientific adventures. I would like to particularly thank Matthias Kick for always beeing there for me, from Grundschule on to this day. Your friendship has been invaluable. Thank you to Kyle Gaffney for introducing me to the field of mathematical biology during my undergraduate years and thank you to Christel Hohenegger for your mentorship. I am ever grateful for the chance you gave me to experience mathematical research. Many thanks also go to Mike Daniels, whose wit and wisdom have contributed greatly to my undergraduate career, and to my study buddy, Jammin Gieber.

Lastly, but most importantly, I would like to thank my family for all the opportunities I have received, in particular my mom, Kristine Senter, for making sure I received the best education I could so that all opportunities would be open to me. I am grateful for my grandfather, Franz Konheisner z'l, for his infinite patience with all of my varying intellectual interests as a child. I would like to thank my aunt, Eva Fleming, my grandmother Marie Konheisner z'l, and my dad, Ali Najeh, for all of their support throughout the years. Special thanks go to my wife Elizabeth. She has been there throughout my PhD studies, supporting me no matter how stressful life became or how late at night I would need to study or work. I am grateful to have you in my life, and for our joint graduate adventure.

TABLE OF CONTENTS

LIST OF FIGURES . . . . . **xii**

LIST OF TABLES . . . . . **xv**

CHAPTER 1: INTRODUCTION **1**

1.1 A semi-automated method for the generation of finite-difference meshes 2

1.2 Boundary Interactions and Locomotion 3

CHAPTER 2: A SEMI-AUTOMATED FINITE DIFFERENCE MESH GENERATION SOFTWARE **6**

2.1 Introduction 6

2.2 *MeshmerizeMe* Implementation 8

2.2.1 Overview of contour extraction 12

2.2.2 Going from curves to mesh 15

2.3 Examples: Bringing everything together 19

2.3.1 IB2d: Dragonfly Wing Veins Example 19

2.3.2 IBAMR: Lymphatic Capillary Example 23

2.3.3 IB2d: Starfish Example 25

2.4 Discussion 28

CHAPTER 3: THREE-DIMENSIONAL FLEXIBLE CLAP AND FLING IN TINY INSECT FLIGHT **31**

3.1 Introduction 31

3.2 Methods 34

3.2.1 Immersed Boundary Method 34

3.2.2 Dimensionless Parameters 37

3.2.3 The 3D Model Wing and its Motion 38

3.2.4 Simulation Parameters 39

3.3 Results . . . 39
3.3.1 Flow Fields . . . 39
3.3.2 Effect of Re . . . 44
3.3.3 Effect of Flexibility . . . 47
3.4 Discussion . . . 51

CHAPTER 4: THE HYDRODYNAMICS OF METACHRONAL PADDLING IN BRINE SHRIMP . . . 54
4.1 Introduction . . . 54
4.2 Methods . . . 55
4.2.1 Immersed Boundary Method . . . 55
4.2.2 Model Formulation . . . 56
4.2.3 Analysis of Flow Fields . . . 58
4.2.4 Measurement of Brine Shrimp Kinematics . . . 59
4.3 Results . . . 60
4.3.1 Brine shrimp kinematics . . . 60
4.3.2 Brine Shrimp Swimming . . . 60
4.3.3 Effect of Re . . . 63
4.3.4 Effect of Leg Spacing for Brine Shrimp . . . 66
4.4 Discussion . . . 67

CHAPTER 5: CONCLUSIONS . . . 70
5.1 Limitations of the studies and future work . . . 71

APPENDIX A: A DERIVATION OF THE NAVIER-STOKES EQUATIONS . . 74
A.1 Conservation of Mass . . . 74
A.2 Conservation of Momentum . . . 75
A.3 Putting it Together . . . 76

APPENDIX B: DETAILS ON THE IMMERSED BOUNDARY METHOD . . . 77
B.1 Governing Equations of IB . . . 77
B.2 Numerical Algorithm . . . 80

APPENDIX C: BACKGROUND ON BÉZIER CURVES 81

REFERENCES 83

LIST OF FIGURES

2.1 Flow chart illustrating the MeshmerizeMe workflow. Blue rounded boxes represent files, while square orange boxes represent user actions. The flow chart illustrates the two main entry points into the work flow from the source image: automatic edge extraction using the ContourizeMe script as well as manually creating the SVG by drawing "over" the image using software like Inkscape. 9

2.2 The *ContourizeMe* GUI in action on Manjaro Linux 18.0.4 with Gnome DE emulated in VirtualBox. The main window to the left allows the user to select the desired type of parameterization of the source image. The user can chose grayscale, RGB, or HSV. The sliders allow the user to set the desired threshholding in that parameterization. The detected edges are displayed in the live image (upper right) in green. The user may optionally display the result of the paramterization and filtering (lower right). Starfish image reproduced from [1]. 11

2.3 A distribution plot of median errors calculated from the meshes created for the error experiments in Section (2.2.2). 18

2.4 **(a)** The original public domain image of a dragonfly wing [2]. **(b)** The partial region of the dragonfly wing that was chosen for numerical simulation in IB2d. Some vessels were opened so that flow would have obvious entry and exit paths. 20

2.5 Snapshots comparing the magnitude of velocity for different Re, $Re = 0.6, 6, 60$. 22

2.6 Snapshots comparing the pressure for different Re, $Re = 0.6, 6, 60$. 23

2.7 (a) Source image for modeling flow through a junction in a lymphatic capillary, courtesy of Dr. Wenjing Xu from the Kathleen Caron lab. (b) SVG rendering of bifurcating vascular structure. (c) Vertex points discretized from SVG file. (d) Colormap of the magnitude of velocity of flow through a bifurcating dermal lymphatic capillary. Note that the vessel ends were artificially extended to allow for fully developed flow within the vessels. 25

2.8 (a) Original Starfish image courtesy of NOAA Sea Grant Program [1] (b) SVG image of the starfish generated by *ContourizeMe* script (c), (d), (f) The discretized geometry at grid resolutions of 128×128, 256×256, and 1024×1024 respectively. (e) Computational geometry containing a starfish in a channel with prescribed oscillatory parabolic inflow (see B.1). 26

2.9 Snapshots showing oscillatory flow past a rigid starfish at $Re = 800$ during the first 6 pulsation periods. The background heatmap illustrates vorticity. 27

2.10 Snapshots showing oscillatory flow past 1, 3, or 5 rigid starfish at $Re = 800$ during the first pulsation period. The background colormap illustrates vorticity. 28

3.1 Clap and fling in 3D. (A) At the beginning of a wing beat the wings are held close together. (B) the wings then rotate apart, and (C) form two large leading edge vortices. (i) Schematic of the fling (2), (ii) a side view of two immersed boundary wings engaged in fling, and (iii) a front-view of two immersed boundary wings engaged in fling. 33

3.2 Top view of two wings performing a clap and fling at $Re = 8$ with $\eta' = 6.25 \times 10^5$. Snapshots were taken during the first stroke every 0.05 units of dimensionless time. i-x represent the downstroke, and xi-xx represent the upstroke. The fling is performed during i=iv, and the clap occurs from xviii-xx. Deformations are visible during fling (ii-iv), wing reversal (ix-x), and clap (xviii-xx). 41

3.3 Vorticity and velocity fields in a slice parallel to the xz-plane for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at times $t' = 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8$, and 2.9. The color map shows ω_y. 42

3.4 Vorticity and velocity fields in a slice parallel to the yz-plane for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at times $t' = 2.0, 2.05, 2.1, 2.15, 2.2, 2.25, 2.9, 2.95$, and 3.0. The color map shows ω_x. 43

3.5 Dimensionless force vs. dimensionless time for one wing (A, B) and a pair of wings (C, D) performing clap and fling at $Re = 4$, 8, and 16. Forces are shown for the third stroke. 45

3.6 Average and peak dimensionless forces as a function of Re for one wing (A, B) and two wings (C, D). Lift is shown in black and drag in red. 46

3.7 Vorticity and velocity fields in a slice for two wings performing a clap and fling at $Re = 4, 8, 16, 32, 64, 128$. The 2D slice showing the flow characteristics is parallel to the xz-plane. Snapshots were taken during the third stroke at time $t' = 2.25$. 48

3.8 Aerodynamic forces for a range of wing flexibilities. A) Dimensionless lift vs. dimensionless time and B) dimensionless drag vs. dimensionless time at $Re = 8$ for $\eta' = \eta_r/16, \eta_r/4$, and $2\eta_r$. C) Average dimensionless lift and drag and D) peak dimensionless lift and drag taken during the third stroke for a range of η'. D) Average lift over average drag for a range of η'. 49

3.9 Vorticity and velocity fields in a slice for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at time $t' = 2.25$. The flexibilities were set to $\eta' = \eta_r/16, \eta_r/8, \eta_r/4, \eta_r/2, \eta_r, 2\eta_r$. 50

4.1 Schematic of a metachronal paddler with four paddles to illustrate the model used in our simulations. The left-hand side represents the head direction in which the paddler swims, while the right-hand side represents the tail area. The angles between the legs and the body of the swimmer are marked as $\alpha_i(t)$, the length of the leg is marked by l, and the constant distance between adjacent paddles is marked by w. All simulations we ran utilized swimmers with eleven paddles. 57

4.2 A frame from the high speed recording of a female brine shrimp swimming freely. The thoracopods were annotated for kinematics measurements (4 to 7 in the image), and the angle between the body and the tip of the leg was measured to describe the movement of the legs. 59

4.3 (A) The average flux $\langle Q(t) \rangle$ for the reference model of a brine shrimp with eleven legs equidistantly placed at a distance of $w = L/2$ swimming with a phase difference of $\phi = 8\,\%$ at $Re = 40$ and (B) the power consumption through the first twelve strokes. The larger peak in power consumption occurs when 40 % of the stroke has been completed and all legs face towards the swimmer, while the second peak occurs at 85 % and occurs when all legs point away from the middle of the swimmer. We note that although it takes a large number of strokes for the flux to approach steady state, power consumption reaches a steady state in a much shorter period of time. 61

4.4 The vorticity and velocity field for the $Re = 40$ reference model with leg distance $L/2$ over the last stroke at times (A) $t = 149.00$ (B) $t = 149.20$ (C) $t = 149.40$ (D) $t = 149.60$ (E) $t = 149.80$ (F) $t = 150.00$. 62

4.5 Average flux across all the different Re that were examined in section 4.3.3. The higher the Re, the longer the flow takes to achieve steady state and the higher the average flux $\langle Q \rangle$ at steady state. 63

4.6 Average flux and power consumption as a function of Re during the final stroke of simulations in section 4.3.3. The average flux steadily increases with Re, reaching a near plateau towards the highest Re simulated. Note the nearly five-fold increase in magnitude of $\langle Q \rangle$ between $Re = 0.01$ and $Re = 40$. The average power is highest for the lowest Re, falling off drastically as Re values increase. 64

4.7 Vorticity and velocity for the end of the final stroke for varying Re as described in section 4.3.3. Subfigures represent (A) $Re = 0.1$ (B) $Re = 0.5$ (C) $Re = 1$ (D) $Re = 5$ (E) $Re = 10$ (F) $Re = 20$ (G) $Re = 30$ and (H) $Re = 40$. 65

4.8 The normed efficiency of the swimmer with eleven paddles using a phase difference of $\Delta\Phi = 0.08$ as functions of both Re (A) and leg-spacing w (B). Efficiency increases nearly linearly with Re, being maximized at $Re = 40$. The lowest Re values are very inefficient, with normed efficiency less than 0.01 for $Re < 1$. When changing the paddle-spacing the normed efficiency shows a less dramatic but more non-linear dependency on the tested values. It was maximized towards the middle of our range of values, with maximal normed efficiency achieved at $w = 3L/4$. 66

4.9 Results of a leg-spacing study for Reynold's Number 40. Our default spacing in the other sections was $w = L/2$, which corresponds to the left-most data point at distance $w = 0.075$. We note that at a distance of greater than $5L/8$ we see a steady decrease in average flux. The average power consumption however seems to be minimized at a distance $w = 3L/4$. 67

4.10 End of last stroke for different widths w between the base of the paddles. (A) $w = L/2$; (B) $w = 5L/8$; (C) $w = 3L/4$; (D) $w = 7L/8$; (E) $w = L$. 68

B.1 Illustrating the subset of the insect vein geometry where the prescribed inflow condition is enforced. 79

LIST OF TABLES

3.1 The numerical parameters. Lengths are nondimensionalized by w_{chord} and time is nondimensionalized by the flapping period. 39

CHAPTER 1

Introduction

All organisms must deal with the transport of fluids (air, water, blood, etc) for gas exchange and many also swim or fly through these fluids. For example, the expansion and contraction of the lungs moves air for the exchange of oxygen and carbon dioxide, and the the expansion and contraction of a jellyfish bell propels it through the water. Most of these organisms are also highly deformable and such elastic deformations can be used to move fluid, move through fluid, and resist fluid forces. For example, many leaves will reconfigure in strong winds into cone shapes that reduce drag [3]. Understanding the interactions of a deforming organism with a fluid gives rise to a particular class of problems known as fluid structure interaction (FSI), which are both ubiquitous and challenging. FSI problems can be grouped into external and internal flow problems. Examples of external flow include flight and gliding. On the internal flow side, examples of problems include cardiovascular flows, lymphatic flows, and airflow in the lungs.

Different strategies have been developed for dealing with FSI problems. One of the most robust classes of methods, specifically for dealing with problems in biological fluid dynamics, is the immersed boundary method [4, 5]. Originally developed by Peskin [6], the fluid equations are described using an Eulerian framework, and the elastic equations describing the immersed structure are provided using a Lagrangian framework. The immersed boundary method includes a way to handle the interaction between the fluid and the structure whereby the structure is moved at the local fluid velocity and the force exerted by the structure on the fluid is interpolated as a source term in the momentum equation. This method has been successfully used to study a wide variety of problems in biological fluid dynamics, including cardiovascular fluid dynamics [6, 7, 8], fish swimming [9, 10], blood clotting [11, 12], insect flight [13, 14], and flows generated around pulsing soft corals [15].

In this thesis, I will consider 1) a method for obtaining a finite difference discretization of a boundary from images or drawings for use in immersed boundary simulations and 2) immersed boundary models for two methods of locomotion where paddles interact during swimming and where

wings interact during flying. The second chapter is motivated by the need for a semi-automated process to generate complicated geometries for biological structures. This is important as most studies using the original finite difference version of the immersed boundary method use relatively simple shapes to avoid this issue. To this end, I will develop a new method to create meshes directly from images of biological specimens in chapter 2. In the following two chapters, the immersed boundary method will be used to reveal the effects of interacting flexible wings and rigid paddles on lift and thrust production. In flight, the clapping of a pair of wings together followed by a rapid fling apart is known to enhance lift generation in tiny insects. In chapter 3, I will model a pair of insect wings as flexible ellipsoids performing the clap and fling motion in three-dimensions. I will then quantify how flexibility affects lift and drag generation. In chapter 4, brine shrimp legs will be modeled as a series of 11 rigid paddles that move in a idealized metachronal fashion to simulate the 11 pairs of paddles in adult brine shrimp. I will then quantify the average volumetric flux generated as a function of the phase difference between neighboring paddles, the spacing between paddles, and the Reynolds number.

1.1 A semi-automated method for the generation of finite-difference meshes

Different strategies have been developed for dealing with FSI problems, with two major classifications that can be used to distinguish between the different methods - one based on the mathematical framework and one based on the mesh discretization. The classification by mathematical framework can be broken down into two major groups. One group attempts to formulate the entire FSI problem in a single mathematical framework so that all fluid and structural equations are solved simultaneously. This class of methods is sometimes refered to as the monolithic approach. A second group of methods, sometimes referred to as the partitioned approach, treats the fluid and structural equations separately. In this way, specialized algorithms can be used in each part of the problem with the two results kept in sync via their interfaces.

The mesh discretization strategies can be classified as conforming and as non-conforming methods. In a conforming mesh method, the interface is treated as a boundary condition with the mesh conforming to this boundary. In this type of method, the mesh has to be recreated whenever the boundary moves or is deformed. In a non-conforming mesh method on the other hand, the interface is considered not a boundary condition but rather a constraint. This allows for the use of meshes

that do not conform to the boundary itself and which need not be recreated whenever the interface moves or is deformed.

For this component of the thesis, we will discretize the boundaries of biological structures from images for use in immersed boundary simulations. The immersed boundary method uses a partitioned, mesh non-conforming approach, as the fluid and structural equations are solved separately. This method will be described in more detail in Appendix B. The immersed boundary method in principle deals with immersed structures that do not take up volume. To deal with volume occupying structures, the structure itself may be represented as a mesh of interconnecting fibers. The semi-automated code we present in chapter 2 discretizes the edges of biological structures with evenly spaced points appropriate for immersed boundary simulations. Volumes can then be generated by filling in the enclosed region with a network of fibers.

1.2 Boundary Interactions and Locomotion

The interaction of boundaries is often used in animal locomotion to generate additional thrust, enhance lift, and maneuver. Such interactions include wing-wing interactions, fin-fin interactions, and body-fin interactions, among others. For example, the pteropod mollusc *Clione limacina* swims using wing-like parapodia that touch during both the upstroke and downstroke, generating additional lift [16]. The ciliary plates of comb jellyfish beat in a metachronal fashion, and the interactions of neighboring plates and the resulting shed vortices generate additional thrust [17].

The first boundary interaction problem I consider in this thesis is the clapping of wings in tiny insect flight. Insect flight is a particularly challenging and interesting topic. As most insects are small, they experience substantially different aerodynamics than larger animals such as birds. Wingbeat frequencies are in the 30 Hz to 1000 Hz range. This requires special adaptation, as the power output of skeletal muscle already deteriorates at frequencies lower than these. It additionally poses a challenge if beating of the wings is to rapidly cease, since muscle contraction is typically regulated by Ca^{2+}, which is slow to turn off as it requires active pumping of the Ca^{2+} into the sarcoplasmic reticulum against its electrochemical gradient. Despite these issues, however, insects are readily observed to fly [18].

A further challenge in quantifying and modeling insect flight is due to the fact that they fly in the intermediate Reynolds number range $O(1)$ to $O(10,000)$, where viscous forces become increasingly important. Interdisciplinary work over the past three decades has revealed novel aerodynamic

mechanisms for insect flight, including the presence of a stable attached leading edge vortex, lift augmentation through wing rotation, and wake capture [19, 20, 21]. The aerodynamics of flight in the smallest insects is possibly even more intriguing. Unlike 'larger' insects, including fruit flies and dragonflies, the maximum lift generated during flight is much smaller than the thrust or drag experienced [13]. As the Reynolds number approaches 1, active flight is no longer observed in nature, potentially due to the fact that the force acting against gravity is significantly lower than the total force generated [22, 23].

The clap and fling mechanism has been observed in every tiny insect that has been filmed in flight to date [24, 25, 26, 27]. In this case, the wings are either clapped together or come very close together at the end of the upstroke. The wings then rotate about the trailing edge before being flung apart during the beginning of the downstroke. As a result, two large leading edge vortices are formed that can greatly enhance lift for the first part of the downstroke [14, 24, 28]. This mechanism has been studied extensively using mathematical, numerical, and experimental models in two dimensions [14, 23, 28, 29]. More recent work has considered three-dimensional effects [30, 31]. However, the role of wing flexibility has not been considered in depth, and this is the goal of chapter 3 of the thesis.

The second example of boundary interaction considered in this thesis considers the interacting swimming appendages of brine shrimp. Brine shrimp, like many other crustaceans, beat their swimming appendages in a metachronal fashion to generate thrust and, in some cases, lift. The movement of each paddle consists of two reciprocal motions, a power stroke and a recovery stroke. The power stroke generates thrust by extending a limb and pushing the fluid past the swimmer. The recovery stroke returns the limb to its initial position at the beginning of the power stroke in a manner such as to reduce drag in the repositioning process. When arranged in series, additional thrust can be generated when neighboring paddles are moved out-of-phase [32].

The goal of chapter 4 is to extend previous work by Granzier-Nakajima *et al.* [33] on metachronal paddling in crayfish to the specific case of brine shrimp. Crayfish use four pairs of swimmerets placed relatively far apart to generate thrust. The swimmerets beat at Reynolds numbers on the order of 100 or higher with a 20-25% phase difference. Brine shrimp, on the other hand, begin swimming at Reynolds numbers close to 1 when newly hatched and swim at a Reynolds number near 40 as adults. They have 11 pairs of closely spaced appendages that beat with a 8-12% phase difference.

Granzier-Nakajima *et al.* [33] previously showed that a 22 % phase difference is optimal for thrust generation using crayfish parameters. In that chapter, we explore how volumetric flux and time to reach steady state depend upon Re and leg spacing for 11 paddles beating with an 8 % phase difference, representative of brine shrimp.

CHAPTER 2

A Semi-Automated Finite Difference Mesh Generation Software[1]

2.1 Introduction

The immersed boundary method (IBM) is a mathematical formulation and numerical method for fully-coupled fluid-structure interaction problems that dates back to Peskin in 1972 [6]. Since its creation, the IBM has been used to study a wide variety of problems in biological fluid dynamics and fundamental fluid dynamics at low to intermediate Reynolds numbers ($Re < 10,000$). Diverse examples include the aerodynamics of insect flight [13, 26, 34], lamprey swimming [10, 9], jellyfish swimming [35, 36], and fluid flows through organs such as the heart and esophagus [8, 37, 7]. The relative ease of implementation and the availability of open source codes has made it particularly useful in research and education [38, 39, 4].

The original IBM formulation discretizes immersed, elastic boundaries on a curvilinear finite difference mesh. Many immersed boundary studies are performed in 2D and use simple geometries with easy mathematical descriptions such as plates [13, 40], strings [41, 42], tubes [37, 43, 44], ellipses [45, 46], hemiellipses [47, 36], and circles [48, 49, 50], or in 3D with spheres [51] or cylinders [52]. In other cases, more complicated geometries are manually constructed by the user via explicit mathematical functions or sets of functions that describe the elastic boundary [53, 8, 54]; this endeavor is, however, non-trivial. David Baraff, a Senior Research Scientist at Pixar Animation Studios has publicly said, "I hate meshes. I cannot believe how hard this is. Geometry is hard." [55].

This immediately highlights a challenge in performing immersed boundary simulations for many biological applications that have complicated geometries. Most meshing tools are finite element based, such as MeshLab [56], Gmsh [57], or TetGen [58], all of which are open source. As far as

[1]This chapter previously appeared as an article in Bioinspiration & Biomimetics. The original citation is as follows: Senter, D. Michael, et al. "A semi-automated finite difference mesh creation method for use with immersed boundary software IB2d and IBAMR." *Bioinspiration & Biomimetics* 16.1 (2020): 016008.

we are aware, there is not an openly available and easy to use tool for generating curvilinear finite difference meshes. A few finite difference meshing tools that are available constrain the mesh to a Cartesian grid using cuboids, such as the open source AEG Mesher [59], which was designed for electromagnetic simulations, and the propriety software Argus ONE [60], which was designed to help incorporate geographic information system (GIS) into numerical models [61].

What's more, many applications, especially those in biology and medicine, usually have some imaging data from which a mesh is estimated. For example, imagine generating a numerical model of blood flow through an arterial network. There are highly resolved images that clearly illustrate arterial branching patterns from which desirable geometric data, e.g., artery diameters, lengths, branching locations, etc., can be obtained. To perform numerical simulations that reveal the spatial variations in the flow due to small scale geometric effects, one must reconstruct this arterial network in detail. Even for 2D simulations this process is non-trivial, as one would first need to recreate the structure using parametric functions and then select particular parameter values that sample the structure's geometry to obtain a computational mesh with equally-spaced points. While this laborious approach may work for some simplified arterial geometries, if the actual arterial network contains walls that are not perfectly smooth or flat, simple tubular models may be insufficiently detailed and hence give rise to non-realistic results. Our software aims to fill this gap using edge detection on the original image, thereby preserving the original pattern and information from the images. Bézier curves are then used to mathematically describe the images, after which they are appropriately sampled to obtain a curvilinear mesh.

The challenge here is two-fold: first a continuous, parameterized description of the boundary of interest must be found, potentially through image segmentation, and then this boundary must be represented as a finite difference mesh with sampling to give the desired geometric spacing between adjacent geometric nodes. Given the widespread use of the IBM approach in both research and education [4, 38, 39, 62, 63], we found it useful to create the open source software package *MeshmerizeMe*, a tool that both detects boundaries in image data and creates finite difference meshes where the nodes are nearly uniformly spaced. The output files are designed to be coupled with IB2d [64, 38, 39], IBAMR [65], and other immersed boundary 2D software.

We provide an overview of the software *MeshmerizeMe* in Section 2.2. In particular, we detail *MeshmerizeMe*'s implementation, workflow (Section 2.2.1), and how the software computes a

discretized mesh from parametric curves (Section 2.2.2). We then present a variety of examples using the software in Section 2.3; including two internal flow examples and one external flow example. The examples include hemolymph flow through dragonfly wing veins (Section 2.3.1), flow through a lymphatic capillary (Section 2.3.2), and oscillatory flow past a starfish (Section 2.3.3).

The most straightforward immersed boundary simulations using this software would be developed to simulate the flow past or through nearly rigid, complex, biological boundaries, as demonstrated in these examples. We do not currently have a method for using multiple images to simulate moving boundaries. There are, however, a few ways that one can model moving and deformable boundaries. To begin, each of the Lagrangian points can be translated or rotated using a prescribed mathematical function rather than moving the boundary based on image tracking. This approach has been used for a variety of biological applications, including flapping insect wings [13], swimming jellyfish [66, 36], flapping swimmerets [67] and heart pumping [44]. If, in addition to vertex points, springs and beams are added to pairs or triads of vertex points, elastic deformations due to the fluid-structure interaction can also be simulated. This type of approach has been used for leaves in flow [3], flapping filaments [41], the deformation of prey prior to puncture [68], and the movement of red blood cells [11, 12].

2.2 *MeshmerizeMe* Implementation

MeshmerizeMe is a software package for the creation of 2D geometry files for use with open source immersed boundary software such as IB2d and IBAMR. The software comes with two main scripts: 1) *ContourizeMe*, which reads in an image file and uses automatic edge detection to extract contours of interest into an SVG file, and 2) *MeshmerizeMe*, which processes SVG files and IB2d- or IBAMR-style input2d files to create *.vertex files describing the geometry of the SVG file at the appropriate resolution. Both SVG and vertex files are UTF encoded text files. SVG is a widely supported vector graphics format, while the vertex-format is used by IB2d and IBAMR to describe Lagrangian mesh points in an immersed boundary simulation. The *MeshmerizeMe* script also includes a tool that uses Matplotlib to allow the user to plot the geometry created by *MeshmerizeMe* for visual verification. These scripts are written to run in Python 3.x, and upon installation both scripts are added to the path on a Linux and Mac environment, but the scripts themselves are also compatible with Windows. This dual-script setup allows the end-user two distinct entry points into the workflow; see Fig. (2.1) for an illustration.

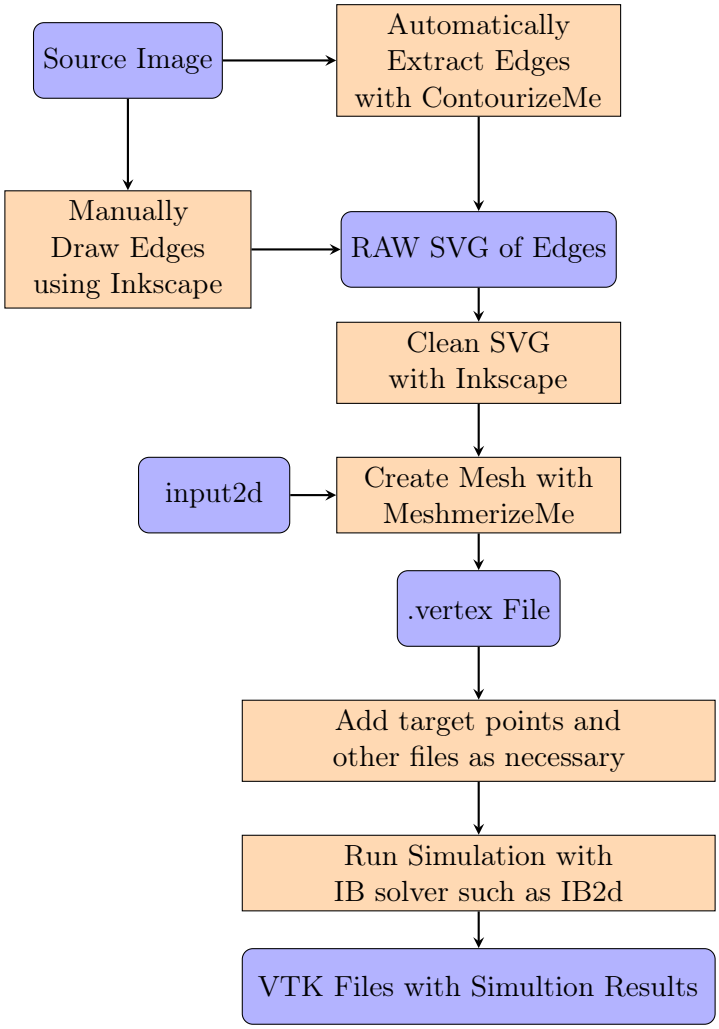

Figure 2.1: Flow chart illustrating the MeshmerizeMe workflow. Blue rounded boxes represent files, while square orange boxes represent user actions. The flow chart illustrates the two main entry points into the work flow from the source image: automatic edge extraction using the ContourizeMe script as well as manually creating the SVG by drawing "over" the image using software like Inkscape.

To provide a concrete example of the workflow, suppose an image is available either from the field or an experiment. To detect the edges and generate an SVG file, the user would run the *ContourizeMe* script on the desired image file (e.g., by typing `ContourizeMe image.jpg` in the commandline). This opens a GUI with several features that may be used to modify and enhance the image (see Figure 2.2). Note that common image formats such as jpg, png, and tiff are supported. For best performance, the image should provide a good contrast between the object boundary and background, while also having little noise. If this is not the case, *ContourizeMe* allows the user to adjust the image contrast and saturation[2] using simple sliders to better highlight the boundary of interest. Using a slider for the pixel cutoff, the smoothness of the matched curve can be adjusted to account for noise. All of these sliders update in real time. If this proves insufficient, unwanted edges that were detected can easily be deleted at a later step. Once the user is satisfied with the result, the curve is exported to an SVG file to be used as input for the *MeshmerizeMe* script.

[2] those changes are temporary and do not affect the original image data

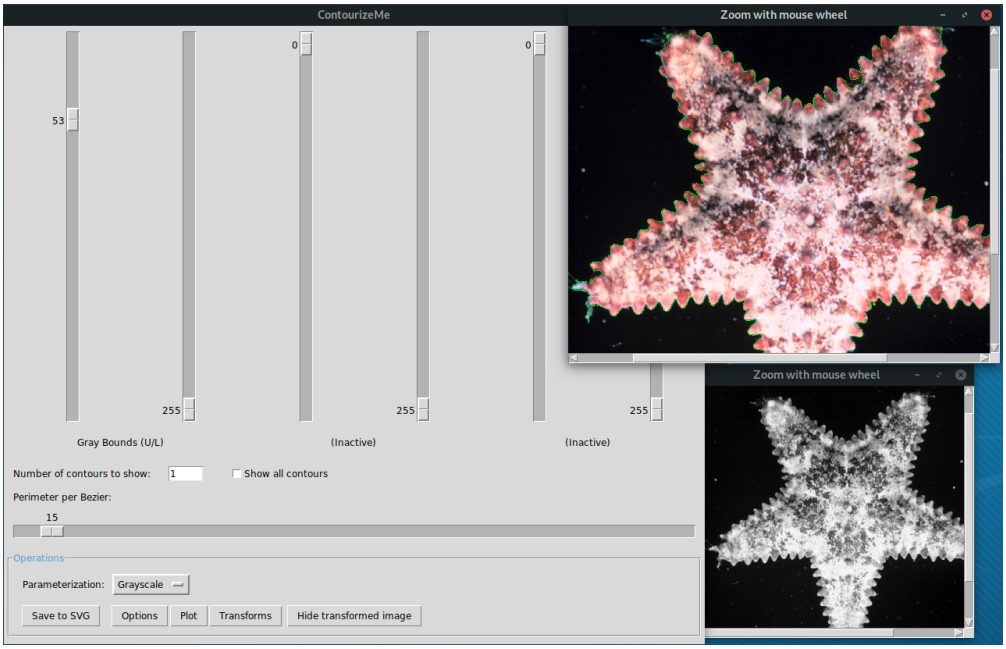

Figure 2.2: The *ContourizeMe* GUI in action on Manjaro Linux 18.0.4 with Gnome DE emulated in VirtualBox. The main window to the left allows the user to select the desired type of parameterization of the source image. The user can chose grayscale, RGB, or HSV. The sliders allow the user to set the desired threshholding in that parameterization. The detected edges are displayed in the live image (upper right) in green. The user may optionally display the result of the paramterization and filtering (lower right). Starfish image reproduced from [1].

In some cases, the original image quality may be high enough to proceed directly to the discretization phase. In most cases, however, the user will want to make minor edits to the SVG file to remove any potential artifacts, such as curves corresponding to background noise in addition to the boundaries of interest. These edits can be done using common vector graphics software such as Inkscape (open-source) or Adobe Illustrator (commercial). Note that this also provides an alternate entry-step in the *MeshmerizeMe* workflow: curves can be freely drawn using such software if image data is either not available, the boundary structure is purely hypothetical, or in cases when the original image is too poor in quality for reliable edge detection. In the latter case, image layers may be used in software such as Inkscape that allows the user to trace over the desired edges to create the SVG file with the necessary boundaries. Once the SVG file has been cleaned in this manner, it is advised to collapse underlying groups and simplify contiguous paths, which in some vector graphics software can be done from the "save" menu. In other cases, software to do this is freely available online, such as SVGO and SVGOMG [69, 70].

With the desired geometry extracted into an SVG image, the next step consists of making a folder in which the SVG file itself can be found, as well as an IB2d-style file called `input2d`. This latter file includes information such as the spatial discretization step size (more details on this can be found below). Our script reads this file to calculate and sample the appropriate mesh. *MeshmerizeMe* is run by pointing it to the appropriate SVG file (e.g., typing `MeshmerizeMe image.svg` from the command line). It will then create the `.vertex` file containing the mesh. Note that the filename of the vertex file will be taken straight from the input2d file, regardless of the SVG filename. If multiple meshes are to be created, *MeshmerizeMe* can be run in batch mode by providing it with a list of file names. We also support piping from STDIN. This allows the user to easily pass a list of file names, such as one created by the find command, to *MeshmerizeMe* for batch processing.

The resulting `.vertex` file can then be used as input for immersed boundary simulations using IB2d and IBAMR. Note that the user will need to supply some additional information as to the relationship between the boundary points, for example whether or not they are connected with springs, beams, masses, and so forth. Currently, files that store this information must be manually created, although a few of these relations are implemented as classes in the MeshmerizeMe library to help with writing such scripts. Once this stage is completed, the immersed boundary simulation is ready to run.

2.2.1 Overview of contour extraction

In this section, we provide an overview of how *ContourizeMe* extracts contours from images. This is motivated by the need to accurately estimate the shapes of objects from planar images or within some cross-section. Many techniques, ranging from edge-finding using image gradients to image segmentation accomplished through supervised training of Deep Neural Nets, have been proposed as generalized methods to extract such edges [71, 72]. The niche filled by *MeshmerizeMe* is to easily obtain 2D meshes from image data that can be directly used in IB2d and IBAMR. In essence, it allows for the semi-automated generation of meshes from image data using simple contour estimation from hand chosen thresholds of pixel values [73] as a first step in the IBM workflow. This replaces the need to completely create the structure mesh manually by finding idealized functions approximating the shape of interest. This method was chosen for its simplicity and fast estimation in obtaining user-verified 2D shapes of arbitrary smoothness and precision.

The *ContourizeMe* GUI was developed with the Python package Tkinter. Contour estimation

from image thresholding works by first applying noise reduction to a given image if needed. The contour is assumed to be represented in the image by a gradient or steep change in the pixel values that separates the foreground, or object of interest (OOI), from the background. For many images this means the existence of one or three inequalities or pixel-value bounds (3 for the case of RGB and HSV values) that quantify this separation. Image noise from one or multiple sources can make these inequalities ill-defined. Possible sources of image noise are numerous and include sensor and electronic-circuit noise, analog-to-digital conversion errors, and even statistical quantum fluctuations [74, 75]. *ContourizeMe* provides implementations of various common noise reduction techniques that the user may choose from depending on the source and strength of the noise present in their own images.

In the next step after determining an appropriate noise reduction technique, the user manually determines one or more pixel value bounds depending on a given parameterization (RGB, grayscale, HSV, etc.) that forms the lowest-area hull that corresponds to the object of interest. This closed region is used to produce a binary image with pixel values of 1 corresponding to those contained in the provided region and 0 corresponding to those not in this region. A topological algorithm in OpenCV [73] is applied to this binary image to give contours that fully describe 'separate' clusters of homogeneous pixels (in this case pixels that all equal 1). This algorithm yields integer pixel estimates of the boundaries, which are then refined to sub-pixel estimates with user specified smoothness via the Chan-Vese algorithm. More details are provided in the following subsections.

These contours themselves are estimations of the shapes of interest that are then used to yield precise descriptions of the shapes as a set of continuous Bézier curves (see C). Bézier curves are constructed using evenly spaced points from the sub-pixel boundary estimates and exported in the SVG format.

Noise reduction Filtering algorithms, the topological contour estimation algorithm, and most of the image manipulations (such as RGB to HSV conversion, thresholding, etc.) are accomplished in *ContourizeMe* via Python bindings of the OpenCV package [73]. OpenCV is a computer vision suite developed in C++ built to tackle various problems including segmentation, 3D reconstruction, edge-finding, and other related tasks.

ContourizeMe's main GUI includes:

- An average filter which essentially is a type of down sampling that assumes the true value of any pixel can be estimated by the average pixel value of a K by K window surrounding that pixel. This is equivalent to convolving the image with a low-pass filter kernel.
- A Gaussian filter which convolves the image with a Gaussian kernel of a specified size and standard deviation in both the x and y directions. This is similar to the average filter, but pixels are weighted via the 2D Gaussian function specified before they are averaged.
- A median filter which instead of the average over a window, takes the median pixel value. This kind of filter is typically used for "salt and pepper" type image noise, and has the advantage of leaving only pixel values that would have been observed in the original image.
- A bilateral filter which is the recommended choice. The bilateral filter behaves similarly to the Gaussian filter, but in addition to weighting pixels by their spatial distance it also weights them by their difference in intensity in an attempt to preserve edges or gradient information.

While the bilateral filter is recommended because of its intended edge-preservation [76], one may want to employ one of the other convolution filters as they can smooth boundaries produced by the thresholding and topological algorithms. We must also stress that this selection is highly limited in scope and much more sophisticated and robust techniques for the denoising of images exist depending on the image acquisition method and content. It may be that making a model of the noise via a deep neural net such as UNet [77], CAIR [78], Noise2Noise or Noise2Void [79] may be required or produce better results. Any method may of course be employed before using this segmentation GUI.

Smoothing the results from OpenCV's algorithm In order to give the user control over the smoothness of the resulting curve they obtain, we use a Python implementation of the Chan-Vese level set algorithm [80]. This method of smoothing the curve ensures that reductions in the curvature are chosen such that they have minimal costs to accuracy and that the contour remains true to the original image. We allow users to specify both the error tolerance in pixels, as estimated from each iteration of the Chan-Vese algorithm, and the parameter α which controls the contribution of the total curvature of the boundary to the energy functional and thus the smoothness of the obtained contour.

2.2.2 Going from curves to mesh

The second part of our software package consists of a script that takes vector based graphics, specifically the "Scalable Vector Graphics" (SVG) standard, to obtain a discretized curvilinear mesh that describes the boundary of the object of interest. The idea behind vector graphics is to represent shapes in terms of control points of non-uniform rational basis splines (NURBS). Only control points of the parameterized curves are stored while the standard defines the basis polynomials themselves. The resulting curves can be represented smoothly at any scaling or resolution of interest and can easily be mapped to the simulation space using an affine transformation.

A variety of vector graphics file formats are available, several of which are proprietary. We have chosen to implement our software using the SVG standard because it is a popular, open-source standard and for its ease of use. SVG files are UTF encoded text files following an XML schema, making them amenable to XML parsing methods. A particular benefit of the SVG standard is that it is widely supported; if edge detection fails or gives insufficient resolution, multiple vector graphics programs such as Inkscape or Adobe Illustrator may be used to clean up or directly hand-draw the boundaries of interest from an image. The standard has support both for Bézier curves as well as geometric primitives (rectangles, triangles, etc.), and the current version of *MeshmerizeMe* utilizes the free path element, which encodes curves as Bézier curves.

To reduce the need for additional configuration files, *MeshmerizeMe* has been built to utilize the 'input2d' file format that is utilized by IB2d and IBAMR. This file is required, and the *MeshmerizeMe* code expects the following variables to be defined in the input2d file:

- `Lx`, `Ly`: the length of the computational domain in the x and y direction, respectively.
- `Nx`: number of points in the x direction.[3]

Even if the user chooses to use a different CFD software for the simulation, *MeshmerizeMe* can still be used to create the requisite mesh points. Strict adherence to the IB2d format is not required. A minimal working example of the input2d file required for *MeshmerizeMe* requires only four lines. The example below will create a mesh appropriate for a $[0, 0.5] \times [0, 0.5]$ domain with a 64×64 mesh.

[3] *MeshmerizeMe* expects a square discretization, that is $\Delta x = \Delta y$, but does not require a square computational domain.

```
Nx =  64
Lx = 0.5
Ly = 0.5
string_name = test
```

Please note that this minimal example is only sufficient for *MeshmerizeMe*. A simulation for IB2d or IBAMR will require additional settings in the `input2d` file, such as the fluid paramaters `mu` and `rho` and temporal information such as the desired time step `dt` and time the simulation is to run. Any such additional settings may be present in the `input2d` file, but will be ignored by *MeshmerizeMe*.

MeshmerizeMe will automatically compute the appropriate boundary point spacing of $\Delta s = \frac{1}{2}\Delta x$, where $\Delta x = \frac{L_x}{N_x}$. We note that it is standard in the immersed boundary literature to set the spacing between the immersed boundary points to half that of the spatial step for the Navier-Stokes solver, $\Delta s = \frac{1}{2}\Delta x$ [4]. This choice of spacing allows the boundary points to move independently while also restricting most of the flow between the points. Using these parameters, the software will parse the supplied SVG file itself and extract the path objects, splitting them up into individual Bézier curve objects. The control points are then converted to the experimental coordinate system defined by `Lx` and `Ly`.

Let $\boldsymbol{\gamma}(t) \in \mathbb{R}^2$ represent a particular Bézier curve. To create the mesh, we seek n parameters $\{t_i\}$ with $0 \leq t_1 < t_2 < \ldots < t_n \leq 1$ such that the distance between points on the curve with these parameters is fixed:

$$d(t_i) = \|\boldsymbol{\gamma}(t_{i+1}) - \boldsymbol{\gamma}(t_i)\| = \Delta s, \quad i = 1, ..., n-1\,. \tag{2.1}$$

We utilize a gradient descent method to find these parameters. Specifically, we define our cost function $J(\boldsymbol{t})$ using the mean squared relative error of all $d(t_i)$ values (scaled by $\frac{1}{2}$ to cancel the power of 2 coming from the partial derivatives in (2.4))

$$J(\boldsymbol{t}) = \frac{1}{2(n-1)} \sum_{i=1}^{n-1} \left(\frac{d(t_i) - \Delta s}{\Delta s} \right)^2 , \boldsymbol{t} = \begin{bmatrix} t_1 & ... & t_n \end{bmatrix} \tag{2.2}$$

and minimize $J(\boldsymbol{t})$ by iteratively updating $\boldsymbol{t}$ with a variable learning rate α:

$$\boldsymbol{t}_{new} = \boldsymbol{t}_{old} - \alpha \nabla J(\boldsymbol{t}) \,, \tag{2.3}$$

where

$$\frac{\partial J}{\partial t_j} = \begin{cases} \dfrac{-1}{(n-1)(\Delta s)^2} \left[\dfrac{1}{d(t_j)} \left(d(t_j) - \Delta s \right) \langle \boldsymbol{\gamma}(t_{j+1}) - \boldsymbol{\gamma}(t_j), \nabla \boldsymbol{\gamma}(t_j) \rangle \right], & \text{if } j = 1 \\ \dfrac{1}{(n-1)(\Delta s)^2} \left[\dfrac{1}{d(t_{j-1})} \left(d(t_{j-1}) - \Delta s \right) \langle \boldsymbol{\gamma}(t_j) - \boldsymbol{\gamma}(t_{j-1}), \nabla \boldsymbol{\gamma}(t_j) \rangle \right], & \text{if } j = n-1 \\ \dfrac{1}{(n-1)(\Delta s)^2} \left[\begin{array}{l} \dfrac{1}{d(t_{j-1})} \left(d(t_{j-1}) - \Delta s \right) \langle \boldsymbol{\gamma}(t_j) - \boldsymbol{\gamma}(t_{j-1}), \nabla \boldsymbol{\gamma}(t_j) \rangle \\ \quad - \dfrac{1}{d(t_j)} \left(d(t_j) - \Delta s \right) \langle \boldsymbol{\gamma}(t_{j+1}) - \boldsymbol{\gamma}(t_j), \nabla \boldsymbol{\gamma}(t_j) \rangle \end{array} \right], & \text{otherwise.} \end{cases} \tag{2.4}$$

Here, we use the notation $\langle \mathbf{a}, \mathbf{b} \rangle$ to denote the inner product between $\mathbf{a}$ and $\mathbf{b}$.

The number of points n to be found per path are estimated by dividing the arc-length by the desired length Δs. This may result in more dense than optimal spacing of points, but in practice achieves sufficient accuracy. The error will depend on the curvature of $\boldsymbol{\gamma}$.

We then evaluate our curve at the points t_i obtained from this technique to determine the discretized boundaries of interest. The produced Lagrangian mesh is output to a file called `fname.vertex` where 'fname' is based on the value of `string_name` taken from the 'input2d' file. The vertex file itself is a simple text file. The first line consists of an integer giving the total number of mesh points and the following lines contain one mesh point each, given as a space delimited pair of floats representing the x and y direction coordinates.

Distribution of Errors in Approximation To test the relative accuracy of our script, we created 5,000 SVG files each containing a randomly generated cubic Bézier curve. For purposes of uniformity during testing, each curve was generated on a 1000×1000 pixel domain to be mapped onto a 1×1 mesh domain using a 256×256 grid. All parameters were set to default values. A script was then run to calculate the minimum, maximum, mean, and median relative errors for each curve. The mean of the median relative error of curves in the script is 3.2% The middle 50% of median relative errors is in the 2.6-3.5% range. See figure (2.3) for the distribution of the median relative errors.

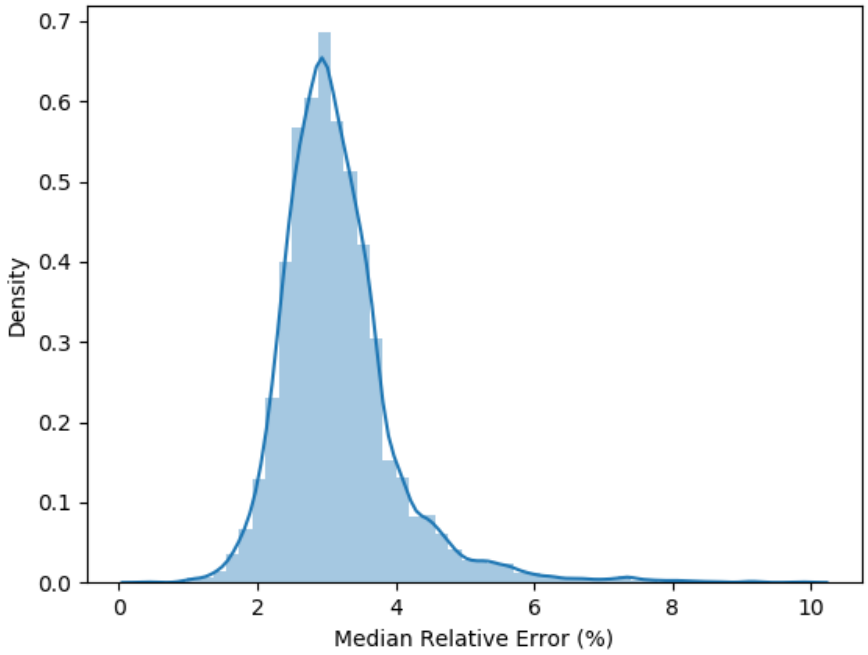

Figure 2.3: A distribution plot of median errors calculated from the meshes created for the error experiments in Section (2.2.2).

Note that if a higher degree of accuracy is desired, the user is able to experiment with different parameters that can influence the average error. *MeshmerizeMe* allows the user to set both the learning rate as well as the convergence threshold of the mean squared error (MSE) as command-line options when creating the mesh.

One potential limitation on the error is the method we have chosen to seed the curves. Specifically, after reading the SVG our script merges all individual SVG path objects into a single path object. This path object is then split into several sub-paths of equal arc-length. Each of these sub-paths are seeded with $n = L_{sp}/\Delta s$ points, where L_{sp} is the length of the sub-path and Δs is the desired Euclidean distance between mesh-points. For sub-paths with very large curvature, this seeding method may overestimate the number of points necessary for ideal discretization. In such cases, the user may re-run the MeshmerizeMe script with the `num-points` flag to manually specify a lower number of points per sub-path. In our experience, this is an unusual occurrence that can usually be solved by experimenting with the `num-points` flag.

It should also be noted that our current algorithm assumes the object of interest is a contiguous

path, that is we assume the object can be represented by a single poly-Bézier curve with at least geometric continuity. In the SVG file, this representation is not limited to representation by a single path element. When the SVG file is parsed, all path elements are merged into a single poly-Bézier path object. This object is then divided into several sub-paths of approximately equal length which are then processed in parallel. If the object of interest consists of two non-contiguous paths or multiple objects are represented in the source SVG, the *MeshmerizeMe* script will report a large error resulting from the distance between two sequential points on non-contiguous paths. In the presence of several non-contiguous paths, the minimization algorithm will lead to a slight skewing of points towards path boundaries.

2.3 Examples: Bringing everything together

We present several examples that illustrate the software's ability to recreate complex geometries. In each example, *ContourizeMe* is used to extract contours from images, *MeshermizeMe* is then used to compute the model's discretized geometry. The flow within or around the geometries is solved using an open-source implementation of IBM, either IB2d or IBAMR. The following examples are illustrated:

1. Hemolymph flow through dragonfly wing veins (Section 2.3.1)
2. Lymph flow through a branching lymphatic capillary (Section 2.3.2)
3. Oscillatory flow past a starfish or array of starfish (Section 2.3.3)

In every example, we present the original image on which the computational geometry is based, followed by images that illustrate how *MeshmerizeMe* computed its associated discretized mesh. Finally, we present computational results to illustrate successful integration of the geometry into the IBM software.

2.3.1 IB2d: Dragonfly Wing Veins Example

For our first example, we chose a public domain image of a dragonfly wing shot with a Canon EOS 5D Mark with a 100 mm lens [2]. For the purpose of running a tractable fluid-structure interaction simulation, we cropped this image to a section of the wing and manually occluded parts of the veins using the open source image manipulation software GIMP [81]. Figure 2.4 shows the original image of the dragonfly wing and the region that was chosen for numerical simulation.

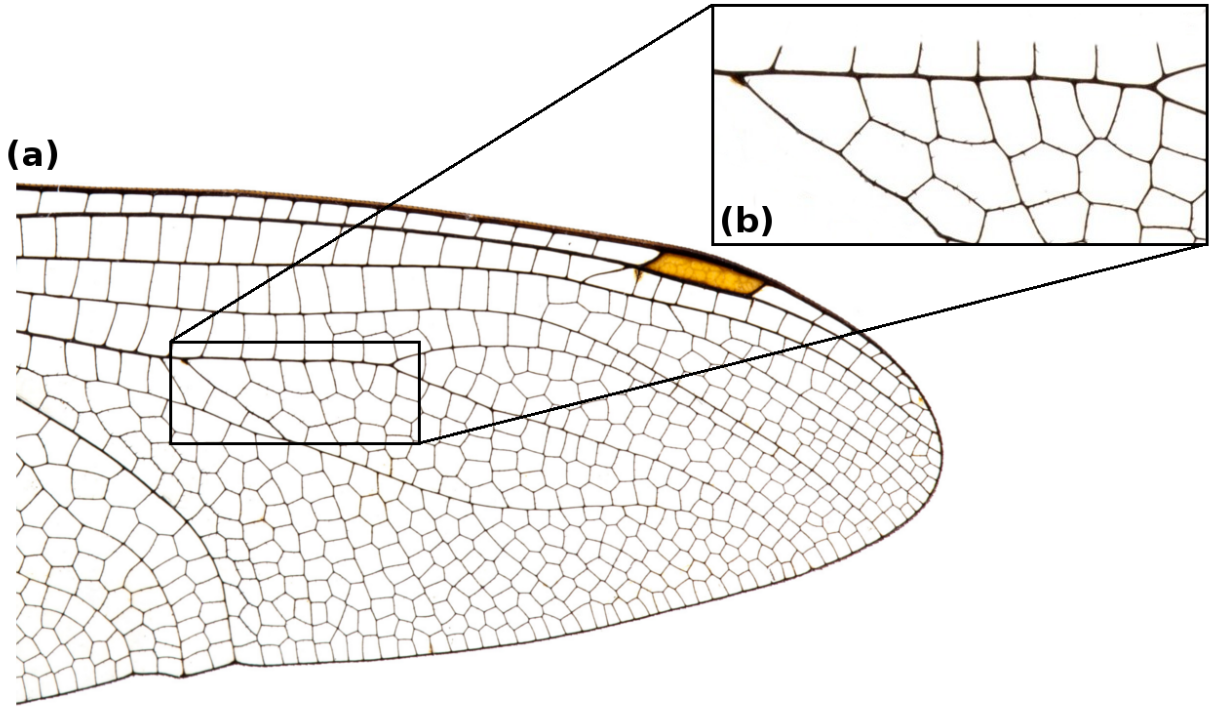

Figure 2.4: **(a)** The original public domain image of a dragonfly wing [2]. **(b)** The partial region of the dragonfly wing that was chosen for numerical simulation in IB2d. Some vessels were opened so that flow would have obvious entry and exit paths.

As shown in Figure 2.2, we used the *ContourizeMe* GUI to create boundaries describing the subsection of the wing vasculature. Briefly, the image was cropped so that only a couple dozen vessel segments would be considered. The image was then loaded into the *ContourizeMe* GUI. Noise reduction was then applied to the image, and a pixel bound was selected such that the lowest-area hull sufficiently matches the edge of the vessel network. The edges were then smoothed using the Chan-Vese algortihm, and the result was exported to SVG.

The *MeshmerizeMe* script was then used to obtain a curvilinear mesh from the SVG file. The x, y coordinates of this mesh were written to the `.vertex` file. The file contained the coordinates for approximately equally-spaced Lagrangian points that were then used as an input into IB2d. For the purpose of running an IB2d simulation where the complex geometry remains relatively fixed while the fluid is driven through it, we only require the vertex point of each Lagrangian Point, i.e. the (x, y) values of each point along the insect wing. Since the wing veins should be relatively rigid and not move, each Lagrangian vertex point was tethered to a target point (see B.1). This has the effect

of applying a force proportional to the distance between the location of the actual boundary and the desired position. In other words, the boundary is pushed back into place as fluid moves through the network. The necessary input information for IB2d is written in the `wing_veins.vertex` and `wing_veins.target` files.

This example can be found in the open source IB2d software available at `www.github.com/nickabattista/IB2d` and this example can be found in the following subdirectory, `IB2d/matIB2d/Examples/Example_MeshmerizeMe/Dragonfly_Wing/`. More details on IB2d and the immersed boundary method in general can be found in Appendix B.

To drive flow through the wing, a penalty force is applied to the fluid that is proportional to the difference between the local fluid speed and the local target velocity (see B.1). For our example, a parabolic flow profile is enforced at the inlet of the insect wing, and all subsequent flows through the veins result from that inflow and are not themselves prescribed (see Figure B.1). The full implementation of this simulation can be found in the `please_Compute_External_Forcing.m` script.

To illustrate flow through the wing vein geometry, we ran multiple simulations corresponding to various Reynolds numbers (Re), given by

$$Re = \frac{\rho L U}{\mu}, \tag{2.5}$$

where ρ is the density of the fluid ($\mathrm{kg/m^2}$), μ is the dynamic viscosity (Ns/m), and L and U are characteristic length and velocity scales respectively. We note that in these simulations, we varied μ while holding $\rho = 1000\,\mathrm{kg/m^2}$, $L = w$ (width of the vein where the inflow is produced), and $U = 5\,\mathrm{m/s}$ (the maximum speed of the parabolic inflow). Simulations were run over several orders of magnitude of Re ranging through $Re \in [0.1, 100]$ on a $[0, 0.5] \times [0, 0.25]$ grid with resolution of $dx = 0.5/1024 = 0.25/512 = dy$.

The result illustrates the flow through the complex geometry as shown in Figs. 2.5 and 2.6. We note that the example of flow through a subset of the veins in a dragonfly wing was chosen to showcase the functionality of the software to capture and digitize intricate complex structures.

Figs. 2.5 and 2.6 compare the flow profiles and pressures generated for three simulations when inflow reaches its steady state through the dragonfly's complex wing vein geometry at $Re = 0.6, 6, 60$.

It is clear there is more flow through the complex morphology with higher pressures for higher Re. Note that when using the immersed boundary method, the entire structure is fully immersed within a fluid, so there is fluid in the region enclosed between veins. Hence the pressure fields in such regions are not physical but instead are artifacts of these regions being within a fluid environment and being enclosed.

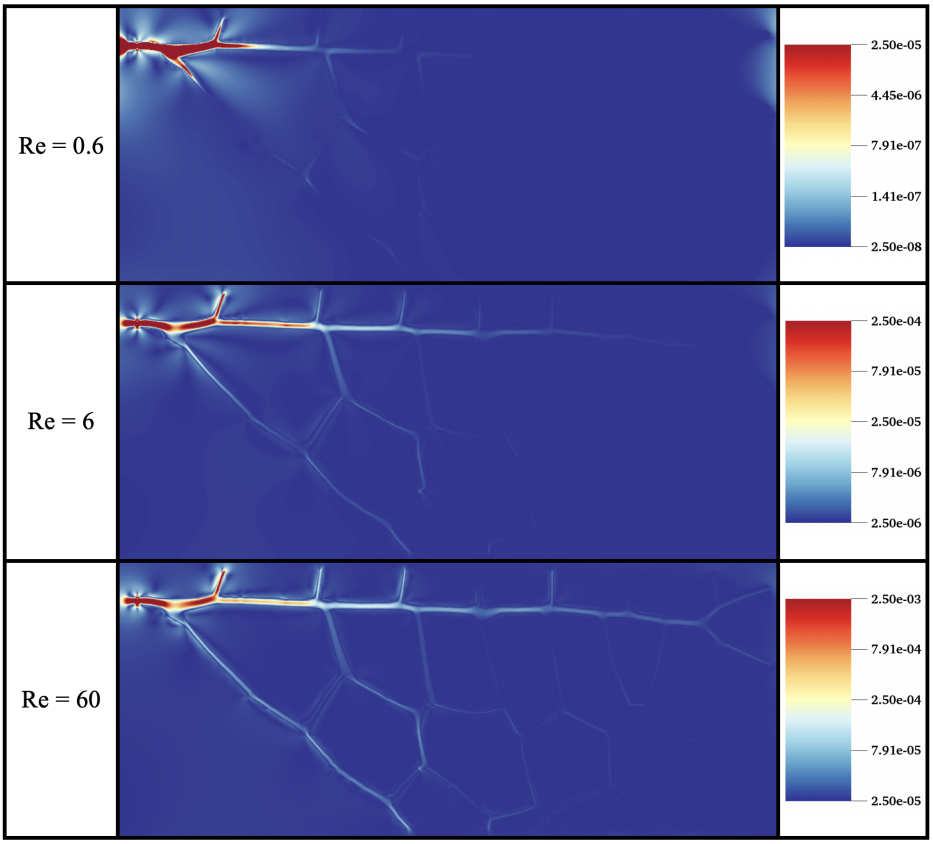

Figure 2.5: Snapshots comparing the magnitude of velocity for different Re, $Re = 0.6, 6, 60$.

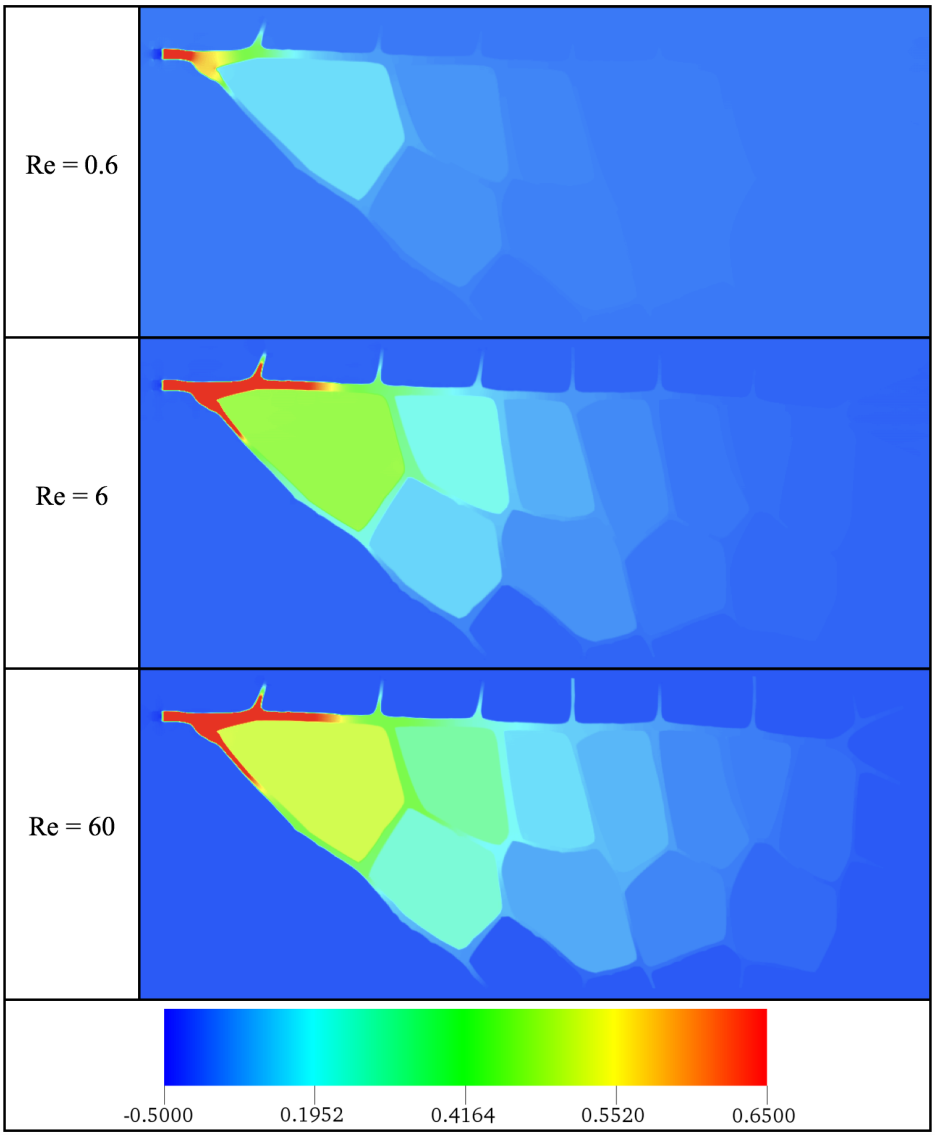

Figure 2.6: Snapshots comparing the pressure for different Re, $Re = 0.6, 6, 60$.

2.3.2 IBAMR: Lymphatic Capillary Example

As another example of how our software can be used, we present a case in which the vessel walls of a junction from a dermal lymphatic capillary are reconstructed from an image. This image is courtesy of Dr. Wenjing Xu from the Kathleen Caron lab (UNC-CH) and was taken from the back region of a wild type mouse embryo. The image was generated with fluorescence microscopy to highlight the lymphatic vessel boundaries as shown in Figure 2.7a. The simulations described below were performed using the immersed boundary method with adaptive mesh refinement (IBAMR) (See Appendix B).

After the contours were extracted using *ContourizeMe*, the ends of the vessels were extended using image software to allow the flow within the vessels to fully develop before reaching the actual vessel geometry. Parabolic outflow was prescribed as a boundary condition to effectively "pull" the fluid into the vessel ends with a maximum velocity of $10^{-5}\,\mathrm{m/s}$. This velocity is consistent with the reported range of observed lymphatic flow velocities, which are as low as 10^{-7} and as high as $10^{-3}\,\mathrm{m/s}$ [82, 83]. Note that the vessel ends were placed at the left domain edge for this purpose. Neumann boundary conditions were used at the right edge of the domain to allow volume conservation (fluid escapes on this side), and periodic boundary conditions were used at the top and bottom of the domain. The vessel was assumed to be nearly rigid over the time scale of a simulation. The Navier-Stokes equations were discretized on a 512×512 grid with 3 levels of mesh refinement and a refinement ratio of 4. The fluid domain size was set to $L = 1.2 \times 10^{-3}\,\mathrm{m}$, where the spatial step size was set to $\Delta x = L/512$. The vessel walls were described using a curvilinear mesh where the distance between immersed boundary points was set to $\Delta s = \Delta x/2$. The time step size was taken as $dt = 5.0 \times 10^{-6}\,\mathrm{s}$. The lymph was parameterized with mass density $\rho = 1000\,\mathrm{kg/m^3}$ and dynamic viscosity $\mu = 10^{-3}\,\mathrm{Ns/m^2}$.

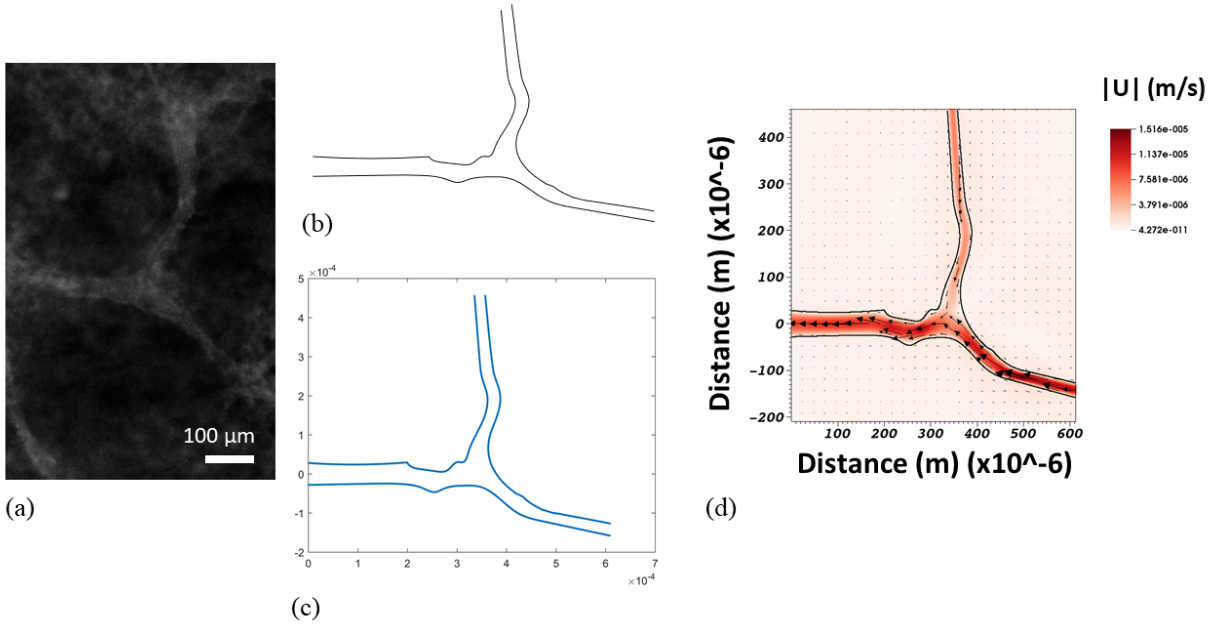

Figure 2.7: (a) Source image for modeling flow through a junction in a lymphatic capillary, courtesy of Dr. Wenjing Xu from the Kathleen Caron lab. (b) SVG rendering of bifurcating vascular structure. (c) Vertex points discretized from SVG file. (d) Colormap of the magnitude of velocity of flow through a bifurcating dermal lymphatic capillary. Note that the vessel ends were artificially extended to allow for fully developed flow within the vessels.

2.3.3 IB2d: Starfish Example

The last example we present is that of flow around a starfish using IB2d. The original JPG image of the starfish was taken from WikiMedia Commons, courtesy of the NOAA Sea Grant Program in the The Coral Kingdom Collection [1]. The SVG file was produced by *ContourizeMe*, and the boundaries were discretized using *MeshmerizeMe* as shown in Figure 2.8. Figures 2.8c, 2.8d, and 2.8f give the discretized geometries for the starfish at resolutions appropriate for an immersed boundary simulation in a square fluid domain with $L_x = L_y = 1$ where the spatial step size within such domain was set to $\Delta x = L_x/128$ (128×128) and $\Delta x = L_x/256$ (256×256), and $\Delta x = L_x/1024$ (1024×1024) respectively. Note that in Figure 2.8f that the starfish's outline is still composed of discretized points. Recall that the average spacing between boundary points is set to one half of the spatial step size, e.g., $\Delta s = 0.5\Delta x$. Once the boundary describing the starfish is discretized at the desired resolution, it is placed inside of a rigid channel, where oscillatory flow will be prescribed to rush past the starfish as shown in Figure 2.8e.

We acknowledge that the starfish geometry is rather complex, and our simulation is relatively coarse. As such, we are likely not resolving the details of the flow very close to the starfish body. We would like to point out, however, that the purpose of this example is to further illustrate *MeshmerizeMe*'s ability to resolve and capture the fine structure detail of an SVG image. Thus, our goal in this example is not necessarily to resolve these fine scale flow structures. This example also highlights that once the geometry has been created for a single starfish, it can be easily altered. For example, this geometry can be copied, translated to different regions of the domain, or rotated since the software provides a parameterized set of points.

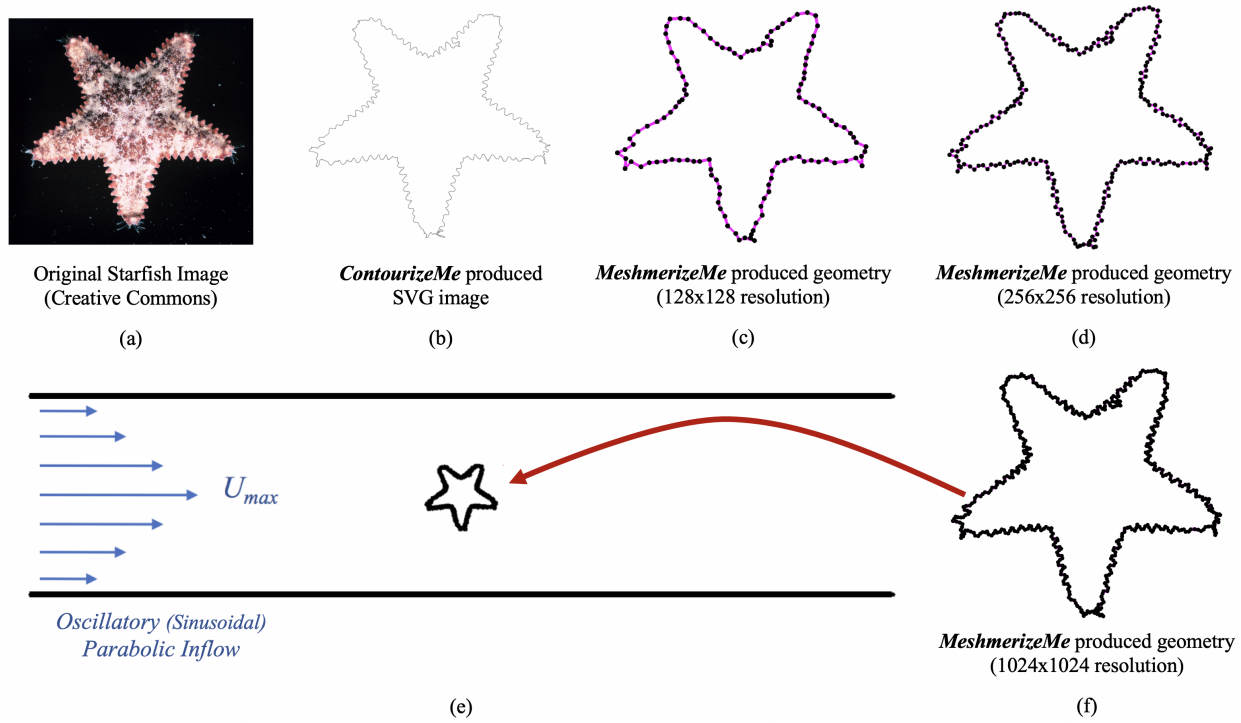

Figure 2.8: (a) Original Starfish image courtesy of NOAA Sea Grant Program [1] (b) SVG image of the starfish generated by *ContourizeMe* script (c), (d), (f) The discretized geometry at grid resolutions of 128×128, 256×256, and 1024×1024 respectively. (e) Computational geometry containing a starfish in a channel with prescribed oscillatory parabolic inflow (see B.1).

We ran a single starfish simulation at $Re = 800$ (see Eq. 2.5), where $L = 0.08\,\text{m}$ (the height of the starfish), $V = 1.0\,\text{m/s}$ (half the maximum oscillatory inflow speed), and ρ and μ are $1000\,\text{kg/m}^2$ and $0.1\,\text{Ns/m}$, respectively. An oscillatory flow condition was used to produce flow past the starfish with frequency of $f = 2\,\text{Hz}$, see B.1. The numerical simulation was performed for a fluid domain with lengths $[0, 1] \times [0, 0.25]$ and a consistent spatial step size in each direction, e.g.,

$dx = 1.0/1024 = 0.25/256 = dy$. Snapshots from the numerical simulation that illustrate a heatmap of vorticity are found in Figure 2.9.

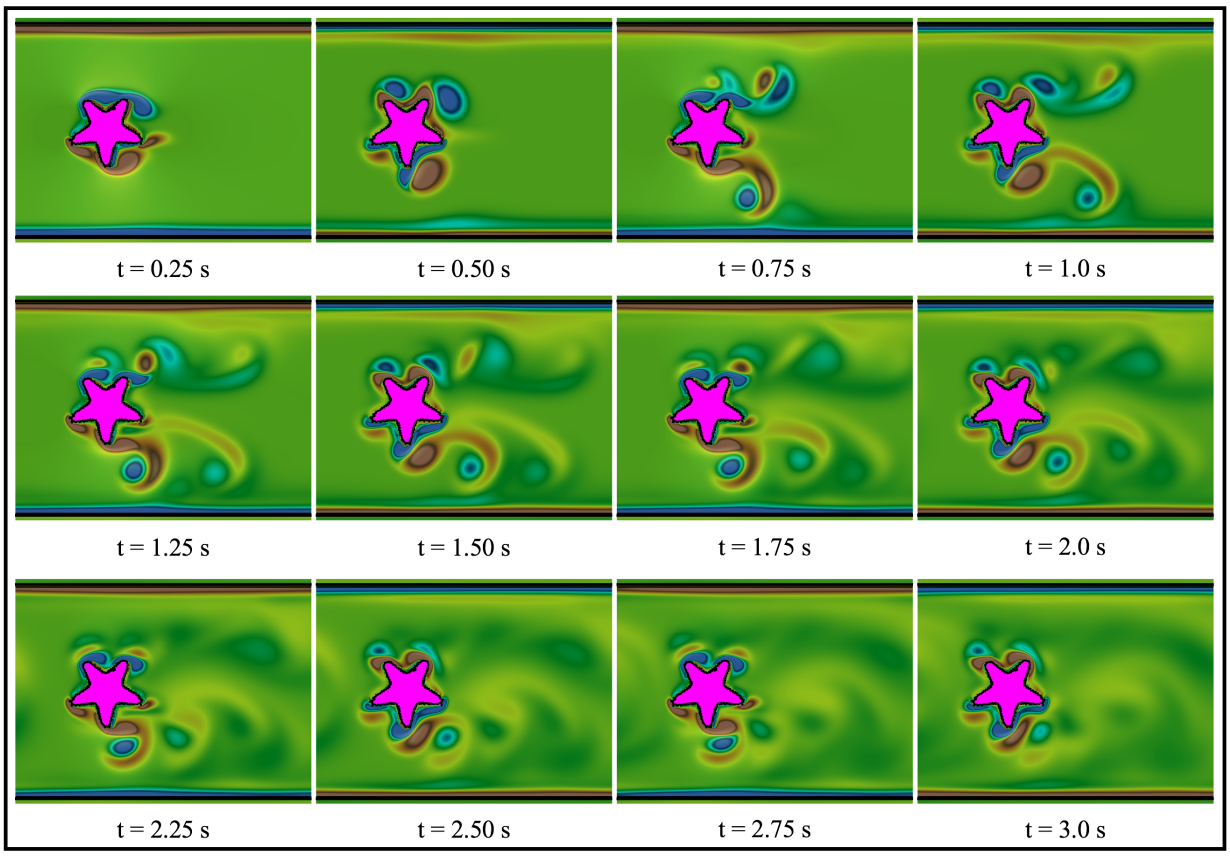

Figure 2.9: Snapshots showing oscillatory flow past a rigid starfish at $Re = 800$ during the first 6 pulsation periods. The background heatmap illustrates vorticity.

Previously, one of the main difficulties in performing this immersed boundary simulation would be the finite difference discretization of the starfish. *MeshmerizeMe* provides a convenient way to do this, without having to manually piece together the geometry either by point-by-point construction or combining user-defined piecewise functions or splines. To demonstrate the versatility of this method, we insert multiple starfish into the channel. Figure 2.10 provides snapshots showing the vorticity during the first pulsation period of oscillatory flow around one, three, or five starfish within a channel. The example for flow around a single starfish can be found in the open source IB2d software's sub-directory, `IB2d/matIB2d/Examples/Example_MeshmerizeMe/Starfish/`.

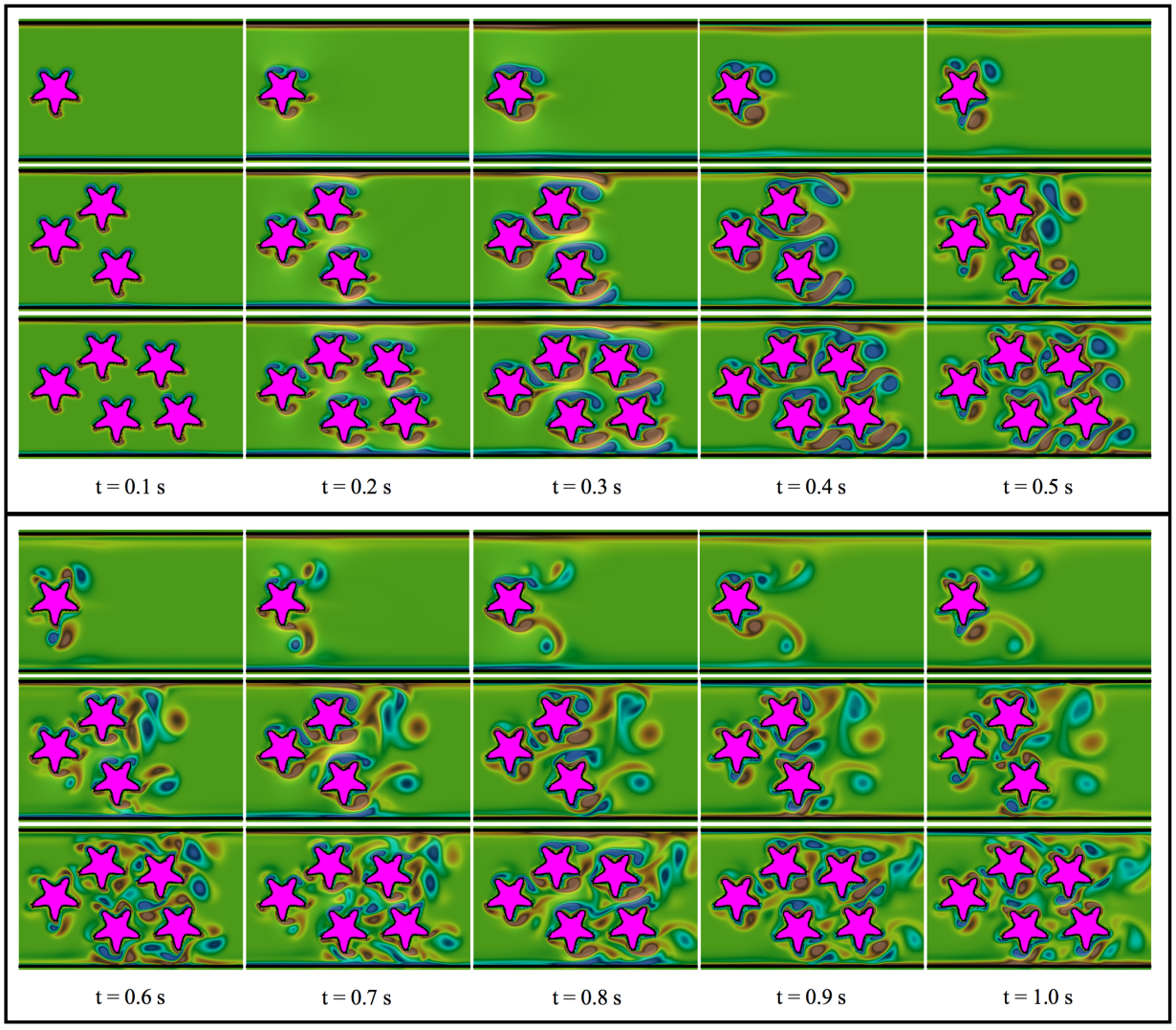

Figure 2.10: Snapshots showing oscillatory flow past 1, 3, or 5 rigid starfish at $Re = 800$ during the first pulsation period. The background colormap illustrates vorticity.

2.4 Discussion

In this chapter, we introduce a software library that will extract edges from images, fit these images with Bézier curves, and discretize the curves into a curvilinear finite difference mesh with nearly constant spacing between points. Such meshes are useful in a variety of mathematical applications, including 2D numerical simulations of fluid-structure interaction (FSI) problems using the immersed boundary method. We present three such applications of the tool used in conjunction with the immersed boundary method including 1) flow of hemolymph in the veins of an insect wing (internal flow), 2) flow of lymph in a mouse lymphatic capillary (internal flow), and 3) flow of water around starfish (external flow). These images were taken with either high resolution digital cameras

or fluorescence microscopy. Prior to this software release paper, the software was successfully applied in another internal flow application of blood flows over the trabeculae in zebrafish embryonic hearts [84]. Here the meshes were created from images taken from an inverted (light) microscope [85]. This illustrates the software's robustness in its ability to construct discretized meshes from various imaging methods. Note that all of the above examples were performed using open-source implementations of the immersed boundary method, either IB2d [39, 38] or IBAMR [65].

While MeshmerizeMe merely provides 2D geometries, its output format can serve as a starting point in the development of 3D models. Commercial CAD software such as Fusion360 allow the import of SVG images, such as those produced by the ContourizeMe script, as sketches. Fusion360's built-in scripts allow the import of a CSV file describing a spline using XYZ coordinates. The latter is easily produced via commandline tools like `tail` and `sed` from the vertex files produced by the MeshmerizeMe script. This likewise imports the curves as a sketch. These imported sketches can then be turned into 3D objects using extrude and rotate commands. From here on, existing 3D meshing tools may be used to produce a mesh suitable for a 3D simulation.

To create simple 3D geometries, the 2D mesh could be extruded manually by adding a third coordinate, and this coordinate could be varied by Δs to obtain a finite difference mesh that describes an outer wall. Similarly, the 2D mesh could be rotated about a central axis to obtain another simple 3D geometry. Sample applications of these simple geometries could include wings or fins with constant cross sections and axisymmetric structures such as tubular hearts, jellyfish, and some worms. The meshes could also be used in other finite difference approaches to FSI problems, including the Method of Regularized Stokeslets [86], the immersed interface method [87], sharp interface methods [88, 89], or the blob projection method [90]. The software library could also be applied in the numerical simulation of other physics problems, including the uptake of particles [91] and electrodiffusion [92]. In future releases of the software library, we plan to add additional functionality that includes the automation for material property model input files, e.g., springs, beams, etc., for the geometry's discretized points.

In addition to use in research, this library may serve as a powerful tool for student research and education, particularly in mathematical modeling at the undergraduate and graduate levels. *MeshmerizeMe* provides students with open source tools that can easily be used to build relatively complicated $2D$ meshes from images. These boundary meshes are easily imported and used in IB2d

and IBAMR, both of which are also open source libraries. One of the coauthors has developed a series of online videos to make the use of this software even easier for students [93]. Both IB2d and *MeshmerizeMe* have been used in the authors' undergraduate and graduate courses, including mathematical modeling, mathematical biology, numerical analysis, and a first year seminar on biological fluid dynamics. Furthermore, the libraries have been successfully used in numerous undergraduate research projects [94, 95] and contemporary locomotion research endeavors [96].

CHAPTER 3

Three-dimensional flexible clap and fling in tiny insect flight

3.1 Introduction

Insects represent an incredible amount of biodiversity and inhabit virtually every biome. It is estimated that there are 5.5 million species of insects [97], and more species of beetles have been described than species of all vertebrates combined. Tiny insects of body lengths on the order of 1 mm or less also inhabit nearly every ecosystem and represent tens to hundreds of thousands of species of thrips [98] and parasitoid wasps [99]. Though not as obvious to the naked eye, their dispersal patterns are significant to the spread of disease [100, 101], prevalence of agricultural pests [102, 103], and strategies for biological control using parasitoid wasps [104].

It has been long known that insect flight differs substantially from the flight of fixed wing aircraft, and the application of traditional quasi-steady state aerodynamics is difficult to apply [105]. Yet, insect flight has fascinated humans for millennia. While the migratory behavior of many bird species has been well studied, significantly less work has documented the seemingly extreme dispersal behaviors of insects. For example, intercontinental travel of insects has been documented across all major oceans [106]. Insects migrate at heights of 100-500m, presumably to take advantage of movement of the air to travel further with limited energy storage [107].

Work over the past two decades using computational fluid dynamics, dynamically scaled robotic insects, and experimental measurements in actual insects has revealed novel mechanisms for lift and thrust generation in insect flight [21, 108, 109, 110]. For example, a stable attached leading edge vortex that allows for the generation of lift at large angles of attack has been observed across scales relevant to insect flight [111, 112]. Lift is also enhanced through wake capture, where the effective velocity of the wing relative to the surrounding air is increased due to the wing moving back through its wake after stroke reversal [113]. The actual reversal of the wings where the wing rotates about the root-to-tip axis has also been shown to enhance circulation and lift generation [114]. For the case of the smallest flying insects, the stable leading edge vortex is important for the generation of

large lift forces [22]. Given significant viscous dissipation, wake capture has less of an effect, and drag becomes much larger than lift, reducing the efficiency of flight [13].

Perhaps to enhance lift generation, the smallest insects in particular have been observed to clap their wings together at the end of upstroke and fling their wings apart at the beginning of downstroke in a mechanism called "clap and fling". This motion increases circulation about the wing and enhances lift across a range of scales and is known as the Weis-Fogh mechanism [24, 28]. At the end of the upstroke, the wings are held very close together (Figure 3.1A). At the beginning of downstroke (the "fling), the wings rotate apart about the trailing edge, forming a "V" shape (Figure 3.1B). Following this fling motion, the wings translate apart in opposite directions. At the beginning of upstroke, the wings rotate and then translate back toward each other. At the end of upstroke, the wings "clap" together and return to their starting position. As a result of the clap motion, two large leading edge vortices are formed at the beginning of the downstroke, and the generation of trailing edge vortices is delayed or reduced (Figure 3.1C). As a result, the circulation about the wings is enhanced for some time during the downstroke. A combination of theoretical, numerical, and experimental work supports that lift is enhanced significantly for Reynolds numbers and kinematics relevant to the smallest insects [14, 26, 29, 30, 31, 115].

Flexibility may also increase lift during clap and fling, in a motion that was originally described as clap and peel [25]. In this case, the wings are peeled apart from the leading to the trailing edge and curve along the wing chord. One can approximate this motion as a rigid 'flat peel' where the wings unzip. Using this simplification, the circulation in this case may be up to 2.6 times greater than that predicted by Lighthill [28]. Two-dimensional numerical simulations of this motion also support that lift can be augmented when the wings are flexible enough to generate a peel motion but not so flexible that lift is reduced due to the wing reconfiguration [26]. To the authors' knowledge, the effect of flexibility in the spanwise direction on lift generation has not been considered.

Interestingly, very large forces can also be required to fling the wings apart at the low Re relevant to the smallest insects [14]. The studies that consider large drag forces have focused on either rigid insect wings or wings that only bend along the chord, and it has been shown that flexibility in the chordwise direction can reduce drag [26]. In addition, the bristled wing structure that characterizes many small insects can aid in drag reduction by acting as a porous structure [23, 27, 116]. Furthermore, a recent computational study has shown that for the parasitoid wasp

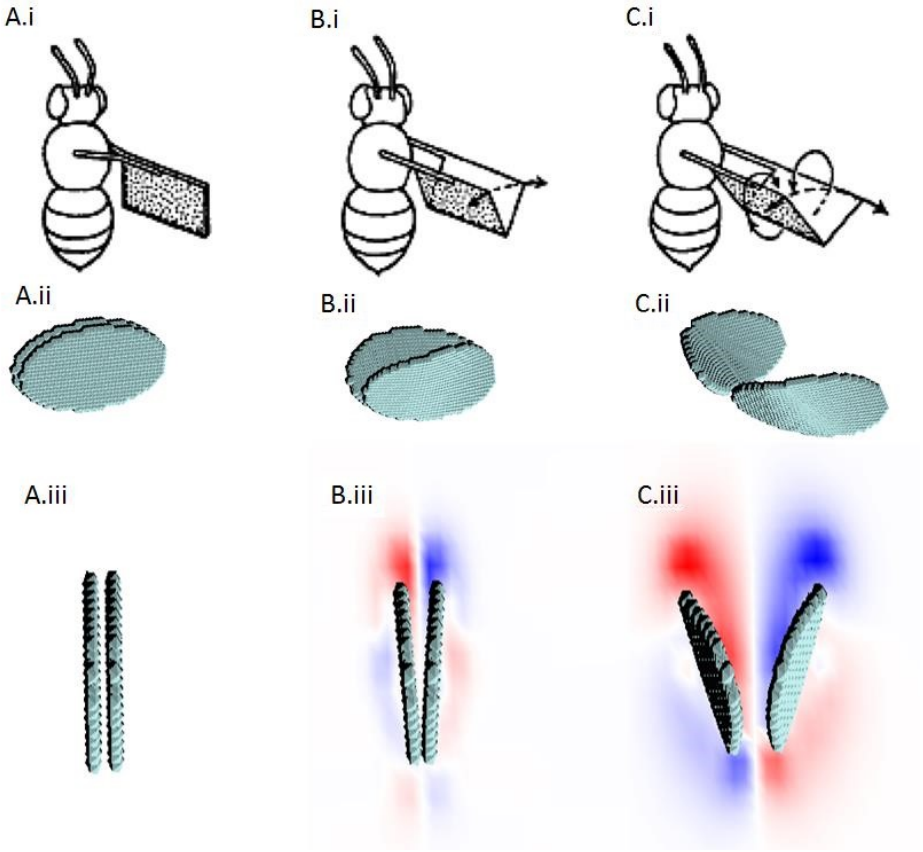

Figure 3.1: Clap and fling in 3D. (A) At the beginning of a wing beat the wings are held close together. (B) the wings then rotate apart, and (C) form two large leading edge vortices. (i) Schematic of the fling (2), (ii) a side view of two immersed boundary wings engaged in fling, and (iii) a front-view of two immersed boundary wings engaged in fling.

Encarsia formosa, an upward motion during the fling can significantly reduce drag while maintaining lift [115]. Another possible mechanism for drag reduction is flexibility in the spanwise direction, but to the authors' knowledge this mechanism has not been well studied. Flexibility has, in general, been demonstrated to play an important role in drag reduction due to the reconfiguration of the organism or appendage [117, 118]. Consequently, we expect that flexibility in the spanwise direction would play an important role in wing performance during the fling. This reduction in drag may be particularly useful for the lowest Reynolds number flyers, especially if the wings are not so flexible as to significantly reduce the lift forces produced.

In this paper, a 3D model of two rigid or two flexible elliptical wings performing an idealized clap and fling stroke was numerically simulated using a hybrid finite element version of the immersed boundary method [119]. The Reynolds number was varied to consider the case of the smallest parasitoid wasp to a fruit fly (e.g. $Re = 4 - 128$). The elastic modulus of the wings was tuned such that single wing deformations were small while deformations generated during the fling produced a peel-like motion. The force required to fling the wings apart was quantified over this range of Re for both a pair of wings and a single wing performing the same motion. The resulting flow fields generated and, in particular, the formation and stability of the leading and trailing edge vortices were related to the instantaneous lift and drag experienced by the wings.

3.2 Methods

3.2.1 Immersed Boundary Method

Numerous fluid-structure interaction problems in biology have been investigated using the immersed boundary method [4, 120, 121]. Some examples include jellyfish and fish swimming [122, 123], crayfish paddling [33, 67], insect flight [22], and flow through heart valves [124, 125]. The advantage of this method is that complex geometries, e.g., internal or external morphology, can easily be handled without the need to generate matching grids for both the fluid and the structure.

Below we present the mathematical formulation of the immersed boundary method. We begin with the equations of motion for a three-dimensional, incompressible viscous fluid in Eulerian form,

$$\rho\left(\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}+\mathbf{u}(\mathbf{x},t)\cdot\nabla\mathbf{u}(\mathbf{x},t)\right)=-\nabla p(\mathbf{x},t)+\mu\nabla^2\mathbf{u}(\mathbf{x},t)+\mathbf{f}(\mathbf{x},t) \tag{3.1}$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t)=0, \tag{3.2}$$

where $\mathbf{u}(\mathbf{x},t)=(u(\mathbf{x},t),v(\mathbf{x},t),w(\mathbf{x},t))$ is the fluid velocity at the time t and spatial location $\mathbf{x}=(\mathbf{x},\mathbf{y},\mathbf{z})$, $p(\mathbf{x},t)$ is the pressure, and $\mathbf{f}(\mathbf{x},t)$ is the force per unit volume applied to the fluid by the immersed boundary. The system of equations is known as the Navier-Stokes equations, and Eq.(3.1) is equivalent to the conservation of momentum for a fluid while Eq.(3.2) is the incompressibility condition.

The fundamental feature of the immersed boundary method are the interaction equations between the fluid and the immersed, elastic structure. The equations describing the elastic structure are written in Lagrangian form. Let $\mathbf{X}=(X,Y,Z)\in U$ denote the Lagrangian material coordinates of the structure, where U is the Lagrangian coordinate domain. The Lagrangian material coordinates are mapped to the physical position of material point $\mathbf{X}$ at time t by $\boldsymbol{\chi}(\mathbf{X},t)=(\chi_{\mathrm{x}}(\mathbf{X},t),\chi_{\mathrm{y}}(\mathbf{X},t),\chi_{\mathrm{z}}(\mathbf{X},t))\in\Omega$. The physical region occupied by the elastic structure at time t is then given as $\boldsymbol{\chi}(U,t)\subset\Omega$.

The immersed boundary equations describing the interaction between the fluid and the elastic structure are then given as

$$\mathbf{f}(\mathbf{x},t)=\int_U \mathbf{F}(\mathbf{X},t),\delta(\mathbf{x}-\boldsymbol{\chi}(\mathbf{X},t))\,\mathrm{d}\mathbf{X} \tag{3.3}$$

$$\int_U \mathbf{F}(\mathbf{X},t)\cdot\mathbf{V}(\mathbf{X})\,\mathrm{d}\mathbf{X}=-\int_U \mathbb{P}(\mathbf{X},t):\nabla_{\mathbf{X}}\mathbf{V}(\mathbf{X})\,\mathrm{d}\mathbf{X}+\int_U \mathbf{G}(\mathbf{X},t)\cdot\mathbf{V}\,\mathrm{d}\mathbf{X} \tag{3.4}$$

$$\frac{\partial\boldsymbol{\chi}(\mathbf{X},t)}{\partial t}=\int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\boldsymbol{\chi}(\mathbf{X},t))\,\mathrm{d}\mathbf{x}. \tag{3.5}$$

where $\mathbf{F}(\mathbf{X},t)$ is the Lagrangian force density and $\delta(\mathbf{x})$ is the Dirac delta function that acts as the kernel of an integral transformation in Eqs. (3.3) and (3.5). Note that $\mathbf{F}$ is defined in terms of the first Piola-Kirchhoff stress tensor, $\mathbb{P}$, in Eq. (3.4) and an external force acting on the body, $\mathbf{G}(\mathbf{X},t)$. This definition uses a weak formulation where $\mathbf{V}(\mathbf{X})$ is an arbitrary Lagrangian test function.

A Neo-Hookean material model is used to describe the material properties of the wing,

$$\mathbb{P}=\eta\mathbb{F}+(\lambda\log(J)-\eta)\mathbb{F}^{-T} \tag{3.6}$$

where $\mathbb{F} = \frac{\partial \boldsymbol{\chi}}{\partial \mathbf{X}}$ is the deformation gradient of the mesh, J is the Jacobian of $\mathbb{F}$, η is the shear modulus, and λ is the bulk modulus. The shear and bulk moduli are defined using the following equations,

$$\eta = \frac{E}{2(1+\nu)} \tag{3.7}$$

and

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{3.8}$$

where E is the Young's modulus and ν is the Poisson ratio.

To drive the motion of either the entire wing or the base of the wing, a target force, $\mathbf{G}(\mathbf{X}, t)$, is used as follows,

$$\mathbf{G}(\mathbf{X}, t) = \kappa(\boldsymbol{\chi}(\mathbf{X}, 0) - \boldsymbol{\chi}(\mathbf{X}, t)), \tag{3.9}$$

where κ is a spring constant that is adjusted to minimize deviations from the preferred configuration (see more details below).

The general numerical implementation of the immersed boundary method is as follows for each time step:

1. Calculate the force that the boundary applies to the fluid.
2. Spread the force from the Lagrangian grid describing the position of the boundaries to the Cartesian grid used to solve the fluid equations (Eq. 3.4).
3. Solve the Navier-Stokes equations (Eq. 3.1 and 3.2).
4. Use the new velocity field to update the position of the boundary. The boundary is moved at the local fluid velocity to enforce the no-slip condition (Eq. 3.5).

A distributed-memory parallel implementation of the immersed boundary method (IBAMR) with Cartesian grid adaptive mesh refinement was used to simulate this fully-coupled fluid-structure interaction problem [126]. Four levels of refinement were used to discretize the Eulerian equations with a refinement ratio of 4 between levels. Regions of the fluid that contain the wing or vorticity above a threshold were discretized at the highest refinement. The effective resolution of the finest level of the grid corresponds to that of a uniform $512 \times 512 \times 512$ discretization. The boundary conditions are set to no-slip ($u = 0$) on all sides of the computational domain.

3.2.2 Dimensionless Parameters

To make comparisons across scale, dimensionless parameters are used to define the relevant length scales, time scale, forces, and material properties. The wing chord, w_{chord}, was defined as the characteristic length, L', and the wing span, w_{span}, was set equal to $2w_{chord}$. The characteristic velocity, U', was set to the average wing tip velocity calculated as $\frac{4\pi w_{span} f}{3}$, which uses a stroke amplitude of $2\pi/3$ and a flapping frequency of f. The characteristic time, t', is set equal to the time it takes to complete one flap. The Re is set equal to the frequency based Reynolds number, which is given as

$$Re = \frac{2\pi\rho w_{span}^2 f}{3\mu}. \tag{3.10}$$

Lift and drag forces are calculated as follows, noting that lift is defined as the force in the vertical direction and drag as the total force in the horizontal direction:

$$\begin{aligned} F_L &= F_z \\ F_D &= \sqrt{F_x^2 + F_y^2} \end{aligned} \tag{3.11}$$

Note that since the stroke plane is horizontal, lift is the force perpendicular to the stroke plane, and drag is the force tangential to the stroke plane. The forces are non-dimensionalized using the following equation,

$$F'_X = \frac{2F_X}{\rho S U^2}, \tag{3.12}$$

where F'_X is the normalized force , X represents lift or drag, F is the dimensional force, and S is the reference area. We will set S equal to the dimensional area of the wing, and U as the average dimensional wingtip velocity.

The wing's elastic modulus is nondimensionalized using w_{chord} as the characteristic length and the flapping frequency, f such that the dimensionless elastic modulus, η' is given as

$$\eta' = \frac{\eta}{\rho w_{chord}^2 f^2} \tag{3.13}$$

where ν is the dimensional elastic modulus. The vorticity ($\omega = \nabla \times u$) is nondimensionalized with

respect to the flapping frequency such that

$$\omega' = \omega / f. \tag{3.14}$$

The movement of the wings was driven by target forces that were imposed to move the wing or part of the wing with a preferred motion. This method has previously been used when a preferred position must be enforced [13, 127]. Essentially, an external force is applied that is proportional to the difference between the preferred position of the boundary and its actual position. The difference between these two positions is controlled with a spring constant, k_{spring}. The dimensionless spring constant is then given as

$$k'_{spring} = \frac{k_{spring}}{\rho(\frac{4\pi w_{span} f}{3})^2 w_{chord}}. \tag{3.15}$$

3.2.3 The 3D Model Wing and its Motion

Two ellipsoid wings were constructed, and a tetrahedral mesh with an element size of $0.04725 w_{chord}$ was generated using Trelis (2020, Csimsoft) for use in immersed boundary simulations. The wings were initialized in the x, z plane and parallel to each other at a 90 deg angle of attack, as would be the case for the initiation of fling. The major and minor axes are half of w_{span} and w_{chord} respectively, for a wing area $A_{wing} = \frac{\pi}{8} w_{span}^2$. The distance between the middle of the wings at the initiation of fling was $0.4 w_{chord}$. The base of the wings were $0.25 w_{chord}$ from the center of rotation.

For the case of rigid wings, the target forces were applied along the entire wing so that the bending of the wing was less than 1/100 of the wing span. For the case of flexible wings, the target forces were applied to a region near the root of the wing that was 1/8 of the wing span. The remainder of the wing was allowed to freely deform as a result of the fluid-structure interaction.

The equations of motion for the preferred movement of the wing that moves with a positive sweep angle are as follows:

$$\sigma = 0.5A(1 - \cos(2\pi f t)) \tag{3.16}$$

$$\alpha = \alpha_0 + B \sin(2\pi f t) \tag{3.17}$$

where α is the angle of attack, σ is the sweep angle, $\alpha_0 = \pi/2$ is the initial angle of attack at the

beginning of fling, f is the flapping frequency, $A = 2\pi/3$ is the sweep amplitude, and $B = \pi/4$ is the maximum change in the angle of attack from α_0 during the wing stroke. During the downstroke, the wing is rotated about the trailing edge. During the upstroke, the wing is rotated about the leading edge. These kinematics have been used in previous 2D simulations to prevent collision of the wings [14] and to appropriately model the peel and reverse peel of the wings as described in the literature [26, 128, 25].

3.2.4 Simulation Parameters

Table 3.1: The numerical parameters. Lengths are nondimensionalized by w_{chord} and time is nondimensionalized by the flapping period.

Dimensionless Parameter	Symbol	Value
Chord Length	w'_{chord}	1
Span Length	w'_{span}	2
Domain Size	D'	25
Spatial Step Size	dx'	0.04725
Time Step Size	dt'	10^{-5}
Elastic Modulus	η'	$3.9 \times 10^4 - 1.25 \times 10^6$
Target Spring Stiffness	k'_{spring}	2.22×10^{11}
Reynolds number	Re	4-128
Final Simulation Time	T'	4
Flapping Frequency	f'	1

The domain of the simulation was a periodic cube with a side length of $25w_{chord}$. A convergence study to ensure the spatial accuracy of the model and a sufficient domain size was performed previously [22]; the lift and drag generated by a flapping elliptical wing for the range of Re considered in the current study were compared using 256^3 and 512^3 grids. The error in these forces was less than 3% at $Re = 10$. Similarly, a convergence study was performed for 2D immersed boundary simulations of the clap and fling motion at $Re = 8$ using a 512^3 grid [14]. Finally, the IBAMR software library has been previously validated at similar Re using a $512^2/512^3$ grid for shells, cavity flow with a soft neo-Hookean disk, and flow past a cylinder [119].

3.3 Results

3.3.1 Flow Fields

The case of two wings at $Re = 8$ and $\eta' = 6.25 \times 10^5$ was selected as a basis for comparison because it is representative of thrips and some parasitoid wasps. There is non-negligible wing

deformation during the fling, but bending is minimal during the stroke, as has been observed in tiny insects [26]. Figure 3.2 shows snapshots of the wings during the initial stroke every 0.05 units of dimensionless time. Snapshots i-x show the downstroke and xi-xx show the upstroke. The most significant deformations occur as the wings fling apart (ii-iv). During the rest of the downstroke, the deformations are small (v-vii). Some small deformations are then observed during wing reversal (ix-x) and again during the clap (xviii-xx).

Figures 3.3 and 3.4 show the velocity field and vorticity in a 2D slice from a side view and the back view. In Figure 3.3, the 2D slice is taken in a plane parallel to the xz-plane where $y = 0.05$. This plane then cuts through the middle of the left (closest) wing when the sweep angle is 45°. The color map showing the y-component of the vorticity, ω_y is translucent such that both wings are visible. The velocity arrows are proportional to the magnitude of the velocity. In Figure 3.4, a 2D slice is taken along a plane parallel to the yz-plane where $x = 0.05$ This plane then cuts through the middle of the wings when the sweep angle is 0° (at the beginning of fling). The color map shows the x-component of vorticity, ω_x, and the arrows show the direction and magnitude of the velocity.

During the downstroke in Figure 3.3ii-iv, the stable, attached leading edge vortex forms and corresponds to the yellow region of the vorticity color map. An oppositely-spinning stable, attached trailing edge vortex (TEV) forms along the trailing edge of the wing and corresponds to the purple region of the vorticity map. This pattern of stable, attached LEV and TEV corresponds to previous computational and experimental results using physical models at this Re [22]. After stroke reversal (vi-ix), a new leading edge vortex forms (purple region in the vorticity map), and a stable, attached trailing edge vortex also forms (yellow region). During the next stroke cycle, these vortices quickly dissipate, and new LEV and TEV form.

Figure 3.4 reveals the initial formation of the LEV and TEV during the fling (i-iv). Since the wings are not touching, both the LEV and TEV form during wing rotation. Note that when the wings are perfectly clapped together, the TEV does not form initially [29, 28]. During wing rotation, the LEV is stronger and enhances the lift generated, as has been reported in previous 2D computational studies [14, 26]. During the clap (vii-ix), one can observe the stable, attached, LEV and TEV on each wing. At the end of the upstroke, the wings are squeezed together which generates a downward jet that also enhances lift relative to the case of one wing.

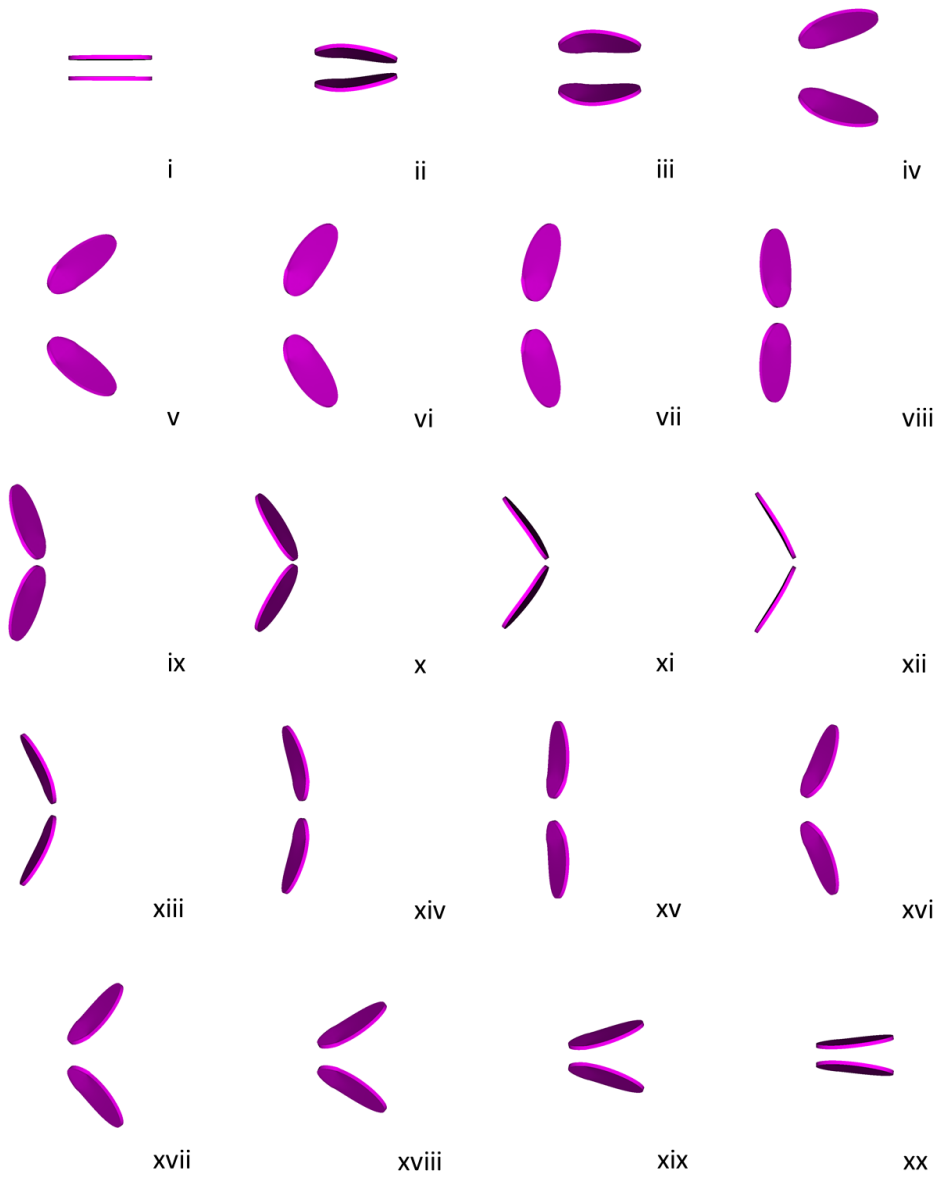

Figure 3.2: Top view of two wings performing a clap and fling at $Re = 8$ with $\eta' = 6.25 \times 10^5$. Snapshots were taken during the first stroke every 0.05 units of dimensionless time. i-x represent the downstroke, and xi-xx represent the upstroke. The fling is performed during i=iv, and the clap occurs from xviii-xx. Deformations are visible during fling (ii-iv), wing reversal (ix-x), and clap (xviii-xx).

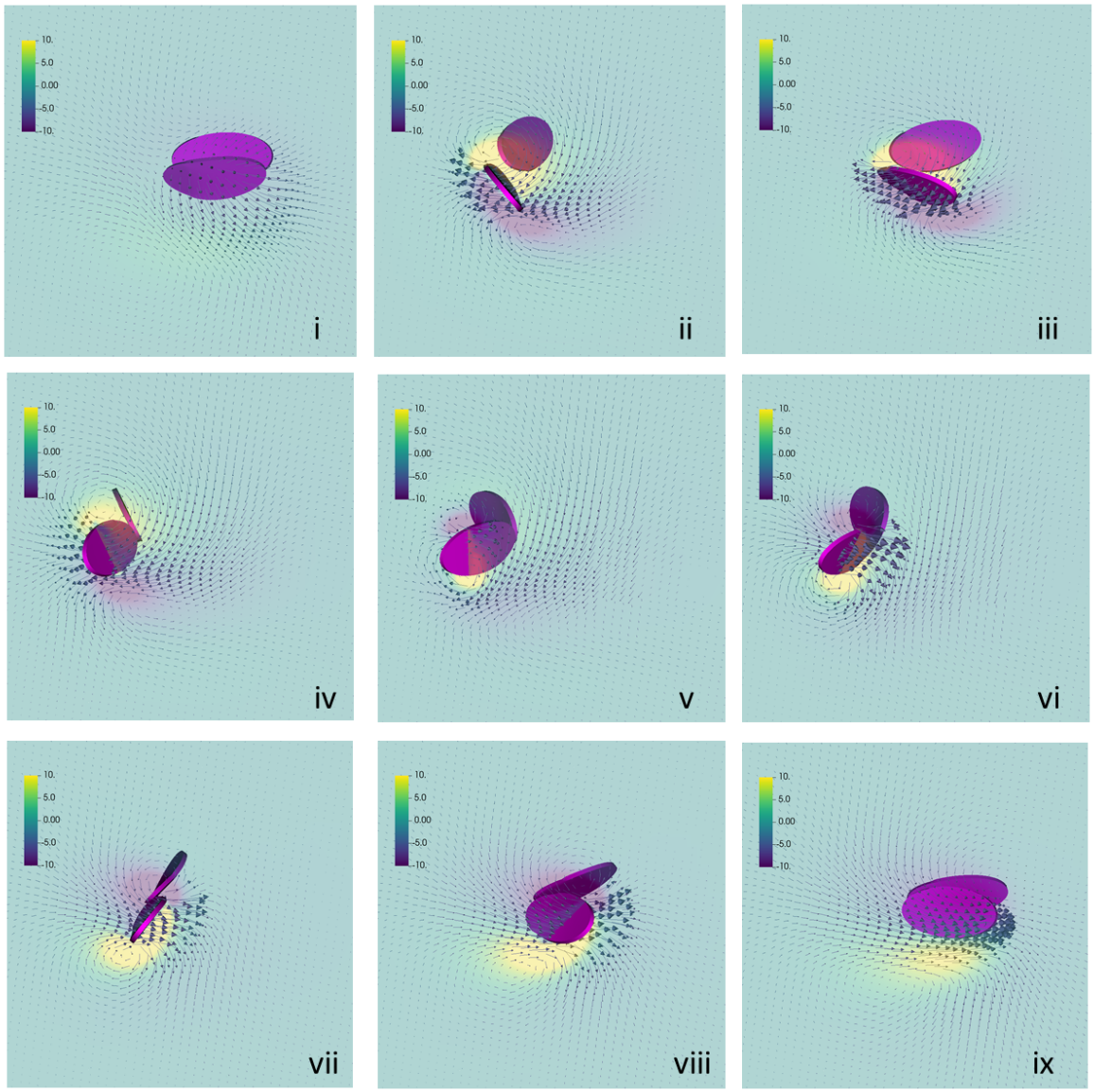

Figure 3.3: Vorticity and velocity fields in a slice parallel to the xz-plane for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at times $t' = 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8$, and 2.9. The color map shows ω_y.

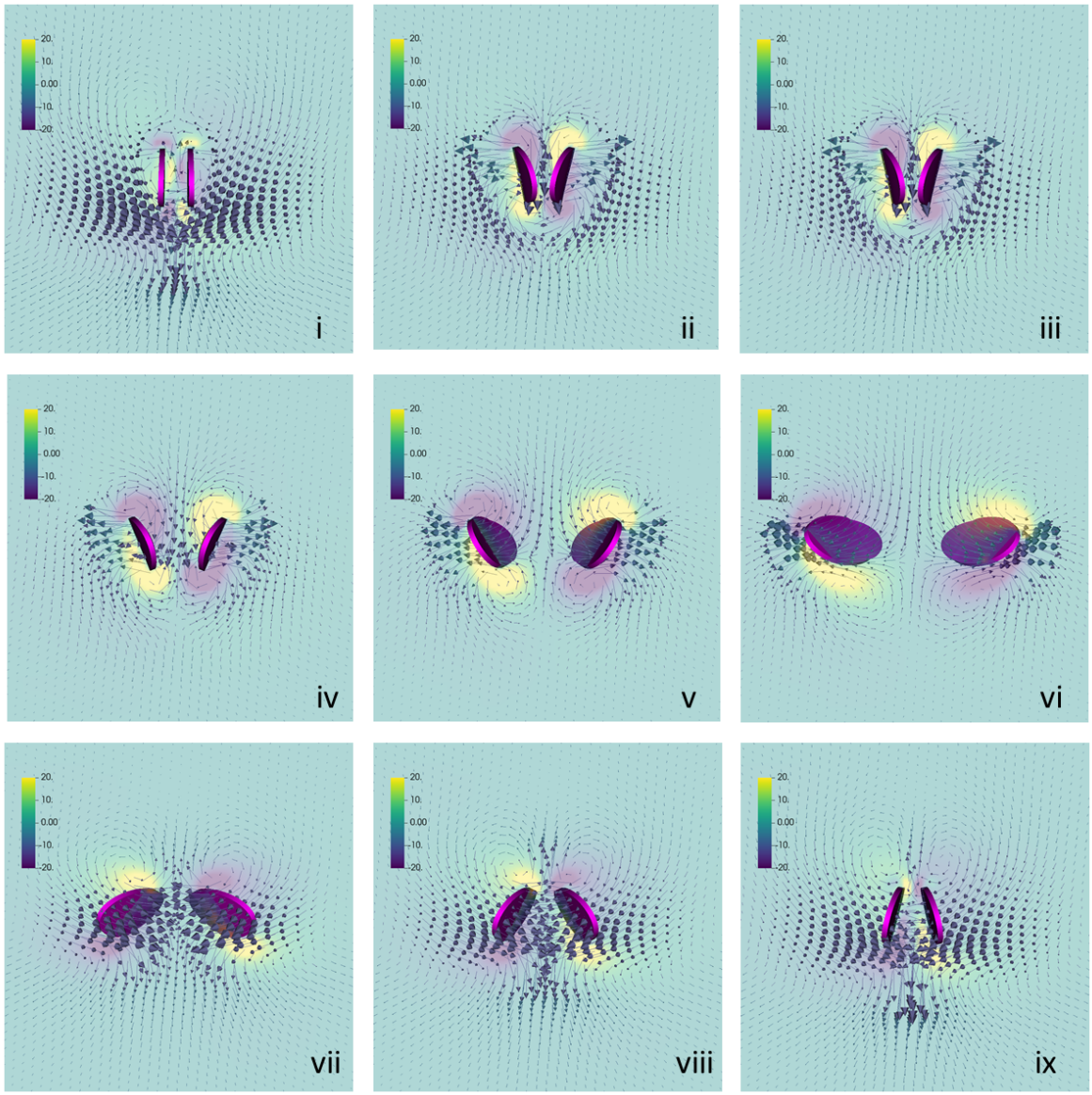

Figure 3.4: Vorticity and velocity fields in a slice parallel to the yz-plane for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at times $t' = 2.0, 2.05, 2.1, 2.15, 2.2, 2.25, 2.9, 2.95$, and 3.0. The color map shows ω_x.

3.3.2 Effect of Re

To quantify the effect of scale, numerical simulations were performed for two wings performing a clap and fling and one wing moving with the same motion for Re = 4, 8, 16, 32, 64, and 128. This range represents flight for the smallest parasitoids ($Re = 4$) to fruit flies ($Re = 128$) with $\eta' = 6.25 \times 10^5$. Figure 3.5 shows the dimensionless force vs. dimensionless time for one wing (A, B) and two wings (C, D) at Re = 4, 18, 16 for the third stroke. For this choice of Re, dimensionless forces are higher for lower Re. In the case of lift (A, C), each peak corresponds to the fling during the downstroke and wing rotation during the upstroke. Lift is higher for two wings than for one, and peak lift decreases after the initial downstroke due to the deformation of the wings. For the case of drag (B, D), the minima correspond to forces generated during the fling and the maxima correspond to forces generated during the clap. The difference in peak drag for the case of one wing vs. two wings is small, but larger forces are generated for longer periods of time when there are two wings.

Figure 3.6 shows the average (A, C) and peak (B, D) dimensionless forces as a function of Re for the cases of one wing and a pair of wings. Note that the average and peak are taken during the third stroke, after start-up transients. Dimensionless forces increase for $Re < 16$, and dimensionless drag increases substantially for $Re < 20$. Dimensionless forces plateau for $Re > 60$, although larger peak drag is observed at higher Re for one wing during wing rotation due to vortex-wing interaction. In general, average and peak forces are higher for the case of two wings vs. one wing moving with the same motion. Average drag is roughly 25% higher at $Re = 4$ for the case of two wings, and peak drag is approximately 40% higher for two wings. This is substantially smaller than the order of magnitude difference observed for rigid wings in 2D [14], and is likely because spanwise flow helps to fill the space between the two wings performing a fling.

Figure 3.7 shows the velocity vectors and vorticity color map for two wings performing a fling at Re = 4, 8, 16, 32, 64, and 128 at $t' = 2.25$. The 2D slice showing velocity and vorticity is taken in a plane parallel to the xz-plane where $y = 0.05$. The color map shows the y-component of the vorticity, ω_y. In all cases, the LEV is stable and attached to the wing (yellow region on the upper surface). For $Re = 4, 8$ (i, ii), the trailing edge vortex is clearly visible and does not separate from the wing. As the Re increases to 16 and 32 (iii, iv), the TEV is advected downstream. For Re = 64, 128 (v, vi), the TEV clearly separates from the wing. In addition, previously shed vortices persist from early strokes and interact with the wing and its wake. These results are consistent with earlier

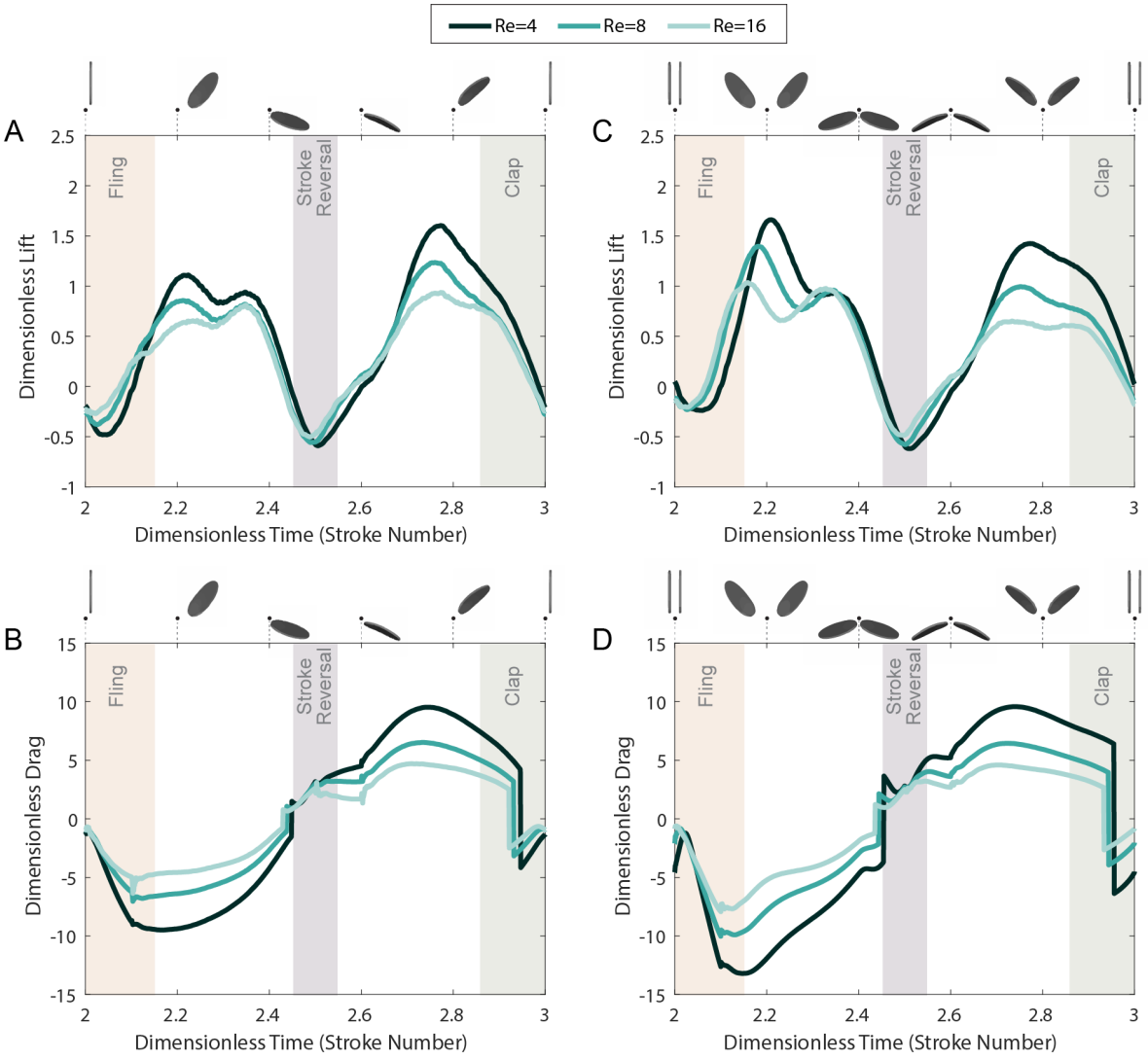

Figure 3.5: Dimensionless force vs. dimensionless time for one wing (A, B) and a pair of wings (C, D) performing clap and fling at $Re = 4$, 8, and 16. Forces are shown for the third stroke.

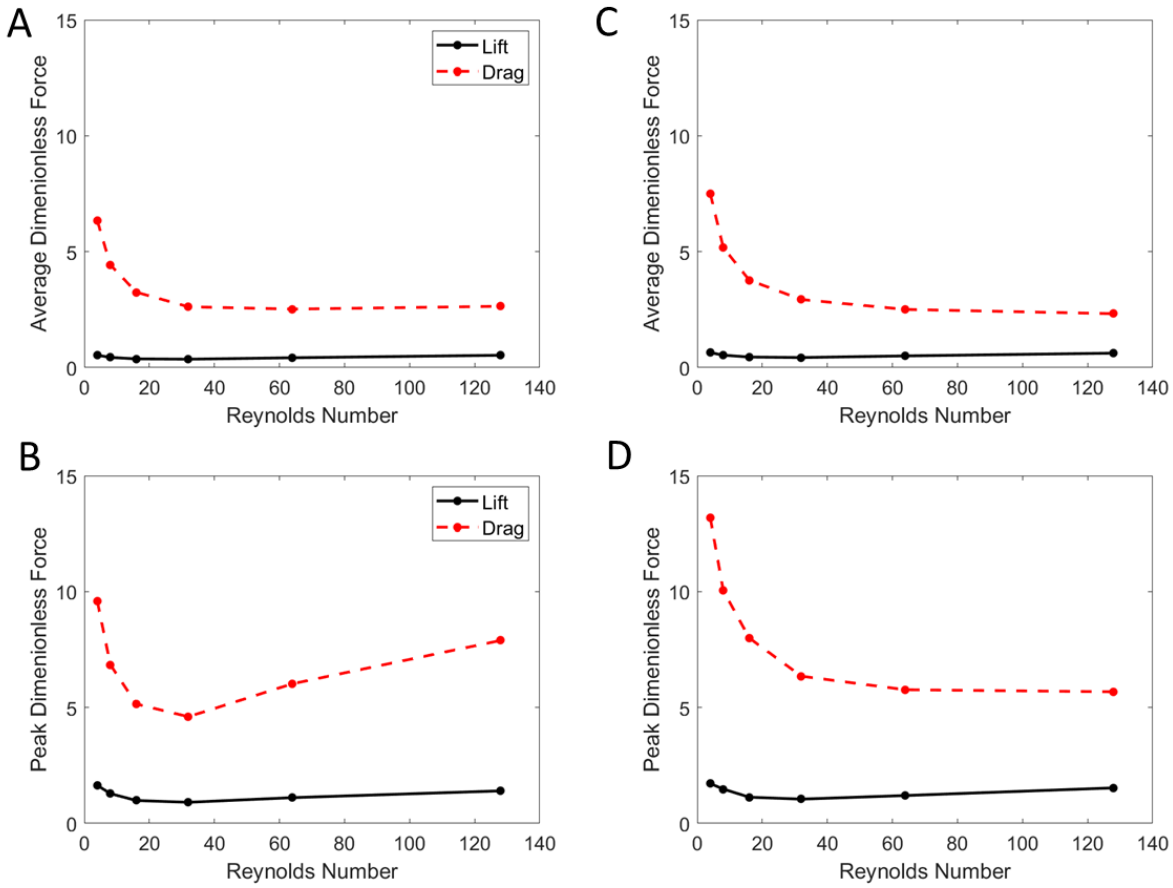

Figure 3.6: Average and peak dimensionless forces as a function of Re for one wing (A, B) and two wings (C, D). Lift is shown in black and drag in red.

2D simulations for this range of Re [13, 14].

3.3.3 Effect of Flexibility

To quantify the effect of flexibility, numerical simulations were performed for two wings performing a clap and fling for $Re = 8$ and η' ranging from $3.9 \times 10^4 - 1.25 \times 10^6$. This represents the case where the wing undergoes significant deformation throughout the stroke to the case where the wing is nearly rigid. Wing flexibilities are reported in relation to the reference case, $\eta_r = 6.25 \times 10^5$. Figure 3.8 shows the dimensionless forces generated for this range of flexibilities. Figure 3.8 A, B shows the dimensionless lift and drag vs. time for $\eta' = \eta_r/16, \eta_r/4, 2\eta_r$. Lift forces are larger for stiffer wings. Interestingly, the most flexible wings that undergo large deformations ($\eta' = \eta_r/16$) experience large negative lift shortly after wing reversal, resulting in low values of average lift generated during the stroke. In terms of drag, large forces are required to fling stiffer wings apart ($\eta' = 2\eta_r$). Since the wings are nearly parallel at this part of the stroke, the forces cancel and are indicative of a relatively large loss of power during flight.

Figure 3.8 C-E shows the average and peak forces generated by two wings performing clap and fling vs. the dimensionless elastic modulus taken during the third stroke. In Figure 3.8 C, both the average dimensionless lift and drag decrease rapidly for $\eta' < 3.125 \times 10^5$. The average dimensionless forces plateau for $\eta' > 3.125 \times 10^5$ as the deformations of the wing are relatively small. Peak forces also decrease for lower values of the dimensionless stiffness. The peak drag does continue to increase for $\eta' > 3.125 \times 10^5$ due to large transient forces generated during wing rotation. Figure 3.8 E shows the average lift over drag for varying dimensionless stiffnesses. This is a crude metric for aerodynamic efficiency if one assumes that the primary goal of flight in these tiny insects is to stay aloft and be carried by the wind. Lift/Drag decreases dramatically for $\eta' < 3.125 \times 10^5$ and plateaus for $\eta' > 3.125 \times 10^5$. Although both lift and drag decrease with decreasing η', lift decreases faster. This suggests that there is a aerodynamic cost when wing flexibility is below some threshold. Above this threshold, aerodynamic gains for stiffer wings are negligible.

Figure 3.9 shows the velocity vectors and vorticity color map for two wings performing a fling at $Re = 8$ and $\eta' = \eta_r/16, \eta_r/8, \eta_r/4, \eta_r/2, \eta_r, 2\eta_r$. Snapshots were taken at $t' = 3.25$, which corresponds to halfway through the third downstroke. The 2D slice showing velocity and vorticity is taken in a plane parallel to the xz-plane where $y = 0.05$. The color map shows the y-component of the vorticity, ω_y. Large deformations of the wing during translation are observed for $\eta' < 3.125 \times 10^5$

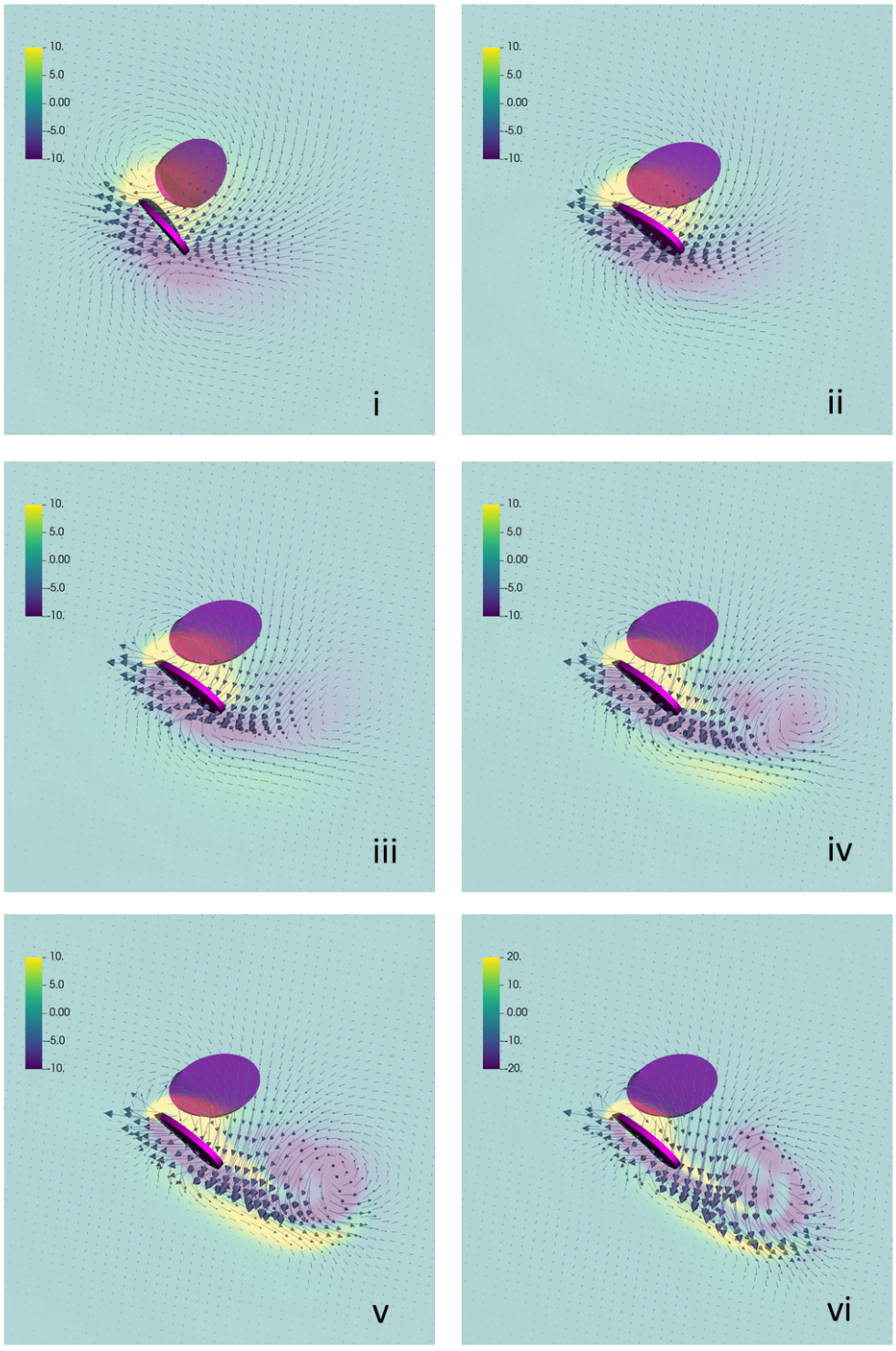

Figure 3.7: Vorticity and velocity fields in a slice for two wings performing a clap and fling at $Re = 4, 8, 16, 32, 64, 128$. The 2D slice showing the flow characteristics is parallel to the xz-plane. Snapshots were taken during the third stroke at time $t' = 2.25$.

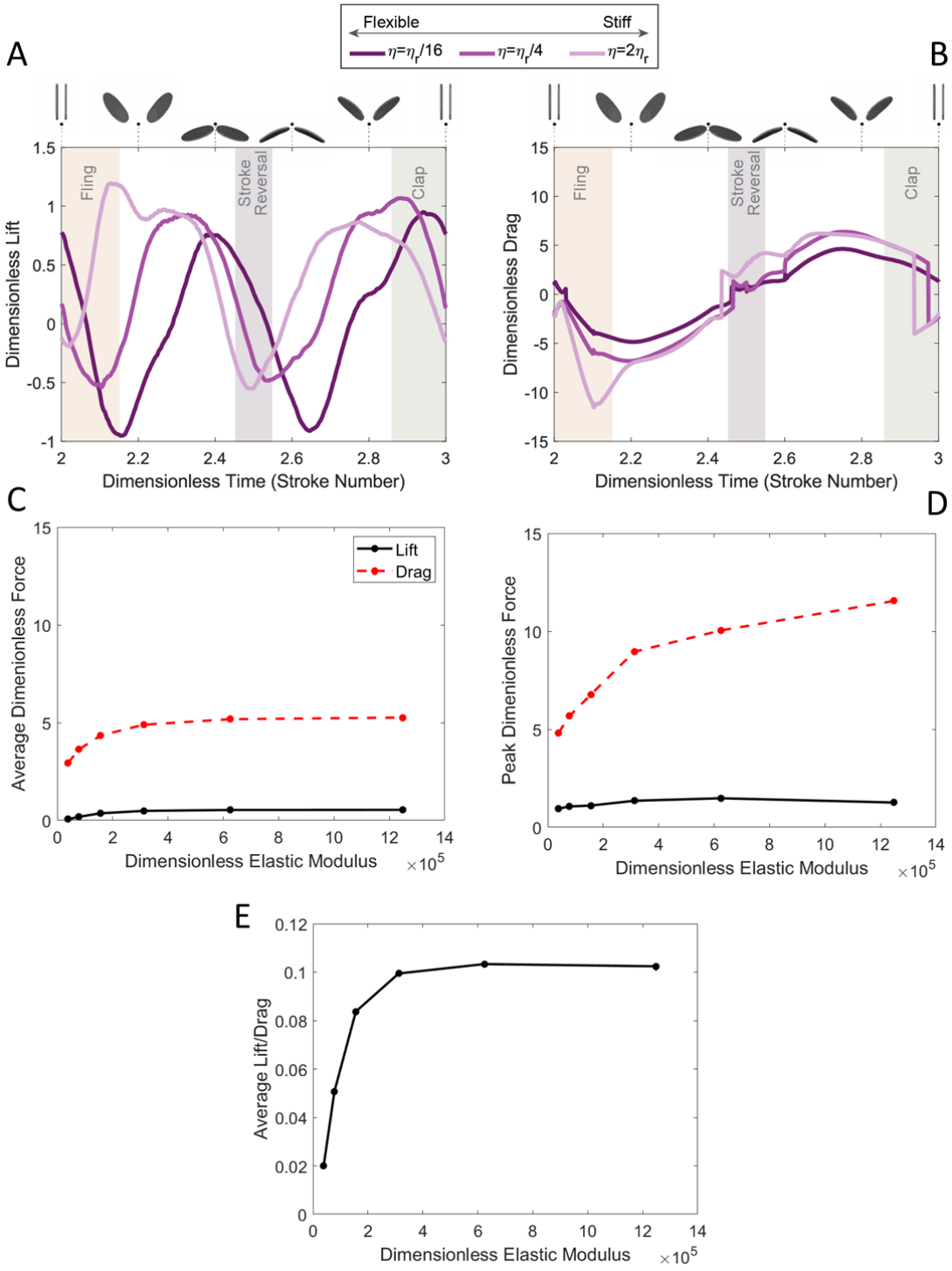

Figure 3.8: Aerodynamic forces for a range of wing flexibilities. A) Dimensionless lift vs. dimensionless time and B) dimensionless drag vs. dimensionless time at $Re = 8$ for $\eta' = \eta_r/16, \eta_r/4$, and $2\eta_r$. C) Average dimensionless lift and drag and D) peak dimensionless lift and drag taken during the third stroke for a range of η'. D) Average lift over average drag for a range of η'.

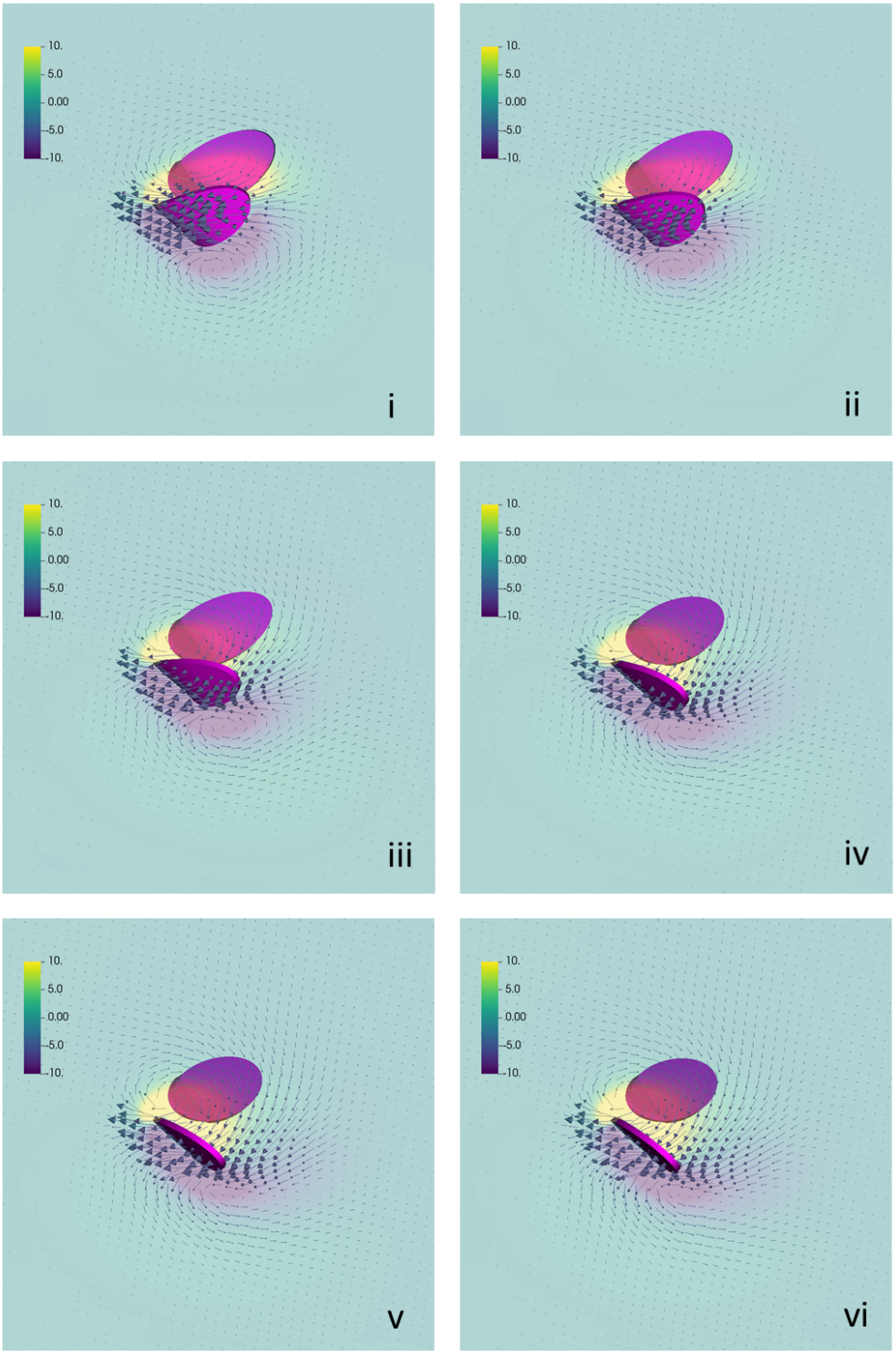

Figure 3.9: Vorticity and velocity fields in a slice for two wings performing a clap and fling at $Re = 8$. Snapshots were taken during the third stroke at time $t' = 2.25$. The flexibilities were set to $\eta' = \eta_r/16, \eta_r/8, \eta_r/4, \eta_r/2, \eta_r, 2\eta_r$.

(Figure 3.9i, ii, iii). Wing deformations are minimal during the downstroke for $\eta' > 3.125 \times 10^5$ (Figure 3.9v, vi). A comparison of the LEV across the range of η' shows that the LEV is advected farther downstream for lower values of η'. In fact, the LEV is advected farther downstream than the TEV in this plane for $\eta' = \eta_r/16$. The opposite is true for $\eta' = 2\eta_r$. The wakes of the stiffer wings are representative of typical studies of insect flight where there is a stable, attached LEV, and the TEV either separates from the wing or is advected downstream [13, 113]. It is this vortical asymmetry that enhances lift generation in insect flight [13, 22], and it is not present if the wings are too flexible.

3.4 Discussion

A major result of this study is that three-dimensional clap and fling allows for spanwise flows in addition to chordwise flows that fill the space between the wings. This effect generates significantly less drag than 2D clap and fling at low Reynolds numbers ($Re < 10$). Indeed, peak drag generated from wing-wing interaction during the fling is only about 40% higher than one wing moving with the same motion. This is strikingly less than the order of magnitude increase observed in 2D [26]. Another significant result is that decreasing wing flexibility reduces lift and drag proportionally until some threshold value. Below this bending rigidity, lift is reduced more than drag, decreasing aerodynamic efficiency. This is due to the fact that negative lift is generated shortly after each wing reversal. Finally, stable attached LEV's and TEV's are observed for $Re < 20$, and stable attached LEV's with separated TEV's are generated for $Re > 32$. This is consistent with single wing studies using numerical and dynamically scaled physical models [101].

In addition to 3D effects and wing flexibility, wingbeat kinematics and wing bristles may also reduce the drag required to fling the wings apart at low Re. Many insects that fly at $Re \approx 10$ have wings that are characterized by bristles that cover a significant portion of the wing planform [23, 129]. There is some airflow between the bristles, particularly during the fling, that reduces the drag the wings generate while maintaining lift [23]. It is also likely that more realistic wing planforms in general can enhance aerodynamic performance of tiny insects [130, 131]. Recent measurements of the wingbeat kinematics of parasitoid wasps and subsequent numerical simulations have also shown that realistic movements of the insect during flight can substantially reduce drag. Cheng and Sun [115] measured the flapping kinematics of the tiny parasitoid wasp *Encarsia formosa* and found that during the clap, the wings rotate to a large angle of attack before they are pressed towards each

other. The wings then move vertically upward, which is much different than the idealized description of clap, and do not get closer than about 0.2 chord lengths. They then used computational fluid dynamics to show that the drag forces required to clap and fling the wings are no longer substantial, and the horizontal forces generated during fling are slightly lower than the vertical forces generated.

As such, future work should consider more realistic aspects of the wing design. For example, insect wings typically have nonuniform material properties due to wing venation patterns, varying thickness, and other heterogeneities [132, 133, 134, 135]. Furthermore, the material properties of the wings in general may not be well described by a neo-Hookean model. As a result, the deformations of the wings under load may be different, resulting in changes in the aerodynamic forces generated. As mentioned previously, many of the wings of the smallest flying insects are bristled. The aerodynamic effect of such wings could be modeled by resolving the flow between individual bristles [23, 116] or by using a porous approximation, as has been done previously in 2D [27]. Several studies have shown that bristled wings experience lower lift and drag than equivalent solid wings when the effective surface area of the wing is considered (e.g. the bristles plus the gap) [23, 136]. Experimental studies using dynamically scaled models have shown that shear layers form around the bristles during fling that reduce the effective leakiness of the wings and minimize the loss of lift [116]. It is also important to note that when only the actual surface area is considered, the bristled wing generates more lift relative to a solid wing [137, 138].

The use of more realistic wing kinematics in numerical simulations of flight would also improve our understanding of how tiny insects generate lift and thrust. For example, variations in kinematics across species and scale could be mapped to performance. The wings of the parasitoid wasps *A. compressa* and *M. Raptor* appear to touch at the end of the clap [26]. On the other hand, there is a visible gap between the wings of thrips at the end of the clap [27]. Variation in the gap size could be due in part to the large forces required to clap the wings together and fling them apart at lower Re. Asymmetries in the timing of the downstroke, variations in the stroke plane angle, and changes in the angles of attack could also serve to enhance lift, reduce drag, increase thrust, or improve maneuverability. For example, advanced wing rotation at the end of the downstroke could increase the average lift generated, as has been shown using a dynamically scaled physical model [114]. More complicated kinematics could also enhance the vertical force generated during the upstroke, particularly at lower Re, as revealed by computational studies of the U-shape upstroke

used by tiny insects such as midges and thrips [139]. When coupled with kinematic studies that quantify wingbeat patterns, computational fluid dynamics offers a powerful tool for understanding the diversity of flight in insects that are too small to feasibly measure aerodynamic forces directly.

CHAPTER 4

The hydrodynamics of metachronal paddling in brine shrimp

4.1 Introduction

Metachronal paddling can be described as the sequential oscillation of appendages whereby adjacent paddles maintain a nearly constant phase difference. This mechanism is used in many different contexts in organisms, such as swimming in crustaceans [33, 67], ctenophores [140], and paramecia [141]; the movement or clearance of mucus in systems such as the mammalian lung [142] and the surface of corals [143, 144]; and the generation of feeding flows in organisms such as *Stentor* [145] and bivalves [146]. In addition to this variety of applications, metachronal paddling is also used across tens of orders of magnitude in Reynolds number (Re), from swimming in unicellular organisms [141] to escape swimming in larger crustaceans [147].

Zooplankton, which span spatial scales from tens of microns to tens of centimeters, have a variety of ways to actively propel themselves through the water, including metachronal paddling. During development, many of these organisms also change their method of locomotion as they grow in size. One example of such an organism is *Artemia*, or brine shrimp. Newly hatched nauplii swim primarily with a pair of antennae in a motion that is similar to a breaststroke. Such a motion is reminiscent of microorganisms that are termed 'pullers' in which water is pulled from the front of the organism to its sides [148, 149]. This swimming mode is typically observed as a low Re mechanism of swimming, and nauplii swim at Re of about 1-4. On the other hand, adult brine shrimp rhythmically move linear arrays of limbs through power strokes and return strokes in a back-to-front metachronal wave. This swimming stroke appears to be similar to that used by other long-tailed crustaceans (e.g. crayfish, krill, shrimp, and lobsters), and is commonly observed in larger scale swimmers [150]. The adult brine shrimp swim at Re of about 40, where inertial forces are significant relative to viscous forces.

Experimental, computational, and modeling work has supported that a back-to-front propagating metachronal wave, where the posterior limbs lead the cycle, generates more flow for propulsion and

feeding than other stroke patterns [67, 151, 152, 153]. Alben *et al.* [154] used a drag-force-based model to show that metachronal paddling leads to less than about a 10% greater body swimming velocity than a synchronized pattern and argued that metachronal paddling leads to a smoother swimming velocity. Numerical simulations have shown that for Re in the range of 50 to 800, the back-to-front metachronal pattern with a 25% inter-paddle phase-difference leads to a 60% increase in average flux compared to synchronous swimming and a 500% increase compared to the front-to-back metachronal pattern [67]. Using parameters relevant to crayfish, Granzier-Nakajima *et al.* [33] expanded this study to show that the 20 % to 25 % phase difference in back-to-front paddling generated the highest average flux for Re ranging from about 0 to 100. Experiments using a dynamically scaled robot also found that metachronal paddling generated significantly more vertical flow that can be used for lift generation [32].

Compared to previous work on metachronal paddling, the propulsive appendages of brine shrimp are spaced closer together and beat with a smaller phase difference ($\approx 10\%$). The hydrodynamics and efficiency of such a swimming pattern have not been rigorously explored. To quantify the flows generated and the efficiency of this swimming pattern, we will use the immersed boundary method to numerically solve the fluid-structure interaction problem of different numbers of legs performing a back-to-front metachronal stroke using parameters relevant to brine shrimp. Using the average volumetric flux over the average power per leg, we quantify the efficiency of metachronal paddling across Reynolds numbers. We will also consider the effect of Re and leg spacing on the resulting flows generated.

4.2 Methods

4.2.1 Immersed Boundary Method

Numerical simulations were performed using the immersed boundary (IB) method [4]. The following outline describes the two-dimensional formulation of the immersed boundary method. The equations of fluid motion are given by the Navier–Stokes equations,

$$\rho(\mathbf{u}_t(\mathbf{x},t) + \mathbf{u}(\mathbf{x},t)\cdot\nabla\mathbf{u}(\mathbf{x},t)) = -\nabla p(\mathbf{x},t) + \mu\nabla^2\mathbf{u}(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t), \tag{4.1}$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t) = 0, \tag{4.2}$$

where $\mathbf{u}(\mathbf{x},t)$ is the fluid velocity, $p(\mathbf{x},t)$ is the pressure, $f(\mathbf{x},t)$ is the force per unit area applied to the fluid by the immersed boundary, ρ is the density of the fluid, and μ is the dynamic viscosity of the fluid. The independent variables are the time t and the position $\mathbf{x}$. The interaction equations are

$$\mathbf{f}(\mathbf{x},t) = S\mathbf{F}(s,t) = \int \mathbf{F}(s,t)\delta\left(\mathbf{x} - \mathbf{X}(s,t)\right) ds, \tag{4.3}$$

$$\mathbf{U}(\mathbf{X}(s,t)) = S^*\mathbf{u}(\mathbf{x},t) = \int \mathbf{u}(\mathbf{x},t)\delta\left(\mathbf{x} - \mathbf{X}(s,t)\right) d\mathbf{x}, \tag{4.4}$$

where $\mathbf{F}(s,t)$ is the force per unit length applied by the boundary to the fluid as a function of Lagrangian position and time, $\delta(\mathbf{x})$ is a two-dimensional delta function, $\mathbf{X}(s,t)$ gives the Cartesian coordinates at time t of the material point labeled by the Lagrangian parameter s. Note that each of these equations involves a two-dimensional Dirac delta function that acts as the kernel of an integral transformation. The operator S spreads the force density from the paddles to the fluid, and the fluid velocity is interpolated to the paddles using the adjoint of the spreading operator, S^*.

We would like the motion of the immersed boundary (the paddles) to be prescribed. Let $\mathbf{U}_p(s,t)$ give the velocity of the walls and paddles. The fluid must match this velocity on the immersed boundary, and this constraint can be expressed as

$$S^*\mathbf{u}(\mathbf{x},t) = \mathbf{U}_p(s,t). \tag{4.5}$$

$\mathbf{F}(s,t)$ is then the force required to enforce this constraint. For further details on determining the immersed boundary forces required to enforce prescribed motion, see Taira and Colonius [155] and of Kallemov *et al.* [156]. For details of the particular implementation of the immersed boundary method used in this chapter, see Granzier-Nakajima *et al.* [33].

4.2.2 Model Formulation

This model consists of n rigid paddles of length l attached to a horizontal wall (representing the brine shrimp body) in a two-dimensional fluid. The paddles are indexed by i from left to right and are equally spaced along the wall at distance w from each other at the base. The motion of the i^{th} paddle is described by the angle α_i formed between the wall and the paddle, with α_i defined to be

$$\alpha_i(t) = \bar{\alpha} - A\cos\left(2\pi\left(\frac{t}{T} + \Phi_i\right)\right), \tag{4.6}$$

where $\bar{\alpha} = \pi/2$ is the mean angle, $A = \pi/4$ is the amplitude, T is the period, and Φ_i is the phase of the stroke, and t is time. As the cyclic movements of the legs of the brine shrimp all have approximately the same phase, we take one time unit to be equal to one full cycle of the power and return paddle strokes. Figure 4.1 shows the model setup.

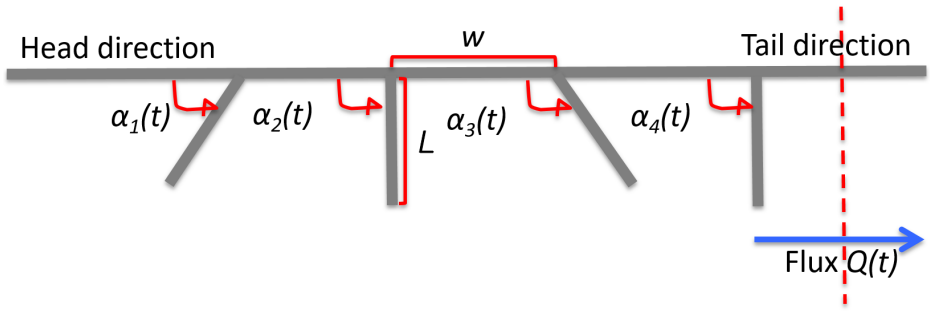

Figure 4.1: Schematic of a metachronal paddler with four paddles to illustrate the model used in our simulations. The left-hand side represents the head direction in which the paddler swims, while the right-hand side represents the tail area. The angles between the legs and the body of the swimmer are marked as $\alpha_i(t)$, the length of the leg is marked by l, and the constant distance between adjacent paddles is marked by w. All simulations we ran utilized swimmers with eleven paddles.

In this model the power and return stroke are mirror images of each other. Thus a single paddle by itself would produce no movement of the organism or surrounding fluid, and the inter-paddle phase difference $\Delta\Phi$ between different paddles in a multi-paddle model is required to produce movement. Note that $\Delta\Phi = 0$ would represent all paddles beating synchronously (in-phase), while $\Delta\Phi = 0.5$ would represent the paddles beating perfectly out of phase. We assume a constant phase difference $\Delta\Phi = 0.08$ between neighboring paddles, which produces a metachronal wave. For this value of $\Delta\Phi$, the paddles move in a back-to-front moving wave (leftward in Figure 4.1) to produce positive flow (rightward in Figure 4.1).

We define the paddle-based Reynolds number (Re) for these simulations as

$$Re = \frac{2\pi A L^2 \rho}{\mu T}, \tag{4.7}$$

where the characteristic length, L, is taken to be the paddle length, and the the characteristic velocity is the maximum speed of the paddle tip, $U_{max} = 2\pi AL/T$.

The model organism is embedded in a rectangular channel with no-slip boundary conditions at the top and bottom and periodic boundary conditions in the horizontal direction. The channel has

unit height and a length of 10 units. The number of paddles is 11, the number of legs of an adult brine shrimp, with paddle length chosen to be 0.15 times the channel height as in [33]. The paddle spacing is $L/2$ by default, with parameter sweeps increasing the paddle spacing by $L/8$ up to L. Re is taken to be 40 by default, but Re values from 0.1 to 40 are considered. For these simulations, Re is changed by modifying the dynamic viscosity of the fluid, μ. This is done so that only one parameter needs to be changed to vary Re. In the simulations, the mesh in the vertical y direction is divided into 256 points and in the horizontal x direction into 2560 points ($\delta x = \delta y = 3.9 \times 10^{-3}$). A time step of $dt = 5 \times 10^{-4}$ is used for the numerical simulations, but the post-processing analysis only uses the output from every 100^{th} time step, so that the analysis time step is effectively $dt = 0.05$.

4.2.3 Analysis of Flow Fields

One way to quantify paddling performance for a tethered array of paddles is to determine the amount of fluid per unit time that flows through the channel, or the flux. In this case, the flux is defined as

$$Q(t) = \int_0^1 \mathbf{u} \cdot \hat{\mathbf{x}}\, dy, \tag{4.8}$$

where $\hat{x}$ is the unit vector that points in the horizontal direction. Q does not depend upon the choice of x because the flow is incompressible and no-slip boundary conditions are used on the top and bottom of the channel. We define the time-averaged flux as

$$\langle Q \rangle = \int_{t'}^{t'+1} Q(t)\, dt, \tag{4.9}$$

where the flux is averaged over a stroke period of time t' to $t' + 1$. Unless otherwise stated, the flux is averaged over the last stroke cycle simulated.

Another way to quantify performance is to determine the ratio of the time-averaged flux to the time-averaged power per paddle. We describe this as a pseudo-efficiency, ε. Note that ε is not dimensionless and has units of distance per unit energy. We use this choice of the pseudo-efficiency to make comparisons with crayfish swimming [33]. To determine this ratio, we first calculate the power supplied to the fluid from the paddles, $J(t)$. This power may be calculated as

Figure 4.2: A frame from the high speed recording of a female brine shrimp swimming freely. The thoracopods were annotated for kinematics measurements (4 to 7 in the image), and the angle between the body and the tip of the leg was measured to describe the movement of the legs.

$$J(t) = \int_{\Gamma} \mathbf{F}(s,t) \cdot \mathbf{U}(s,t)\, ds, \tag{4.10}$$

where Γ represents the paddles, s gives the parametric coordinate on the paddle, $\mathbf{F}$ is the force density applied to the fluid, and $\mathbf{U}$ is the velocity of the paddles. We can then define ε as

$$\varepsilon = \frac{\langle Q \rangle}{\langle J \rangle}. \tag{4.11}$$

4.2.4 Measurement of Brine Shrimp Kinematics

Artemia franciscana were cultured in the lab environment from Grade A Brine Shrimp Eggs (Brine Shrimp Direct, All Aquaria, L.C.) to adulthood. High speed recordings of brine shrimp larvae and adults were recorded under a Leica DM500 light microscope and Fisher Scientific stereo microscope. The kinematics of five female and five male *Artemia* were quantified using Adobe Photoshop 2021 and Fiji 2 [157]. Size measurements were taken in Fiji 2.

The position of each leg was approximated by extending a line from the base of the leg to its tip (see Figure 4.2). The angle between the axis of the body and the leg was then measured as a

function of time for each of the legs. The phase difference was estimated by measuring the times when each leg is held 90 degrees relative to the body during the power stroke. The difference in this time between neighboring legs was divided by the period and averaged. The distance between legs was estimated by measuring difference in the position of the first leg and the last leg and dividing by 10, as we assume equal leg spacing for our simulations.

4.3 Results

4.3.1 Brine shrimp kinematics

The average leg length for the adult brine shrimp was 1.79 ± 0.20 mm, and the average ratio of the space between legs over the leg length was 0.19 ± 0.023. Given that rigid paddles will overlap with this spacing, we extended the space between legs to $L/2$ and then performed a parameter sweep to consider the effect of leg spacing. The average phase difference between neighboring legs was $8.26\% \pm 0.25\%$.

4.3.2 Brine Shrimp Swimming

A reference simulation was performed to quantify the flow fields generated by a typical adult brine shrimp. In this case, eleven paddles are equidistantly placed at a distance of $w = L/2$, swimming with a phase difference of $\phi = 8\,\%$ at $Re = 40$. Note that the legs are spaced further apart than an actual adult brine shrimp to avoid the collision of rigid legs as mentioned above (section 4.3.1).

The flux averaged over a single period for 150 stroke cycles is shown in figure 4.3. We note that it takes a surprisingly long time for the average flux to reach a steady state, taking more than 100 complete strokes cycles. On the other hand, the power consumption $J(t)$ reaches stability within the first 12 strokes, as shown in figure 4.3. Once periodic steady state is achieved, the larger peak occurs at 40 % of the stroke and corresponds to the time when the legs point towards the middle of the thorax, while the smaller peak occurs at 85 % of the stroke and corresponds to the time when the legs point outward, away from the brine shrimp. During these points in time, some of the paddles are moving leftward against strong rightward flow.

Figure 4.4 shows the vorticity and velocity of the fluid flow generated for several snapshots during the final stroke ($t = 149 - 150$). While all simulations were conducted on a 10×1 domain, the flow field in the images has been cropped to $[0, 1.8] \times [0.3, 0.8]$ in order to better see the flow near the legs. The colormap shows the vorticity, with blue representing clockwise flow and red representing

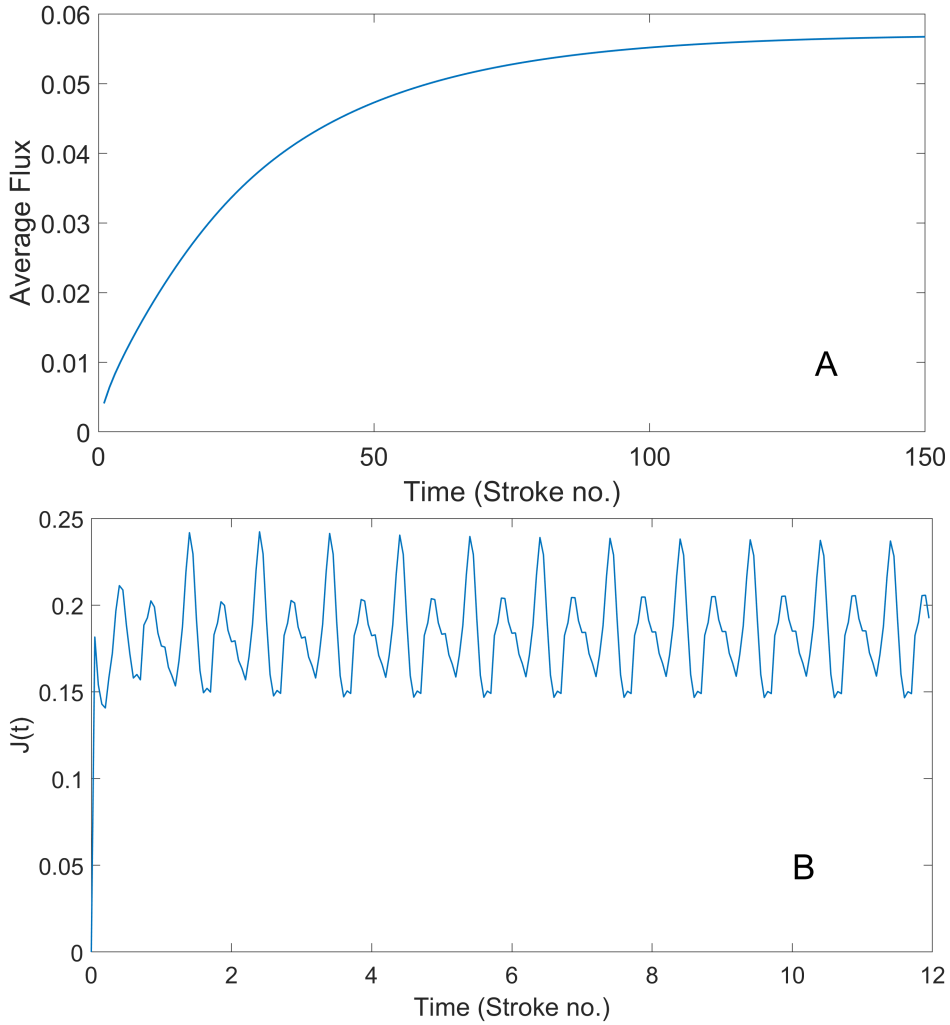

Figure 4.3: (A) The average flux $\langle Q(t) \rangle$ for the reference model of a brine shrimp with eleven legs equidistantly placed at a distance of $w = L/2$ swimming with a phase difference of $\phi = 8\,\%$ at $Re = 40$ and (B) the power consumption through the first twelve strokes. The larger peak in power consumption occurs when $40\,\%$ of the stroke has been completed and all legs face towards the swimmer, while the second peak occurs at $85\,\%$ and occurs when all legs point away from the middle of the swimmer. We note that although it takes a large number of strokes for the flux to approach steady state, power consumption reaches a steady state in a much shorter period of time.

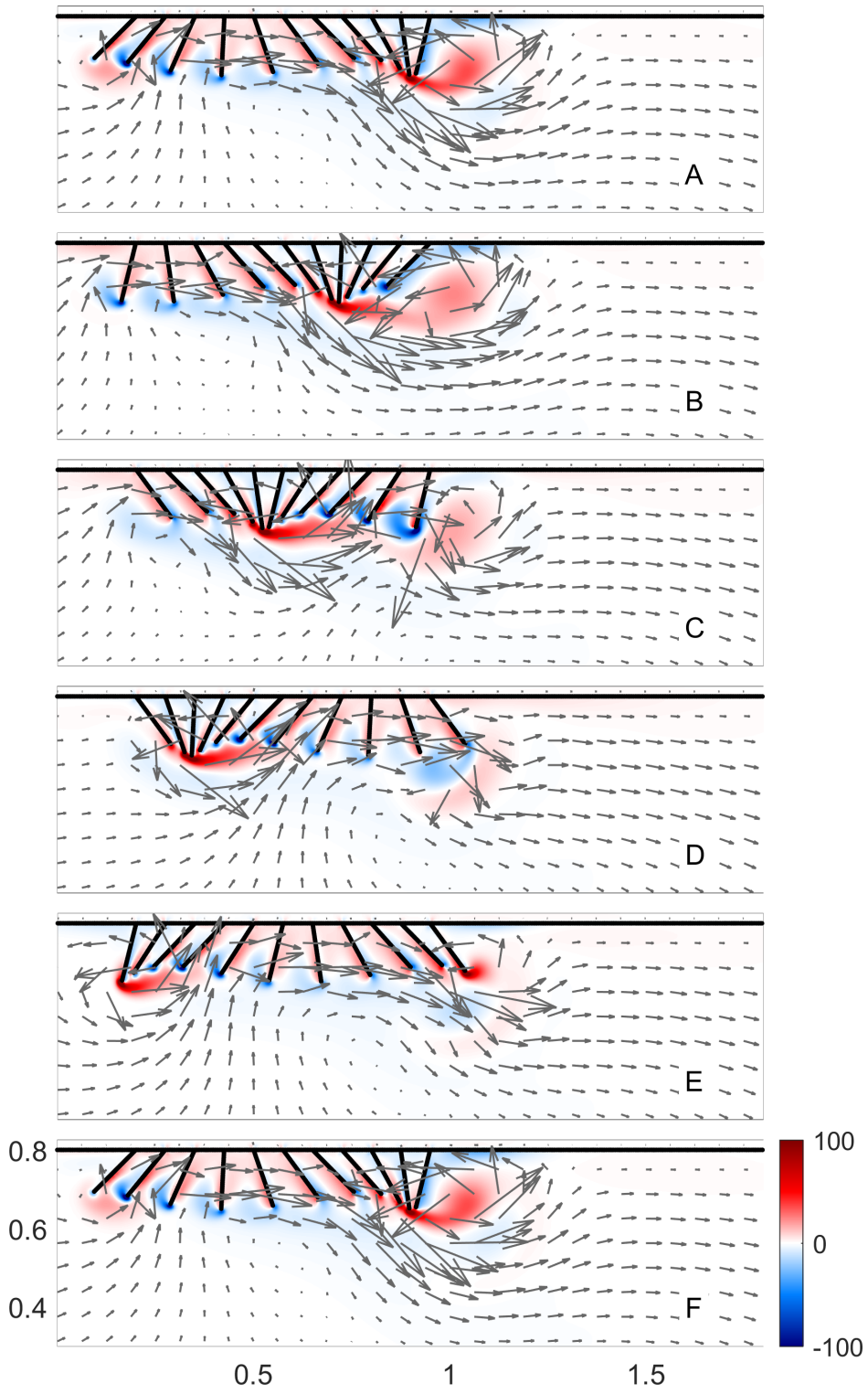

Figure 4.4: The vorticity and velocity field for the $Re = 40$ reference model with leg distance $L/2$ over the last stroke at times (A) $t = 149.00$ (B) $t = 149.20$ (C) $t = 149.40$ (D) $t = 149.60$ (E) $t = 149.80$ (F) $t = 150.00$.

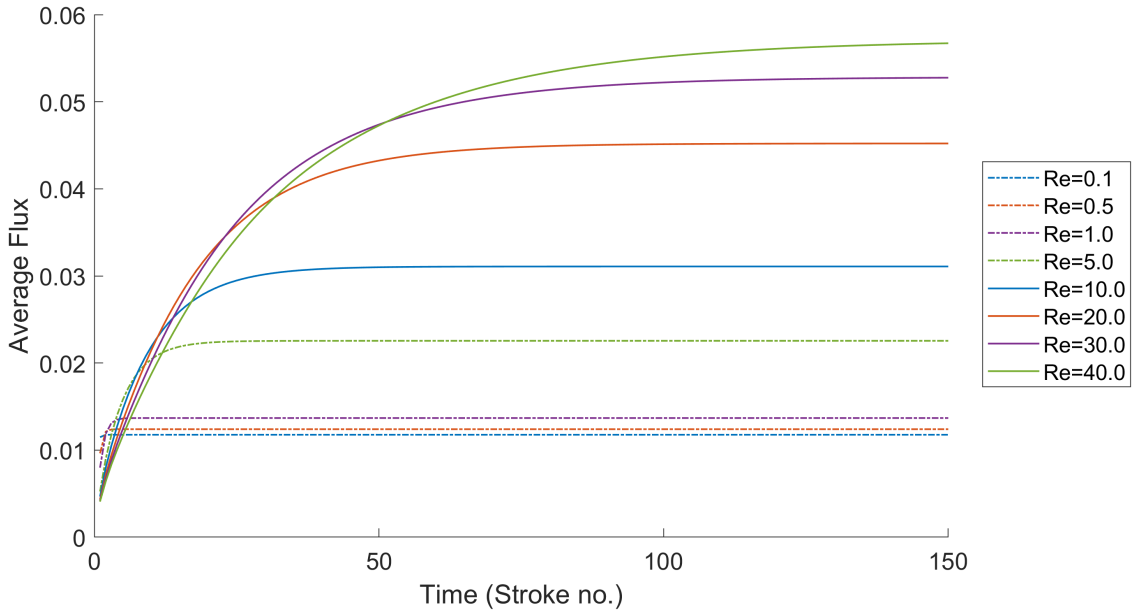

Figure 4.5: Average flux across all the different Re that were examined in section 4.3.3. The higher the Re, the longer the flow takes to achieve steady state and the higher the average flux $\langle Q \rangle$ at steady state.

counterclockwise flow. The arrows show that direction and magnitude of the velocity. Clockwise (blue) vortices form at the tip of each paddle during the power stroke and are shed at the beginning of the return stroke. During the return stroke, counterclockwise (red) vortices form at the tip of each paddle until they are shed at the end of this half-stroke. Strong jets are formed where regions of positive and negative vorticity collide, for example, between paddles 9 and 10 at $t = 149.0$ and between paddles 3 and 4 at $t = 149.40$. Given the relatively low Re, shed vortices dissipate relatively quickly, such as the counterclockwise (red) vortex to the right of the array of paddles at $t = 149.0$. Similarly signed vortices can also combine and strengthen between adjacent paddles.

4.3.3 Effect of Re

We next analyzed the effect of changes in Re on the flow. We repeated the simulation as described in section 4.3.2 and only varied the Re by changing the dynamic viscosity μ. Simulations were performed for $Re \in \{0.1, 0.5, 1, 5, 10, 20, 30, 40\}$, representing the Re from newly hatched nauplii through adults. Note that the low Re cases are not biologically relevant as nauplii swim with a pair of antennae. This analysis does, however, allow us to explore what the hydrodynamics of brine shrimp swimming would be like if newly hatched nauplii had the same morphology as adults.

For comparison purposes, we have visualized the average flux over a simulation of 150 strokes

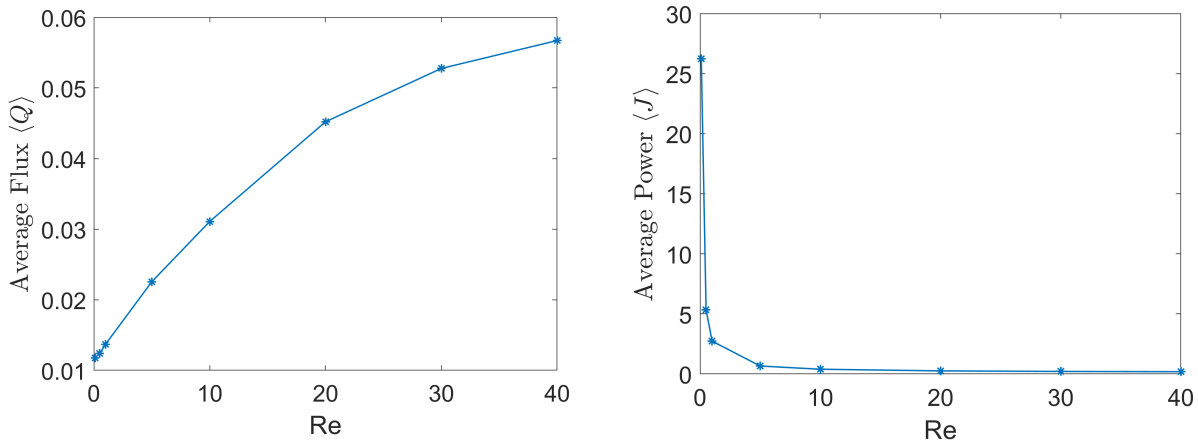

Figure 4.6: Average flux and power consumption as a function of Re during the final stroke of simulations in section 4.3.3. The average flux steadily increases with Re, reaching a near plateau towards the highest Re simulated. Note the nearly five-fold increase in magnitude of $\langle Q \rangle$ between $Re = 0.01$ and $Re = 40$. The average power is highest for the lowest Re, falling off drastically as Re values increase.

for all Re in figure 4.5. We can see that the average flux increases with Re. Furthermore, the flux quickly reaches steady state within a couple of stroke cycles for $Re \leq 1$. The time required to reach steady state increases with Re. For $Re = 40$, the average flux has not yet reached steady state, even after 150 stroke cycles. This has implications for escape swimming given the relatively small acceleration of adults. Interestingly, brine shrimp have few predators and paddle most of the time for feeding and exchange.

The average flux as a function of Re during the final stroke is graphed in figure 4.6. The average flux is steadily increasing with the Re and starts to plateau at the highest Re. Note that the average flux at $Re = 40$ is nearly five times that for $Re = 0.1$, and the average power rapidly decreases with increase in Re. The normed efficiency, graphed in 4.8, increases almost linearly with Re for this range of values.

Figure 4.7 shows a color map of the vorticity and velocity vectors for the end of the final stroke where $Re = 0.1$, 1.0, 5.0, 10.0, 20.0 and 40.0. The blue regions show clockwise (negative) vorticity and the red regions show counterclockwise (positive) vorticity. For $Re \leq 1$, the vortices that form on each paddle during the power stroke (blue) combine to form a large region of clockwise vorticity in the region below the paddles. A region of positive vorticity forms to the right of the 11th paddle, and the interaction of these two vortices creates a downward jet with moderate flux from left to right. As the Re increases, the regions of negative vorticity do not extend much below the paddles.

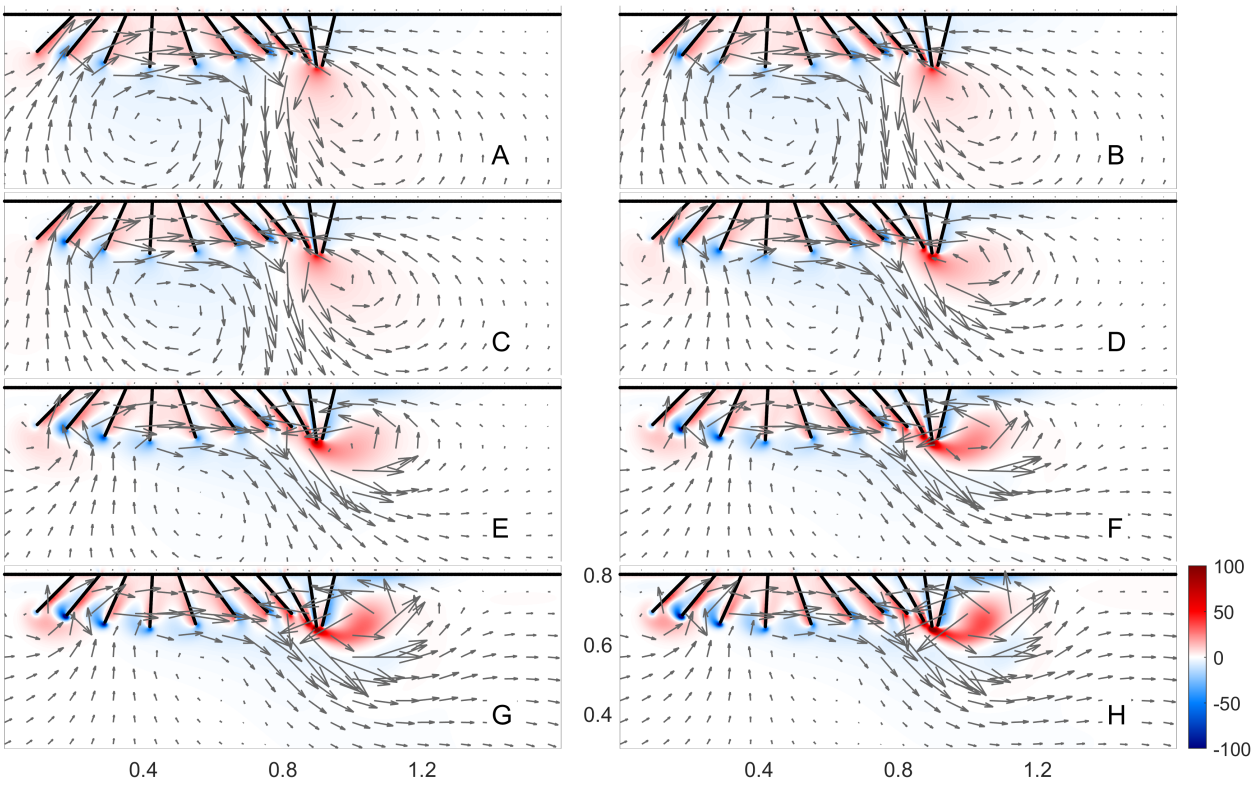

Figure 4.7: Vorticity and velocity for the end of the final stroke for varying Re as described in section 4.3.3. Subfigures represent (A) $Re = 0.1$ (B) $Re = 0.5$ (C) $Re = 1$ (D) $Re = 5$ (E) $Re = 10$ (F) $Re = 20$ (G) $Re = 30$ and (H) $Re = 40$.

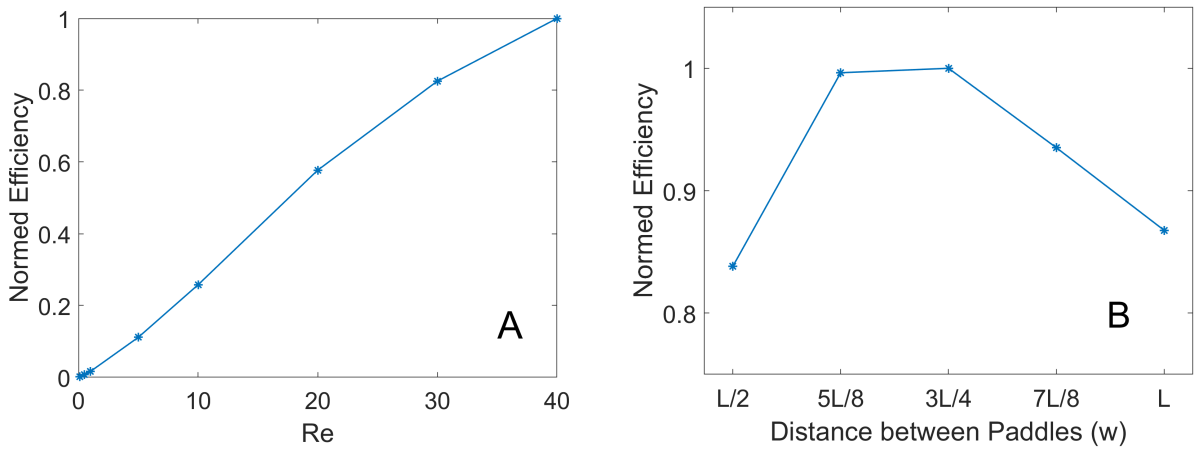

Figure 4.8: The normed efficiency of the swimmer with eleven paddles using a phase difference of $\Delta\Phi = 0.08$ as functions of both Re (A) and leg-spacing w (B). Efficiency increases nearly linearly with Re, being maximized at $Re = 40$. The lowest Re values are very inefficient, with normed efficiency less than 0.01 for $Re < 1$. When changing the paddle-spacing the normed efficiency shows a less dramatic but more non-linear dependency on the tested values. It was maximized towards the middle of our range of values, with maximal normed efficiency achieved at $w = 3L/4$.

A strong positive vortex forms to the right of the last paddle, and its interaction with the negative vortices generated by each paddle creates a strong jet that is about 45 degrees from the horizontal, creating significant positive flux.

4.3.4 Effect of Leg Spacing for Brine Shrimp

Due to our model's use of rigid paddles, we cannot achieve as close a spacing as found in brine shrimp. To consider the effect of our paddle distance, we ran simulations with leg spacing $L/2$, $5L/8$, $3L/4$, $7L/8$ and L for paddles of length L and $Re = 40$. The remaining settings were the same as the reference model described in section 4.3.2.

The average flux and power consumption as a function of the leg spacing are graphed in figure 4.9. While maximal flux is achieved for the tightest spacing between the paddles, this scenario also exhibits the largest power consumption for the range of spacings considered. Note that the average flux decreases with increasing distance between the paddles. The flux plateaus for spacings less than 5L/8, suggesting that our results for a spacing of L/2 are representative of brine shrimp. In terms of power consumption, the average power decreases initially with increased spacing, reaches a minimum near 3L/4, and then increases again. As a result, the tightest spacing is not the most efficient, as can be seen in figure 4.8. Here, the normed efficiency vs. the distance between paddles is shown. Peak efficiency occurs for a leg spacing between 5L/8 and 3L/4.

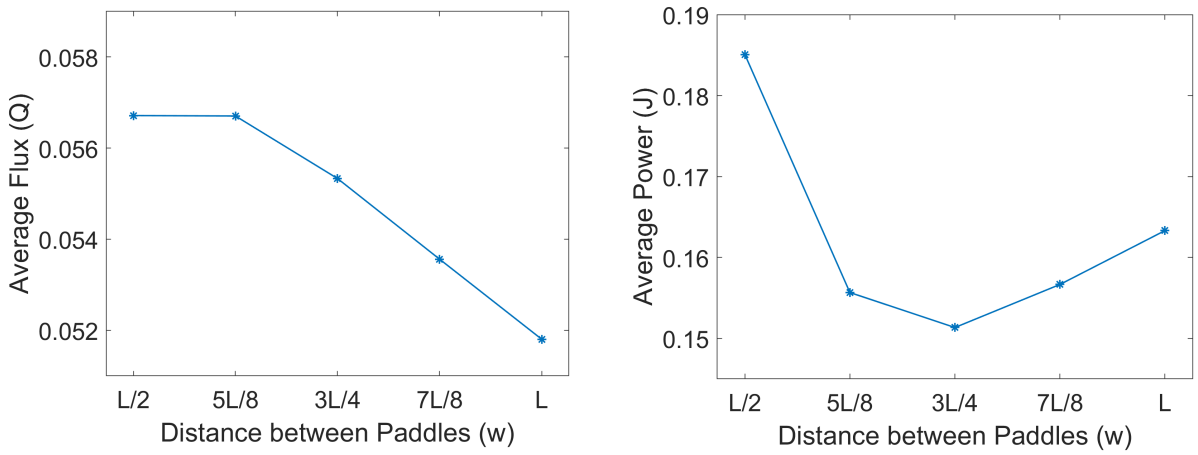

Figure 4.9: Results of a leg-spacing study for Reynold's Number 40. Our default spacing in the other sections was $w = L/2$, which corresponds to the left-most data point at distance $w = 0.075$. We note that at a distance of greater than $5L/8$ we see a steady decrease in average flux. The average power consumption however seems to be minimized at a distance $w = 3L/4$.

Figure 4.10 shows the vorticity and velocity field at the end of the last stroke cycle for leg spacings set to $L/2$, $5L/8$, $3L/4$, $7L/8$ and L. As before, the color map indicates the vorticity with red corresponding to counterclockwise (positive) vorticity and blue corresponding to clockwise (negative) vorticity. Flux is maximized at the tightest leg spacing where vortex interactions are the strongest. Note how tip vortices combine for the tightest leg spacing and are nearly independent for the widest spacing. This effect can be observed by comparing the strength of the vortex to the right of the last leg as the leg spacing increases as well as the magnitude of the downward jet due to the interaction of this vortex with the oppositely spinning tip vortices formed during the power stroke (the jet appears between legs 8 and 9).

4.4 Discussion

In this chapter, we have numerically simulated 11 paddles beating in a back-to-front metachronal fashion with an 8 percent phase leg between neighboring paddles. These kinematics are similar to those of adult brine shrimp, swimming near $Re = 40$. We found that over 150 stroke cycles are required to reach steady state, suggesting that this form of locomotion is not ideal for escape swimming, but may be beneficial for steady cruising, feeding, and gas exchange. Periodic steady state is, however, quickly achieved for $Re \leq 1$. Both average flux and normed efficiency increase with increasing Re. In terms of the distance between paddles, the average flux is maximized for paddle spacings near L/2, while the average power is minimized for paddle spacings near 3L/4. As a result,

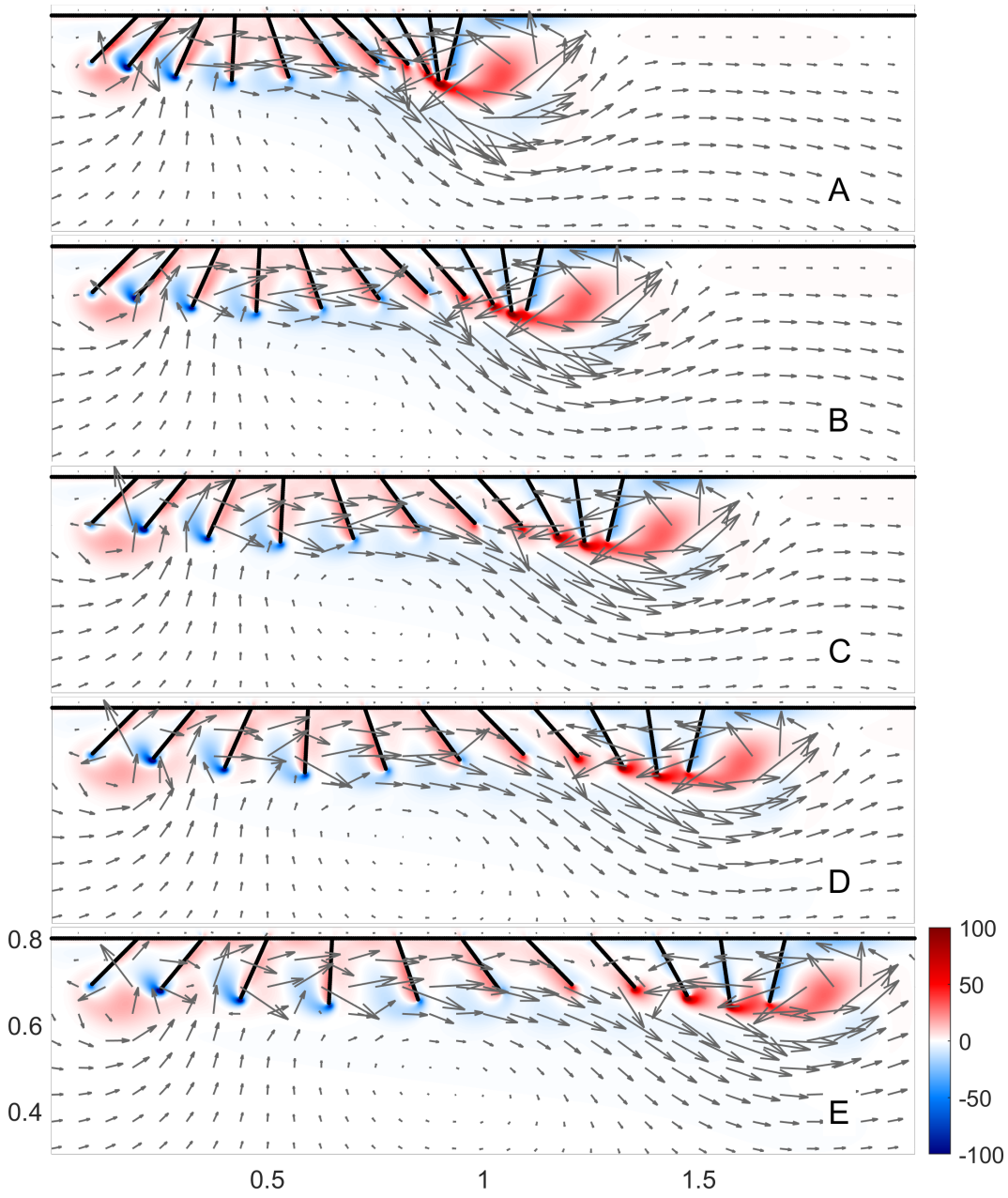

Figure 4.10: End of last stroke for different widths w between the base of the paddles. (A) $w = L/2$; (B) $w = 5L/8$; (C) $w = 3L/4$; (D) $w = 7L/8$; (E) $w = L$.

the maximum normed efficiency occurs between a leg spacing of 5L/9 and 3L/4. This suggests that the tight spacing between brine shrimp legs optimizes flux rather than efficiency.

While this study represents a good step forward in resolving the complex fluid-structure interaction of brine shrimp swimming, a number of model simplifications have been made that should be further explored. First of all, this is a 2D study, and brine shrimp swimming is inherently three-dimensional since 1) the appendages do move out of the plane and 2) there is fluid motion in the third dimension as the fluid is pulled between and squeezed out of the space between adjacent paddles. The motion of the thoracopods has been simplified so that they only bend at the base and move in a sinusoidal motion. The actual thoracopods are jointed and flexible, and the motion is asymmetrical and more complex than simple sinusoidal motion. The thoracopods are also bristled, permitting some flow between the bristles such that the appendage acts as a porous paddle. Finally, we were not able to position the paddles as close together as in the actual organism given the simplifications made in the paddle motion. Further studies should consider flexibility and porosity, which would also allow the paddles to be placed close together.

CHAPTER 5

Conclusions

Good news, everyone!

Professor Farnsworth, Futurama.

Numerous fluid-structure interaction problems in biology have been investigated using the immersed boundary method. The advantage of this method is that complex geometries, e.g., internal or external morphology, can easily be handled without the need to generate matching grids for both the fluid and the structure. Consequently, the difficulty of modeling the structure lies often in discretizing the boundary of the complex geometry (morphology). Both commercial and open source mesh generators for finite element methods have long been established; however, the traditional immersed boundary method is based on a finite difference discretization of the structure. In chapter 2, we developed a software library to generate finite different meshes for discretizing the boundaries of biological structures for direct use in the 2D immersed boundary method. This library provides tools for extracting such boundaries as discrete mesh points from digital images. We give several examples of how the method can be applied that include passing flow through the veins of insect wings, within lymphatic capillaries, and around starfish using open-source immersed boundary software.

In chapter 3, we numerically simulated the clap and fling of two flexible wings. Of the smallest insects filmed in flight, most if not all clap their wings together at the end of the upstroke and fling them apart at the beginning of the downstroke. This motion increases the strength of the leading edge vortices generated during the downstroke and augments the lift. At the Reynolds numbers (Re) relevant to flight in these insects (roughly $4 < Re < 40$), the drag produced during the fling is substantial, although this can be reduced through the presence of wing bristles, chordwise wing flexibility, and more complex wingbeat kinematics. It is not clear how flexibility in the spanwise direction of the wings can alter the lift and drag generated. We used a hybrid version of the immersed boundary method utilizing finite elements to simulate a 3D idealized clap and fling motion across a range of wing flexibilities. We found that spanwise flexibility, in addition to three-dimensional

spanwise flow, can reduce the drag forces produced during the fling while maintaining lift, especially at lower Re. While the drag required to fling 2D wings apart may be more than an order of magnitude higher than the force required to translate the wings, this effect is significantly reduced in 3D. Similar to previous studies, dimensionless drag increases dramatically for $Re < 20$, and only moderate increases in lift are observed. Both lift and drag decrease with increasing wing flexibility, but below some threshold, lift decreases much faster. This study highlighted the importance of flexibility in both the chordwise and spanwise directions for low Re insect flight. The results also suggest that there is a large aerodynamic cost if insect wings are too flexible.

In chapter 4, we used the immersed boundary method to quantify the flows generated by the metachronal motion of brine shrimp swimming appendages. Metachronal paddling is used by a variety of organisms to propel themselves through a fluid. This mode of swimming is characterized by an array of appendages that beat out of phase, such as the swimmerets used by long-tailed crustaceans such as crayfish and lobster. This form of locomotion is typically observed over a range of Reynolds numbers greater than 1 where the flow is dominated by inertia. The majority of experimental, modeling, and numerical work on metachronal paddling is in the higher Reynolds number regime. In this paper, we construct a simplified numerical model of one of the smaller metachronal swimmers, the brine shrimp. Brine shrimp are particularly interesting since they swim at Reynolds numbers on the order of 10 and sprout additional paddling appendages as they grow. We use the immersed boundary method to numerically solve the fluid-structure interaction problem of multiple rigid paddles undergoing cycles of power and return strokes with a constant phase difference and spacing that are based on brine shrimp parameters. Using a phase difference of 8%, we quantify the volumetric flux and efficiency per paddle as a function of the Reynolds number and the spacing between the paddles. We find that the time to reach periodic steady state for adult brine shrimp is large ($\approx$ 150 stroke cycles) and decreases with decreasing Reynolds number. Both efficiency and average flux increase with Reynolds number. In terms of leg spacing, the average flux decreases with increased spacing while the efficiency is maximized for intermediate leg spacing.

5.1 Limitations of the studies and future work

There are several ways to further the work done on MeshmerizeMe that I believe would have a positive impact on future use. At this stage, the software only handles the creation of vertices. The management of the vertices to create springs, beams, muscles, and other types of models is in the

hands of the user. A rudimentary class for the the creation of target points, mass points, springs, and a few other basic structures has already been implemented. These classes allow a user proficient in Python to manually create the necessary files for the simulation based on the vertex file and associated class structures. I believe it to be worthwhile to complete this library to cover all of the basic constituents of the IB2D and IBAMR 2D models. This will make it easier for researchers to create their complete model using Python without the need to switch to different software interfaces.

While coding is efficient for the experienced user, it does present a hurdle for the learner. This can become especially difficult when dealing with a large array of vertex points and needing to figure out which points to connect in what way. One can image a more intuitive interface that gives a visual confirmation of the choices made by the user. Currently, MeshmerizeMe includes a rudimentary plotting feature that allows the user to preview the created mesh. A future version of MeshmerizeMe could include a PyQt-based GUI that allows users to select and group vertex points in a graphical interface and select a class option that should be applied to them. For example, selected points could be turned into target points or connected with springs. This type of interface would help to further reduce the difficulty in entering the biofluids CFD world.

A third potential avenue for development of MeshmerizeMe is to adapt the core of MeshmerizeMe to handle video input. This thesis included two kinematic studies of locomotion - swimming and flying. In both cases, video footage of the relevant organism's locomotion is available to researchers. It would be a great benefit to be able to prescribe kinematics based on actual video footage. A useful Matlab-based tool to extract positional information from collected video footage is already available [158]. The latest iteration of this software utilizes a deep learning methodology to partially automate the tracking of key points in a video, for example the wing tip of a flying bird. A similar method could be implemented with MeshmerizeMe. A still image from the beginning of the video would be used for mesh generation, while supervised boundary tracking methods could be implemented to track how the discovered boundary deforms over time. This would be used to create target points that move the boundary in a realistic manner in IB simulations.

In chapter 3, two flexible elliptical wings were simulated moving with an idealized motion. There are a number of ways that these simulations could be made more realistic. It has been shown for the tiny parasitoid wasp *Encarsia* that the wings do not beat in a horizontal plane [115]. In fact, the wings move upward while being flung apart at the beginning of the downstroke, which is thought

to reduce the drag generated. The incorporation of more realistic wing beat kinematics into our simulations would provide additional insights into how these insects enhance lift while minimizing drag. Furthermore, the flexible wings are treated as a neo-Hookean material with uniform thickness. In reality, the material properties of insect wings are nonuniform and nonlinear [20], in part due to the presence of wing venation. Some tiny insect wings are also bristled such that they act as a porous structure [23]. The treatment of the wings as a porous structure with nonuniform material properties would also provide additional insights into wing design and the aerodynamics of tiny insects.

Similarly, we made a number of modeling simplifications in chapter 4. One limitation of our model has been the placement of the brine shrimp legs. Based on observations by Kemal Ozalp (UNC), the spacing of the brine shrimp legs is tighter than was possible in our paddle model. The decision to use wider spacing was made to accommodate the rigid paddles' full motion, which mirrored what was observed. A future model should implement the tight spacing found in nature by incorporating a joint into each paddle or by allowing the paddle to be flexible. In this way, we could accommodate both the tight spacing and the full range of motion. Our simulations are also two dimensional, and the effects of flow in the third dimension as the fluid is pulled and pushed in all directions between the appendages is likely important. Some recent work by Alex Hoover has considered three-dimensional metachronal paddling in a marine worm, and it is possible to modify this model for brine shrimp. Finally, the brine shrimp swimming appendages are bristled and likely act as porous structures. Additional realism could be incorporated into the model by using a porous description of the paddles.

APPENDIX A

A DERIVATION OF THE NAVIER-STOKES EQUATIONS

The state of a moving fluid in three dimensions is completely described if five quantities are known:

1. the three components of velocity
2. and two thermodynamic properties, typically density and pressure.

These five quantities may be determined from three physical conservation laws. We will derive them in both their integral and differential forms.

A.1 Conservation of Mass

Consider some reference volume Ω_0 of fluid, e.g. a small sphere of fluid. Let $\rho(\mathbf{x})$ be the density of the fluid at some point $\mathbf{x}$ in space. By integrating this density over the control volume of interest we obtain the total mass of fluid enclosed in Ω_0 as $\int \rho \, \mathrm{d}\Omega$. We next consider the flux across a surface element S of the control volume. The mass flowing through this surface element is given by $\rho \mathbf{u} \cdot \mathrm{d}S$, where $\mathbf{u}$ is the fluid velocity. By convention $\mathrm{d}S$ is taken along the *outward* normal, resulting in a positive flow corresponding to fluid exiting the control volume and a negative flow corresponding to mass entering our reference volume. Under the assumption that mass is neither created nor destroyed in our control volume, the change in mass must equal this flux taken over the whole control volume:

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, \mathrm{d}\Omega = - \oint_{S} \rho \mathbf{u} \cdot \mathrm{d}\mathbf{S}, \tag{A.1}$$

or more simply:

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, \mathrm{d}\Omega + \oint_{S} \rho \mathbf{u} \cdot \mathrm{d}\mathbf{S} = 0. \tag{A.2}$$

This is the integral form for the conservation of mass, also referred to as the equation of continuity.

To rewrite this in differential form we utilize Green's theorem to rewrite the surface integral as a volume intergal, allowing us to write the relation as a single integral equation. By commuting the

time integration with the volume integral this yields

$$\int_{\Omega} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right] \mathrm{d}\Omega = 0. \tag{A.3}$$

As the control volume is arbitrary, this ought to hold for any choice of control volume, thus giving the differential form of mass conservation as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{A.4}$$

In an incompressible fluid $\frac{\partial \rho}{\partial t} = 0$, in which case this reduces to

$$\nabla \cdot \mathbf{u} = 0. \tag{A.5}$$

A.2 Conservation of Momentum

We next consider the conservation of momentum for our fluid particles, which we will derive in differential form. In classical mechanics, conservation of momentum is typically expressed as $\mathbf{F} = m\mathbf{a}$, where $\mathbf{F}$ is the sum of all forces acting on a body and $\mathbf{a}$ is the acceleration of said body. For our purposes of dealing with a unit parcel of fluids, $m \equiv \rho$, and it remains to determine the acceleration. The velocity of a fluid particle $\mathbf{u}(\mathbf{x}, t)$ is a function of space and time, so by the chain rule

$$\begin{aligned} \frac{d}{dt}\mathbf{u} &= \sum_{i=1}^{n} \left(\frac{\partial \mathbf{u}}{\partial x_i} \frac{\partial x_i}{\partial t} \right) + \frac{\partial \mathbf{u}}{\partial t} \\ &= \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}. \end{aligned} \tag{A.6}$$

This is also known as the material derivative, sometimes written as $\frac{D\mathbf{u}}{Dt}$. So the right-hand side of Newton's second law can be expressed as $\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right)$. It remains to define the forces acting on our fluid.

The fluid experiences two kinds of forces - fluid forces and external body forces:

$$\mathbf{F} = \underbrace{\nabla \cdot \sigma}_{\text{internal forces}} + \mathbf{f}_{\text{external}} \tag{A.7}$$

where σ is a stress tensor representing the internal stresses in the fluid. This stress tensor can be divided into two terms, one representing volumetric stress and one representing shear stresses. This way we may write

$$\sigma = -pI + T, \tag{A.8}$$

where p is pressure, I is an identity matrix, and T is a tensor of the shear stresses experienced by our fluid parcel.

To determine the form of T, assumptions need to be made about the fluid. For our purposes we assume a so-called Newtonian fluid, defined as a fluid for which shear stress is proportional to the rate of deformation. In this case, T has components

$$\tau_{i,j} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \tag{A.9}$$

where μ is the proportionality constant. In this case $\nabla \cdot \sigma = -\nabla p + \mu \Delta \mathbf{u}$, where Δ is the Laplacian, and our conservation of momentum may be written as

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f}_{\text{ext}}. \tag{A.10}$$

A.3 Putting it Together

Combining our results regarding conservation of momentum and conservation of mass, we arrive at the complete Navier-Stokes equations used in the simulations in this thesis, which describe the motion of an incompressible, viscous fluid.

$$\begin{aligned} \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) &= -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f}_{\text{ext}} \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{A.11}$$

APPENDIX B

DETAILS ON THE IMMERSED BOUNDARY METHOD

The two-dimensional formulation of the immersed boundary (IB) method used in chapter 2 to study flow through dragonfly wing veins and lymphatic capillaries and around starfish is provided below. *IB2d* [64, 38, 39] and *IBAMR* [65] are the two open-source implementations that were used for these studies. For a full review of the immersed boundary method, please see Peskin [4].

B.1 Governing Equations of IB

The conservation of momentum equations that govern an incompressible and viscous fluid are listed below:

$$\rho\Big[\frac{\partial \mathbf{u}}{\partial t}(\mathbf{x},t) + (\mathbf{u}(\mathbf{x},t)\cdot\nabla)\mathbf{u}(\mathbf{x},t)\Big] = -\nabla p(\mathbf{x},t) + \mu\Delta\mathbf{u}(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t) \tag{B.1}$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t) = 0 \tag{B.2}$$

where $\mathbf{u}(\mathbf{x},t)$ is the fluid velocity, $p(\mathbf{x},t)$ is the pressure, $\mathbf{f}(\mathbf{x},t)$ is the force per unit area applied to the fluid by the immersed boundary, ρ and μ are the fluid's density and dynamic viscosity, respectively. The independent variables are the time t and the position $\mathbf{x}$. The variables $\mathbf{u}, p$, and $\mathbf{f}$ are all written in an Eulerian frame on the fixed Cartesian mesh, $\mathbf{x}$.

The interaction equations, which handle all communication between the fluid (Eulerian) grid and curvilinear mesh describing the immersed boundary (Lagrangian grid) are given by the following two integral equations:

$$\mathbf{f}(\mathbf{x},t) = \int \mathbf{F}(s,t)\delta\left(\mathbf{x}-\mathbf{X}(s,t)\right)ds \tag{B.3}$$

$$\mathbf{U}(s,t) = \int \mathbf{u}(\mathbf{x},t)\delta\left(\mathbf{x}-\mathbf{X}(s,t)\right)d\mathbf{x} \tag{B.4}$$

where $\mathbf{F}(s,t)$ is the force per unit length applied by the boundary to the fluid as a function of Lagrangian position, s, and time, t, $\delta(\mathbf{x})$ is a three-dimensional delta function, and $\mathbf{X}(s,t)$ gives the Cartesian coordinates at time t of the material point labeled by the Lagrangian parameter, s. The Lagrangian forcing term, $\mathbf{F}(s,t)$, gives the deformation forces along the boundary at the Lagrangian parameter, s. Equation (B.3) applies this force from the immersed boundary to the fluid through

the external forcing term in Equation (B.1). Equation (B.4) moves the boundary at the local fluid velocity. This enforces the no-slip condition. Each integral transformation uses a two-dimensional Dirac delta function kernel, δ, to convert Lagrangian variables to Eulerian variables and vice versa.

The way deformation forces are computed, e.g., the forcing term, $\mathbf{F}(s,t)$, in the integrand of Equation (B.3), is specific to the application. To hold the geometry nearly rigid, all of the Lagrangian points along the immersed boundary were tethered to target points. This has the effect of holding the boundary in place through a penalty forcing term where the force applied to the fluid is proportional to the difference between the actual location of the boundary and the desired location. In this model, the target force penalty term took the following form,

$$\mathbf{F}(s,t) = k_{targ}\left(\mathbf{Y}(s,t) - \mathbf{X}(s,t)\right), \tag{B.5}$$

where k_{targ} is a stiffness coefficient and $\mathbf{Y}(s,t)$ is the prescribed position of the target boundary. Note that $\mathbf{Y}(s,t)$ is a function of both the Lagrangian parameter, s, and time, t; however, in this model k_{targ} was chosen to be very large to minimize movement of the boundary.

For the case of the dragonfly wings, another penalty forcing term was used to prescribe the inflow conditions into the wing veins. This penalty force was applied directly onto the Eulerian (fluid) grid. The penalty force was proportional to the difference between the local fluid velocity and the desired fluid velocity and is given as

$$\mathbf{f}_{inflow} = k_{flow}\left(\mathbf{u}(\mathbf{x},t) - \mathbf{u}_{flow}(\mathbf{x},t)\right), \tag{B.6}$$

where k_{flow} is the penalty-strength coefficient, and $\mathbf{u}_{flow}(\mathbf{x},t)$ is the desired background flow profile as in [159]. For the simulations involving hemolymph flow through wing veins, we enforce the following parabolic inflow into the wing vein along the x-direction,

$$\mathbf{u}_{flow}(\mathbf{x},t) = \begin{cases} -U_{max}\tanh(2t)\left(\frac{(MP+w/2-y)(MP-w/2-y)}{w^2/4}\right) & \text{if inside prescribed region} \\ 0 & \text{elsewhere} \end{cases}, \tag{B.7}$$

where U_{max} is the desired max peak velocity in the parabolic inflow, MP is the midpoint of the vein and w is the width of the vein. Note that a hyperbolic tangent is used to ramp up the inflow during

the course of the simulation.

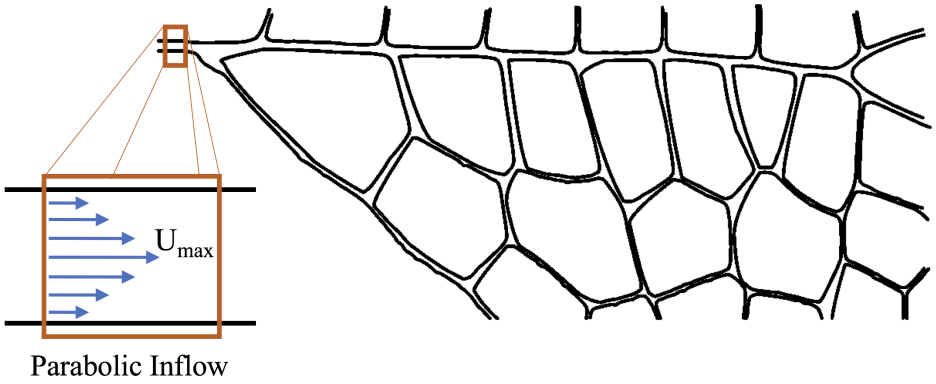

Figure B.1: Illustrating the subset of the insect vein geometry where the prescribed inflow condition is enforced.

Similarly, for the simulations of oscillatory flow past one or more starfish, we enforce the following oscillatory parabolic inflow into the starfish channel in the x-direction,

$$\mathbf{u}_{flow}(\mathbf{x},t) = \begin{cases} -U_{max}\sin(2\pi ft)\left(\frac{(MP+w/2-y)(MP-w/2-y)}{w^2/4}\right) & \text{if inside prescribed region} \\ 0 & \text{elsewhere} \end{cases}, \tag{B.8}$$

where f is the frequency of pulsation (set to 2 Hz) and the other parameters are analogous to before except with w and MP being the width and midpoint of the channel, respectively.

Using a regularized delta function as the kernel in the interaction equations given by Eqs.(B.3-B.4) makes the immersed boundary method relatively easy to implement and flexible. To approximate these integrals, a discretized (and regularized) delta function was used. In this paper, we use one described in [4], e.g., $\delta_h(\mathbf{x})$,

$$\delta_h(\mathbf{x}) = \frac{1}{h^3}\phi\left(\frac{x}{h}\right)\phi\left(\frac{y}{h}\right)\phi\left(\frac{z}{h}\right), \tag{B.9}$$

where $\phi(r)$ is defined as

$$\phi(r) = \begin{cases} \frac{1}{8}(3-2|r|+\sqrt{1+4|r|-4r^2}), & 0 \le |r| < 1 \\ \frac{1}{8}(5-2|r|+\sqrt{-7+12|r|-4r^2}), & 1 \le |r| < 2 \\ 0 & 2 \le |r|. \end{cases} \tag{B.10}$$

B.2 Numerical Algorithm

For the wing vein and starfish examples that use IB2d, we impose periodic and no slip boundary conditions on a rectangular domain. To solve Equations (B.1), (B.2), (B.3) and (B.4) we need to update the velocity, pressure, position of the boundary, and force acting on the boundary at time $n+1$ using data from time n. The IB does this in the following steps [4], with an additional step (*4b*) for IBAMR [160, 65]:

Step 1: Find the force density, $\mathbf{F}^n$, on the immersed boundary from the current boundary configuration, $\mathbf{X}^n$.

Step 2: Use Equation (B.3) to spread this boundary force from the Lagrangian boundary mesh to the Eulerian fluid lattice points.

Step 3: Solve the Navier-Stokes equations, Equations (B.1) and (B.2), on the Eulerian grid. Doing so, we are updating $\mathbf{u}^{n+1}$ and p^{n+1} from $\mathbf{u}^n$, p^n, and $\mathbf{f}^n$. Note that a staggered grid projection scheme is used to perform this update.

Step 4:

4a. Update the material positions, $\mathbf{X}^{n+1}$, using the local fluid velocities, $\mathbf{U}^{n+1}$, using $\mathbf{u}^{n+1}$ and Equation (B.4).

4b. (*IBAMR* only) Refine Eulerian grid in areas of the domain that contain an immersed structure or where the vorticity exceeds a predetermined threshold, if on a selected time-step for adaptive mesh refinement.

APPENDIX C

Background on Bézier Curves

Bézier curves are a type of interpolating polynomial known as a spline. An nth degree Bézier polynomial may conveniently be written as a sum of $n+1$ weighted control points $\mathbf{P}_i$. The weights are known as Bernstein basis polynomials and take the form

$$b_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i.$$

A curve $\boldsymbol{\gamma}(t)$ may then be written as

$$\boldsymbol{\gamma}(t) = \sum_{i=0}^{n} \mathbf{P}_i \, b_{i,n}(t).$$

The parameter t is defined to be on the closed interval $[0,1]$. The derivative of a Bézier curve is itself a Bézier curve. Specifically, the first derivative is given by

$$\boldsymbol{\gamma}'(t) = n \sum_{i=0}^{n-1} (\mathbf{P}_{i+1} - \mathbf{P}_i) \, b_{i,n-1}(t).$$

Paths are modeled as a curve $\boldsymbol{\Gamma}(s)$ that is at least a C^0 sequence of curves, where $s = [0,1]$ is a parameter used to map to the individual Bézier curves that make up the path. If $\mathbf{P}_i^j$ is the ith control point of the jth curve in $\boldsymbol{\Gamma}$, then C^0 continuity translates into the requirement that

$$\mathbf{P}_n^j = \mathbf{P}_0^{j+1}.$$

To achieve C^1 continuity, we additionally require that

$$\mathbf{P}_n^j - \mathbf{P}_{n-1}^j = \mathbf{P}_1^{j+1} - \mathbf{P}_0^{j+1}.$$

Perhaps the most common Bézier curve in applications is the cubic Bézier which takes the explicit form

$$\boldsymbol{\gamma}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3.$$

The explicit derivative of the cubic Bézier is given by

$$\boldsymbol{\gamma}'(t) = 3(1-t)^2(\mathbf{P}_1 - \mathbf{P}_0) + 6(1-t)t(\mathbf{P}_2 - \mathbf{1}) + 3t^2(\mathbf{P}_3 - \mathbf{P}_2).$$

To rescale a curve from one domain V to another domain U, an affine transform of the control points is sufficient. In the particular case $U, V \subset \mathbb{R}^2$ this transform may be represented as a simple matrix operator $A : \mathbb{R}^3 \to \mathbb{R}^3$ by representing a point $\mathbf{p} \in U, \mathbf{q} \in V$ in the form $(x, y, 1)^T$. Scaling and translating of control points may be achieved by the function

$$\mathbf{q} = \mathbf{A}\mathbf{p} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}. \tag{C.1}$$

For our particular use case we want to map a point P_i from the SVG coordinate system $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ to the coordinate system $[0, L_x] \times [0, L_y]$ used in the simulations. Note that the origin of the SVG coordinates is in the upper left-hand corner of the image, while the coordinate system used in IB2d and IBAMR has its origin in the lower left hand corner. Accounting for this and letting $w = x_{\max} - x_{\min}$ and $h = y_{\max} - y_{\min}$ our operator will be defined as

$$A = \begin{bmatrix} \frac{L_x}{w} & 0 & -\frac{L_x}{w} x_{\min} \\ 0 & -\frac{L_y}{h} & L_y \\ 0 & 0 & 1 \end{bmatrix} \tag{C.2}$$

REFERENCES

[1] J. P. McVey, "Reef0297.jpg." `https://commons.wikimedia.org/wiki/File:Reef0297.jpg`, 2007. NOAA Sea Grant Program: The Coral Kingdom Collection; Accessed: 2019-06-25.

[2] P. Kratochvil, "Insect wing structure: macro photo of a dragonfly wing structure," 2013. [Camera: Canon EOS 5D Mark II 1/160s, f 16.0, ISO 100, 100 mm; uploaded September 26, 2013; accessed February 22, 2018].

[3] L. A. Miller, A. Santhanakrishnan, S. K. Jones, C. Hamlet, K. Mertens, and L. Zhu, "Reconfiguration and the reduction of vortex-induced vibrations in broad leaves," *Journal of Experimental Biology*, vol. 215, pp. 2716–2727, 2012.

[4] C. S. Peskin, "The immersed boundary method," *Acta Numerica*, vol. 11, pp. 479–517, 2002.

[5] G. Hou, J. Wang, and A. Layton, "Numerical methods for fluid-structure interaction - a review," *Communications in Computational Physics*, vol. 12, no. 2, pp. 337–377, 2012.

[6] C. Peskin, "Flow patterns around heart valves: a numerical method," *Journal of Computational Physics*, vol. 10(2), pp. 252–271, 1972.

[7] D. McQueen and C. S. Peskin, "Shared-memory parallel vector implementation of the immersed boundary method for the computation of blood flow in the beating mammalian heart," *The Journal of Supercomputing*, vol. 11, pp. 213–236, 1997.

[8] B. E. Griffith, X. Luo, D. M. McQueen, and C. S. Peskin, "Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method," *International Journal of Applied Mechanics*, vol. 1(1), pp. 137–177, 2009.

[9] E. Tytell, C. Hsu, and L. Fauci, "The role of mechanical resonance in the neural control of swimming in fishes," *Zoology*, vol. 117, pp. 48–56, 2014.

[10] E. Tytell, C. Hsu, T. Williams, A. Cohen, and L. Fauci, "Interactions between internal forces, body stiffness, and fluid environment in a neuromechanical model of lamprey swimming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 19832–19837, 2010.

[11] L. M. Crowl and A. L. Fogelson, "Computational model of whole blood exhibiting lateral platelet motion induced by red blood cells," *International Journal of Numerical Methods In Biomedical Engineering*, vol. 26, pp. 471–487, 2009.

[12] L. M. Crowl and A. L. Fogelson, "Analysis of mechanisms for platelet near-wall excess under arterial blood flow conditions," *Journal of Fluid Mechanics*, vol. 676, pp. 348–375, 2011.

[13] L. A. Miller and C. S. Peskin, "When vortices stick: an aerodynamic transition in tiny insect flight," *Journal of Experimental Biology*, vol. 207, pp. 3073–3088, 2004.

[14] L. A. Miller and C. S. Peskin, "A computational fluid dynamics study of 'clap and fling' in the smallest insects," *Journal of Experimental Biology*, vol. 208, pp. 195–212, 2005.

[15] J. E. Samson, L. A. Miller, D. Ray, R. Holzman, U. Shavit, and S. Khatri, "A novel mechanism of mixing by pulsing corals," *Journal of Experimental Biology*, vol. 222, no. 15, 2019.

[16] D. W. Murphy, D. Adhikari, D. R. Webster, and J. Yen, "Underwater flight by the planktonic sea butterfly," *Journal of Experimental Biology*, vol. 219, no. 4, pp. 535–543, 2016.

[17] A. Dauptain, J. Favier, and A. Bottaro, "Hydrodynamics of ciliary propulsion," *Journal of Fluids and Structures*, vol. 24, no. 8, pp. 1156–1165, 2008.

[18] M. Dickinson, "Insect flight," *Current Biology*, vol. 16, no. 9, pp. R309–R314, 2006.

[19] S. Vogel, *Life in moving fluids : the physical biology of flow.* Princeton, N.J.: Princeton University Press, 1994.

[20] R. Dudley, *The biomechanics of insect flight : form, function, evolution.* Princeton University Press, 1999.

[21] S. P. Sane, "The aerodynamics of insect flight," *Journal of experimental biology*, vol. 206, no. 23, pp. 4191–4208, 2003.

[22] A. Santhanakrishnan, S. K. Jones, W. B. Dickson, M. P. M, V. T. Kasoju, M. H. Dickinson, and L. A. Miller, "Flow structure and force generation on flapping wings at low reynolds numbers relevant to the flight of tiny insects," *Fluids*, vol. 3, p. 45, 2018.

[23] S. K. Jones, Y. J. J. Yun, T. L. Hedrick, B. E. Griffith, and L. A. Miller, "Bristles reduce the force required to 'fling' wings apart in the smallest insects," *Journal of Experimental Biology*, vol. 219, no. 23, pp. 3759–3772, 2016.

[24] T. Weis-Fogh, "Quick estimates of flight fitness in hovering animals, including novel mechanisms for lift production," *Journal of Experimental Biology*, vol. 59, pp. 169–230, 1973.

[25] C. P. Ellington, "The aerodynamics of hovering insect flight. iv. aerodynamic mechanisms," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 305, no. 1122, pp. 79–113, 1984.

[26] L. A. Miller and C. S. Peskin, "Flexible clap and fling in tiny insect flight," *Journal of Experimental Biology*, vol. 212, pp. 3076–3090, 2009.

[27] A. Santhanakrishnan, A. K. Robinson, S. Jones, A. A. Low, S. Gadi, T. L. Hedrick, and L. A. Miller, "Clap and fling mechanism with interacting porous wings in tiny insect flight," *Journal of Experimental Biology*, vol. 217, no. 21, pp. 3898–3909, 2014.

[28] M. J. Lighthill, "On the weis-fogh mechanism of lift generation," *Journal of Fluid Mechanics*, vol. 60, no. 1, pp. 1–17, 1973.

[29] T. Maxworthy, "Experiments on the weis-fogh mechanism of lift generation by insects in hovering flight. part 1. dynamics of the 'fling'," *Journal of Fluid Mechanics*, vol. 93, no. 1, pp. 47–63, 1979.

[30] D. Kolomenskiy, H. K. Moffatt, M. Farge, and K. Schneider, "The lighthill-weis-fogh clapflingsweep mechanism revisited," *Journal of Fluid Mechanics*, vol. 676, pp. 572–606, 2011.

[31] F.-O. Lehmann, S. P. Sanjay, and M. Dickinson, "The aerodynamic effects of wingwing interaction in flapping insect wings," *Journal of Experimental Biology*, vol. 208, pp. 3075–3092, 2005.

[32] M. P. Ford, H. K. Lai, M. Samaee, and A. Santhanakrishnan, "Hydrodynamics of metachronal paddling: effects of varying reynolds number and phase lag," *Royal Society Open Science*, vol. 6, no. 10, p. 191387, 2019.

[33] S. Granzier-Nakajima, R. D. Guy, and C. Zhang-Molina, "A numerical study of metachronal propulsion at low to intermediate reynolds numbers," *Fluids*, vol. 5, no. 2, p. 86, 2020.

[34] S. K. Jones, R. Laurenza, T. L. Hedrick, B. E. Griffith, and L. A. Miller, "Lift- vs. drag-based for vertical force production in the smallest flying insects," *Journal of Theoretical Biology*, vol. 384, pp. 105–120, 2015.

[35] A. P. Hoover and L. A. Miller, "A numerical study of the benefits of driving jellyfish bells at their natural frequency," *Journal of Theoretical Biology*, vol. 374, pp. 13–25, 2015.

[36] G. Hershlag and L. A. Miller, "Reynolds number limits for jet propulsion: a numerical study of simplified jellyfish," *Journal of Theoretical Biology*, vol. 285, pp. 84–95, 2011.

[37] E. Jung and C. Peskin, "2-D simulations of valveless pumping using immersed boundary methods," *SIAM Journal on Scientific Computing*, vol. 23, pp. 19–45, 2001.

[38] N. A. Battista, W. C. Strickland, and L. A. Miller, "IB2d: a Python and MATLAB implementation of the immersed boundary method," *Bioinspiration & Biomimetics*, vol. 12(3), p. 036003, 2017.

[39] N. A. Battista, W. C. Strickland, A. Barrett, and L. A. Miller, "IB2d Reloaded: a more powerful Python and MATLAB implementation of the immersed boundary method," *Mathematical Methods in Applied Sciences*, vol. 41, pp. 8455–8480, 2018.

[40] L. Zhu, G. He, S. Wang, L. A. Miller, X. Zhang, Q. You, and S. Fang, "An immersed boundary method by the lattice Boltzmann approach in three dimensions," *Computers and Mathematics with Applications*, vol. 61, pp. 3506–3518, 2011.

[41] L. Zhu and C. S. Peskin, "Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method," *Journal of Computational Physics*, vol. 179, pp. 452–468, July 2002.

[42] J. Ryu, S. G. Park, B. Kim, and H. Jinsung, "Flapping dynamics of an inverted flag in a uniform flow," *Journal of Fluids and Structures*, vol. 57, pp. 159–169, 2015.

[43] A. J. Baird, T. King, and L. A. Miller, "Numerical study of scaling effects in peristalsis and dynamic suction pumping," *Biological Fluid Dynamics: Modeling, Computations, and Applications*, vol. 628, pp. 129–148, 2014.

[44] L. D. Waldrop and L. A. Miller, "Large-amplitude, short-wave peristalsis and its implications for transport," *Biomechanics and Modeling in Mechanobiology*, pp. 1–14, 2015.

[45] Y. Kim and C. S. Peskin, "Penalty immersed boundary method for an elastic boundary with mass," *Physics of Fluids*, vol. 19, p. 053103, 2007.

[46] J. M. Stockie, "Modelling and simulation of porous immersed boundaries," *Computers and Structures*, vol. 87, pp. 701–709, 2009.

[47] Y. Kim and C. S. Peskin, "2D parachute simulation by the immersed boundary method," *SIAM Journal on Scientific Computing*, vol. 28, pp. 2294–2312, 2006.

[48] D. S. Lee, M. Y. Ha, S. J. Kim, and H. S. Yoon, "Application of immersed boundary method for flow over stationary and oscillating cylinders," *Journal of Mechanical Science and Technology*, vol. 20(6), pp. 849–863, 2006.

[49] A. Pinelli, I. Z. Nagavi, U. Piomelli, and J. Favier, "Immersed-boundary methods for general finite-difference and finite-volume Navier-Stokes solvers," *Journal of Computational Physics*, vol. 229(24), pp. 9073–9091, 2010.

[50] D. C. Lo, C. P. Lee, and I. F. Lin, "An efficient immersed boundary method for fluid flow simulations with moving boundaries," *Applied Mathematics and Computation*, vol. 328(1), pp. 312–337, 2018.

[51] R. Campregher, J. Militzer, S. S. Mansur, N. Silveira, and A. da Silveira Neto, "Computations of the flow past a still sphere at moderate reynolds numbers using an immersed boundary method," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 31(4), pp. 333–352, 2009.

[52] W. C. Strickland, L. A. Miller, A. Santhanakrishnan, C. Hamlet, and V. P. N. A. Battista, "Three-dimensional low Reynolds number flows near biological filtering and protective layers," *Fluids*, vol. 2(4), p. 62, 2017.

[53] C. S. Peskin and D. M. McQueen, "Fluid dynamics of the heart and its valves," in *Case Studies in Mathematical Modeling: Ecology, Physiology, and Cell Biology* (F. R. Adler, M. A. Lewis, and J. C. Dalton, eds.), ch. 14, pp. 309–338, New Jersey: Prentice-Hall, 1996.

[54] N. A. Battista, A. N. Lane, J. Liu, and L. A. Miller, "Fluid dynamics of heart development: effects of trabeculae and hematocrit," *Mathematical Medicine & Biology*, vol. 35(4), pp. 493–516, 2018.

[55] T. J. Wilson, "Simultaneous untangling and smoothing of hexahedral meshes (M.S. Thesis)," *Universitat Politecènica de Catalunya*, 2011.

[56] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference* (V. Scarano, R. D. Chiara, and U. Erra, eds.), pp. 129–136, The Eurographics Association, 2008.

[57] C. Geuzaine and J. F. Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *International Journal of Numerical Methods in Engineering*, vol. 79(11), pp. 1309–1331, 2009.

[58] S. Hang, "TetGen, a Delaunay-based quality tetrahedral mesh generator," *ACM Transactions on Mathematical Software*, vol. 41(2), pp. 1–36, 2015.

[59] M. K. Berens, I. D. Flintoft, and J. F. Dawson, "Structured mesh generation: open-source automatic nonuniform mesh generation for FDTD simulation," *IEEE Antennas & Propagation Magazine*, vol. 58(3), pp. 45–55, 2016.

[60] Argus Holdings, LTD, "Argus ONE: open numerical environments," 2015.

[61] C. I. Voss, D. Boldt, and A. M. Shapiro, "A graphical-user interface for the U.S. Geological Survey's SUTRA code using argus ONE (for simulation of variable-density saturated-unsaturated ground-water flow with solute or energy transport)," *U.S. Geological Survey Open-File Report*, vol. 1, pp. 97–421, 1997.

[62] N. A. Battista, "Fluid-Structure Interaction for the classroom: interpolation, hearts, and swimming!," *arXiv*, 2018.

[63] N. A. Battista and M. S. Mizuhara, "Fluid-Structure Interaction for the classroom: speed, accuracy, convergence, and jellyfish!," *arXiv*, 2019.

[64] N. A. Battista, A. J. Baird, and L. A. Miller, "A mathematical model and Matlab code for muscle-fluid-structure simulations," *Integrative and Comparative Biology*, vol. 55(5), pp. 901–911, 2015.

[65] B. E. Griffith, "An adaptive and distributed-memory parallel implementation of the immersed boundary (IB) method." Accessed: 2019-06-18.

[66] C. Hamlet and L. A. Miller, "Feeding currents of the upside-down jellyfish in the presence of background flow," *Bulletin of Mathematical Biology*, vol. 74(11), pp. 2547–2569, 2012.

[67] C. Zhang, R. D. Guy, B. Mulloney, Q. Zhang, and T. J. Lewis, "Neural mechanism of optimal limb coordination in crustacean swimming," *Proceedings of the National Academy of Sciences*, vol. 111, no. 38, pp. 13840–13845, 2014.

[68] C. Hamlet, W. Strychalski, and L. Miller, "Dynamics of ballistic strategies in nematocyst firing," *Fluids*, vol. 5, no. 1, p. 20, 2020.

[69] "SVG Optimizer is a Nodejs-based tool for optimizing SVG vector graphics files." `https://github.com/svg/svgo`. Accessed: 2019-06-25.

[70] J. Archibald, "SVGOMG is SVGO's missing GUI, aiming to expose the majority, if not all the configuration options of SVGO." `https://jakearchibald.github.io/svgomg/`. Accessed: 2019-06-25.

[71] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, June 2015.

[72] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[73] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[74] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Englewood, Cliffs, NJ: Prentice Hall, 2 ed., 2002.

[75] A. Bovick, *Handbook of image and video processing*. New York, NY: Academic Press, 2000.

[76] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, ICCV '98, (Washington, DC, USA), pp. 839–846, IEEE, IEEE Computer Society, 1998.

[77] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[78] G. T. Berge, O. Granmo, T. O. Tveit, M. Goodwin, L. Jiao, and B. V. Matheussen, "Using the Tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications," *CoRR*, vol. abs/1809.04547, 2018.

[79] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: learning image restoration without clean data," *CoRR*, vol. abs/1803.04189, 2018.

[80] X. Wang, D. Huang, and H. Xu, "An efficient local Chan-Vese model for image segmentation," *Pattern Recognition*, vol. 43, no. 3, pp. 603–618, 2010.

[81] "GNU Image Manipulation Program." `https://www.gimp.org/`. Accessed: 2019-08-21.

[82] C. D. Bertram, C. Macaskill, and J. E. J. Moore, "An improved model of an actively contracting lymphatic vessel composed of several lymphangions: pumping characteristics," *arXiv preprint arXiv:1512.01269*, 2015.

[83] J. B. Dixon, S. T. Greiner, A. A. Gashev, G. L. Cote, J. E. Moore Jr., and D. C. Zawieja, "Lymph flow, shear stress, and lymphocyte velocity in rat mesenteric prenodal lymphatics," *Microcirculation*, vol. 13, no. 7, pp. 597–610, 2006.

[84] N. A. Battista, D. R. Douglas, A. N. Lane, L. A. Samsa, J. Liu, and L. A. Miller, "Vortex dynamics in trabeculated embryonic ventricles," *Journal of Cardiovascular Development and Disease*, vol. 6, p. 6, 2019.

[85] J. Liu, M. Bressan, D. Hassel, J. Huisken, D. Staudt, K. Kikuchi, K. Poss, T. Mikawa, and D. Stainier, "A dual role for ErbB2 signaling in cardiac trabeculation," *Development*, vol. 137, pp. 3867–3875, 2010.

[86] R. Cortez, "The method of regularized Stokeslets," *SIAM Journal on Scientific Computing*, vol. 23(4), pp. 1204–1225, 2001.

[87] Z. Li, "An overview of the immersed interface method and its applications," *Taiwanese Journal of Mathematics*, vol. 7(1), pp. 1–49, 2003.

[88] O. Ubbink and R. I. Issa, "Method for capturing sharp fluid interfaces on arbitrary meshes," *Journal of Computational Physics*, vol. 153, pp. 26–50, 1999.

[89] H. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna, "A sharp interface cartesian grid method for simulating flows with complex moving boundaries," *Journal of Computational Physics*, vol. 20, pp. 345–380, 2001.

[90] R. Cortez and M. Minion, "The blob projection method for immersed boundary problems," *Journal of Computational Physics*, vol. 161, pp. 428–453, 2000.

[91] L. D. Waldrop, L. A. Miller, and S. Khatri, "A tale of two antennules: the performance of crab odour-capture organs in air and water," *Journal of the Royal Society, Interface*, vol. 13, 2016.

[92] P. Lee, B. E. Griffith, and C. S. Peskin, "The immersed boundary method for advection-electrodiffusion with implicit timestepping and local mesh refinement," *Journal of Computational Physics*, vol. 229, pp. 5208–5227, 2010.

[93] N. A. Battista, "IB2d video tutorials!." `https://github.com/nickabattista/IB2d/`. Accessed: 2019-06-18.

[94] J. G. Miles and N. A. Battista, "Don't be jelly: Exploring effective jellyfish locomotion," *arXiv*, 2019.

[95] J. G. Miles and N. A. Battista, "Naut your everyday jellyfish model: exploring how tentacles and oral arms impact locomotion," *Fluids*, vol. 4(3), p. 169, 2019.

[96] F. Pallasdies, S. Goedeke, W. Braun, and R. Memmesheimer, "From single neurons to behavior in the jellyfish *Aurelia aurita*," *biorXiv*, 2019.

[97] N. Stork, "How many species of insects and other terrestrial arthropods are there on earth?," *Annual review of entomology*, vol. 63, pp. 31–45, 2018.

[98] J. G. Morse and M. S. Hoddle, "Invasion biology of thrips," *Annual Review of Entomology*, vol. 51, pp. 67–89, 2006.

[99] D. L. J. Quicke, *Parasitic Wasps*. London, UK: Chapman and Hall Ltd, 1997.

[100] D. E. Ullman, R. Meideros, L. R. Campbell, A. E. Whitfield, J. L., and T. L. German, "Thrips as vectors of tospoviruses," *Advances in Botanical Research*, vol. 36, pp. 113–140, 2002.

[101] D. R. Jones, "Plant viruses transmitted by thrip," *European journal of plant pathology*, vol. 113, pp. 119–157, 2005.

[102] B. J. Crespi, D. A. Carmean, and T. W. Chapman, "Ecology and evolution of galling thrips and their allies," *Annual Review of Entomology*, vol. 42, pp. 51–71, 1997.

[103] J. M. Palmer, D. V. R. Reddy, J. A. Wightman, and G. V. R. Rao, "New information on the thrips vectors of tomato spotted wilt virus in groundnut crops in india," *International Arachis Newsletter*, vol. 7, pp. 24–25, 1990.

[104] A. Austin and M. Dowton, *Hymenoptera: Evolution, Biodiversity and Biological Control*. Collingwood: Csiro Publishing, 2000.

[105] C. Ellington, "The novel aerodynamics of insect flight: applications to micro-air vehicles," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3439–3448, 1999.

[106] J. Bowden and C. G. Johnson, "Migrating and other terrestrial insects at sea," in *Marine insects* (L. Cheng, ed.), pp. 97–117, Amsterdam: Elsevier /North-Holland, 1976.

[107] S. Vogel, *Comparative biomechanics: life's physical world*. Princeton: Princeton University Press, 2013.

[108] Z. J. Wang, "Dissecting insect flight," *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 183–210, 2005.

[109] T. L. Hedrick, S. A. Combes, and L. A. Miller, "Recent developments in the study of insect flight," *Canadian Journal of Zoology*, vol. 93, no. 12, pp. 925–943, 2015.

[110] I. Cohen, S. C. Whitehead, and T. Beatus, "Fluid dynamics and control of insect flight," *Nature Reviews Physics*, vol. 1, pp. 638–639, 2019.

[111] C. P. Ellington, C. van den Berg, A. P. Willmott, and A. L. R. Thomas, "Leading-edge vortices in insect flight," *Nature*, vol. 384, pp. 626–630, 1996.

[112] J. M. Birch, W. B. Dickson, and M. H. Dickinson, "Force production and flow structure of the leading edge vortex on flapping wings at high and low reynolds numbers," *Journal of Experimental Biology*, vol. 207, pp. 1063–1072, 2004.

[113] J. M. Birch and M. H. Dickinson, "The influence of wingwake interactions on the production of aerodynamic forces in flapping flight," *Journal of Experimental Biology*, vol. 206, pp. 2257–2272, 2003.

[114] M. Dickinson, F.-O. Lehmann, and S. Sane, "Wing rotation and the aerodynamic basis of insect flight," *Science (New York, N.Y.)*, vol. 284, pp. 1954–60, 07 1999.

[115] X. Cheng and M. Sun, "Revisiting the clap-and-fling mechanism in small wasp encarsia formosa using quantitative measurements of the wing motion," *Physics of Fluids*, vol. 31, no. 10, p. 101903, 2019.

[116] V. T. Kasoju, C. L. Terrill, M. P. Ford, and A. Santhanakrishnan, "Leaky flow through simplified physical models of bristled wings of tiny insects during clap and fling," *Fluids*, vol. 3, p. 44, 2018.

[117] E. de Langre, A. Gutierrez, and J. Cossé, "On the scaling of drag reduction by reconfiguration in plants," *Comptes Rendus Mécanique*, vol. 340, no. 1, pp. 35–40, 2012.

[118] S. Vogel, "Drag and flexibility in sessile organisms," *American Zoologist*, vol. 24, no. 1, pp. 37–44, 1984.

[119] B. E. Griffith and X. Luo, "Hybrid finite difference/finite element immersed boundary method," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 33, no. 12, p. e2888, 2017.

[120] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 239–261, 2005.

[121] B. E. Griffith and N. A. Patankar, "Immersed methods for fluid-structure interaction," *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 421–448, 2020.

[122] A. P. Hoover, B. E. Griffith, and L. A. Miller, "Quantifying performance in the medusan mechanospace with an actively swimming three-dimensional jellyfish model," *Journal of Fluid Mechanics*, vol. 813, pp. 1112–1155, 2017.

[123] C. L. Hamlet, K. A. Hoffman, E. D. Tytell, and L. J. Fauci, "The role of curvature feedback in the energetics and dynamics of lamprey swimming: A closed-loop model," *PLoS Computational Biology*, vol. 14, p. e1006324, 2018.

[124] L. Feng, H. Gao, B. E. Griffith, S. A. Niederer, and X. Y. Luo, "Analysis of a coupled fluid-structure interaction model of the left atrium and mitral valve," *International journal for numerical methods in biomedical engineering*, vol. 35, p. e3254, 2019.

[125] S. Sherifova, G. Sommer, C. Viertler, P. Regitnig, T. Caranasos, M. A. Smith, B. E. Griffith, R. W. Ogden, and G. A. Holzapfel, "Failure properties and microstructure of healthy and aneurysmatic human thoracic aortas with a focus on the media," *Acta Biomater*, vol. 99, pp. 443–456, 2019.

[126] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin, "An adaptive, formally second order accurate version of the immersed boundary method," *Journal of Computational Physics*, vol. 223, pp. 10–49, 2007.

[127] A. L. Davis, A. P. Hoover, and L. A. Miller, "Lift and drag acting on the shell of the american horseshoe crab (*Limulus polyphemus*)," *Bulletin of Mathematical Biology*, vol. 81, pp. 3803–3822, 2019.

[128] C. P. Ellington, "The aerodynamics of hovering insect flight. iii. kinematics," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 305, no. 1122, pp. 41–78, 1984.

[129] M. P. Ford, V. T. Kasoju, M. G. Gaddam, and A. Santhanakrishnan, "Aerodynamic effects of varying solid surface area of bristled wings performing clap and fling," *Bioinspiration & Biomimetics*, vol. 14, no. 4, p. 046003, 2019.

[130] P. Broadley, M. R. Nabawy, M. K. Quinn, and W. J. Crowther, "Wing planform effects on the aerodynamic performance of insect-like revolving wings," in *AIAA AVIATION 2020 FORUM*, 2020.

[131] S. S. Bhat, J. Zhao, J. Sheridan, K. Hourigan, and M. C. Thompson, "Aspect ratio studies on insect wings," *Physics of Fluids*, vol. 31, no. 12, p. 121301, 2019.

[132] S. A. Combes and T. L. Daniel, "Flexural stiffness in insect wings i. scaling and the influence of wing venation," *Journal of Experimental Biology*, vol. 206, no. 17, pp. 2979–2987, 2003.

[133] S. Combes and T. Daniel, "Flexural stiffness in insect wings: Effects of wing venation and stiffness distribution on passive bending," *American Entomologist*, vol. 51, 03 2005.

[134] S. A. Combes and T. L. Daniel, "Flexural stiffness in insect wings ii. spatial distribution and dynamic wing bending," *Journal of Experimental Biology*, vol. 206, no. 17, pp. 2989–2997, 2003.

[135] F. O. Lehmann, S. Gorb, N. Nasir, and P. Schutzner, "Elastic deformation and energy loss of flapping fly wings," *Journal of Experimental Biology*, vol. 214, pp. 2949–2961, 2011.

[136] S. Sunada, H. Takashima, T. Hattori, K. Yasuda, and K. Kawachi, "Fluid-dynamic characteristics of a bristled wing," *Journal of Experimental Biology*, vol. 205, no. 17, pp. 2737–2744, 2002.

[137] G. Davidi and D. Weihs, "Flow around a comb wing in low-reynolds-number flow," *AIAA Journal*, vol. 50, pp. 249–253, 2012.

[138] S. H. Lee and D. Kim, "Aerodynamics of a translating comb-like plate inspired by a fairyfly wing," *Physics of Fluids*, vol. 29, p. 081902, 2017.

[139] Y. Z. Lyu, H. J. Zhu, and M. Sun, "Flapping-mode changes and aerodynamic mechanisms in miniature insects," *Phys Rev E.*, vol. 99, p. 012419, 2019.

[140] D. Barlow and M. A. Sleigh, "Water propulsion speeds and power output by comb plates of the ctenophore pleurobrachia pileus under different conditions," *Journal of Experimental Biology*, vol. 183, no. 1, pp. 149–164, 1993.

[141] Jana, S. H. Um, and S. Jung, "Paramecium swimming in capillary tube," *Physics of Fluids*, p. 041901, 2012.

[142] J. Elgeti and G. Gompper, "Emergence of metachronal waves in cilia arrays," *Proceedings of the National Academy of Sciences*, vol. 110, no. 12, pp. 4470–4475, 2013.

[143] G. F. Ricardo, R. J. Jones, P. L. Clode, and A. P. Negri, "Mucous secretion and cilia beating defend developing coral larvae from suspended sediments," *PLoS One*, vol. 28, p. e0162743, 2016.

[144] J. B. Lewis and W. S. Price, "Patterns of ciliary currents in atlantic reef corals and their functional significance," *Journal of Zoology*, vol. 178, no. 1, pp. 77–89, 1976.

[145] K. Y. Wan, S. K. Hürlimann, A. M. Fenix, R. M. McGillivary, T. Makushok, E. Burns, J. Y. Sheung, and W. F. Marshall, "Reorganization of complex ciliary flows around regenerating stentor coeruleus," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 375, no. 1792, p. 20190167, 2020.

[146] J. Ward, "Biodynamics of suspension-feeding in adult bivalve molluscs: Particle capture, processing, and fate," *Invertebrate Biology*, vol. 115, pp. 218–231, 1996.

[147] E. O. Campos, D. Vilhena, and R. L. Caldwell, "Pleopod rowing is used to achieve high forward swimming speeds during the escape response of odontodactylus havanensis (stomatopoda)," *Journal of Crustacean Biology*, vol. 32, no. 2, pp. 171–179, 2012.

[148] M. T. Downton and H. Stark, "Simulation of a model microswimmer," *Journal of Physics: Condensed Matter*, vol. 21, no. 20, p. 204101, 2009.

[149] S. E. Spagnolie and E. Lauga, "Hydrodynamics of self-propulsion near a boundary: predictions and accuracy of far-field approximations," *Journal of Fluid Mechanics*, vol. 700, no. 10, pp. 105–147, 2012.

[150] M. A. Sleigh and D. I. Barlow, *Aspects of Animal Movement*, ch. Metachronism and control of locomotion in animals with many propulsive structures., pp. 49–70. Cambridge University Press, 1980.

[151] D. W. Murphy, D. R. Webster, and S. K. et al., "Metachronal swimming in antarctic krill: gait kinematics and system design," *Marine Biology*, vol. 158, pp. 2541–2554, 2011.

[152] J. L. Lim and M. E. DeMont, "Kinematics, hydrodynamics and force production of pleopods suggest jet-assisted walking in the american lobster (homarus americanus)," *Journal of Experimental Biology*, vol. 212, no. 17, pp. 2731–2745, 2009.

[153] K. Kohlhage and J. Yager, "An analysis of swimming in remipede crustaceans," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 346, no. 1316, pp. 213–221, 1994.

[154] S. Alben, K. Spears, S. Garth, D. Murphy, and J. Yen, "Coordination of multiple appendages in drag-based swimming," *Journal of The Royal Society Interface*, vol. 7, no. 52, pp. 1545–1557, 2010.

[155] K. Taira and T. Colonius, "The immersed boundary method: A projection approach," *Journal of Computational Physics*, vol. 225, no. 2, pp. 2118–2137, 2007.

[156] B. Kallemov, A. Bhalla, B. Griffith, and A. Donev, "An immersed boundary method for rigid bodies," *Communications in Applied Mathematics and Computational Science*, vol. 11, no. 1, pp. 79–141, 2016.

[157] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, and e. a. B. Schmid, "Fiji: an open-source platform for biological-image analysis," *Nature methods*, vol. 9, no. 7, pp. 676–682, 2012.

[158] T. L. Hedrick, "Software techniques for two- and three-dimensional kinematic measurements of biological and biomimetic systems," *Bioinspiration & Biomimetics*, vol. 3, p. 034001, jul 2008.

[159] A. Santhanakrishnan, N. Nguyen, J. Cox, and L. A. Miller, "Flow within models of the vertebrate embryonic heart," *Journal of Theoretical Biology*, vol. 259, pp. 449–461, 2009.

[160] B. E. Griffith, "Simulating the blood-muscle-vale mechanics of the heart by an adaptive and parallel version of the immsersed boundary method (Ph.D. Thesis)," *Courant Institute of Mathematics, New York University*, 2005.