Title
High-Order Discontinuous Galerkin Fluid-Structure Interaction Methods

Author
Froehle, Bradley M.

Publication Date
2013

Peer reviewed|Thesis/dissertation

High-Order Discontinuous Galerkin Fluid-Structure Interaction Methods

by

Bradley Michael Froehle

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Per-Olof Persson, Chair
Professor Jon Wilkening
Professor Larry S. Karp

Fall 2013

High-Order Discontinuous Galerkin Fluid-Structure Interaction Methods

Abstract

High-Order Discontinuous Galerkin Fluid-Structure Interaction Methods

by

Bradley Michael Froehle

Doctor of Philosophy in Mathematics

University of California, Berkeley

Professor Per-Olof Persson, Chair

We present a high-order accurate scheme for fully coupled fluid-structure interaction problems. The fluid is discretized using a discontinuous Galerkin method on unstructured tetrahedral meshes, and the structure uses a high-order volumetric continuous Galerkin finite element method. Standard radial basis functions are used for the mesh deformation. The time integration is performed using a partitioned approach based on implicit-explicit Runge-Kutta methods. The resulting scheme fully decouples the implicit solution procedures for the fluid and the solid parts, which we perform using two separate efficient parallel solvers. We demonstrate up to fifth order accuracy in time on a non-trivial test problem, on which we also show that additional subiterations are not required. We solve a benchmark problem of a cantilever beam in a shedding flow, and show good agreement with other results in the literature.

In addition, we create several simulations which are motivated by real-world phenomena. First, we investigate flow around a thin membrane at high-angle of attack, demonstrating the ability of the leading edge of the membrane to align with the incident flow. Examples are provided in both two and three dimensions. Next, we consider biologically inspired flight, by investigating wing-like structures driven in a flapping motion in both two and three dimensions.

Finally, we demonstrate how the method may be used in acoustics problems, simulating a tuning fork in three dimensions. Here we accurately capture decay rates purely from the fluid-structure interaction and without any damping coefficients built into the structure model.

To My Wife

Because I couldn't have done it without you.

Contents

Contents ii

List of Figures iv

List of Tables vi

List of Algorithms vii

1 Introduction 1
- 1.1 Previous Work . . . 2
- 1.2 Overview . . . 3

2 Governing Equations 5
- 2.1 Compressible Navier-Stokes . . . 5
  - 2.1.1 Boundary Conditions . . . 6
  - 2.1.2 Isentropic & Isothermal Formulations . . . 8
- 2.2 Arbitrary Lagrangian Eulerian formulation . . . 8
- 2.3 Rigid Body Dynamics . . . 10
- 2.4 Neo-Hookean Elasticity Model . . . 11
  - 2.4.1 Quasi-Static Formulation . . . 12

3 Discretization 14
- 3.1 Fluid Spatial (Discontinuous Galerkin) . . . 14
- 3.2 Structure Spatial (Continuous Galerkin) . . . 16
  - 3.2.1 Quasi-Static Formulation . . . 17
- 3.3 Temporal Discretization (Runge-Kutta Methods) . . . 18
  - 3.3.1 Implicit-Explicit (IMEX) Runge-Kutta Schemes . . . 21
- 3.4 Implicit Solvers . . . 23
  - 3.4.1 Parallel Newton-Krylov Fluid Solvers . . . 23
  - 3.4.2 Sparse Direct Structure Solvers . . . 27

4 Fluid-Structure Interaction 29
- 4.1 Coupling . . . 29

4.1.1 Fluid-to-structure coupling . . . 30
4.1.2 Structure-to-fluid coupling . . . 31
4.2 Temporal Integrator . . . 37
4.3 Validation . . . 40
4.3.1 ALE / Expanding Pressure Wave . . . 40
4.3.2 FSI / Pitching and Heaving Airfoil . . . 43
4.3.3 FSI / Cantilever . . . 45

5 Membranes & Flapping Wings **49**
5.1 Membrane, 2D . . . 50
5.2 Membrane, 3D . . . 52
5.3 Flapping Wing, 2D . . . 52
5.4 Flapping Wing, 3D . . . 55

6 Acoustics **59**
6.1 Tuning Fork . . . 59
6.1.1 The Model . . . 59
6.1.2 Results . . . 61
6.1.3 Conclusions . . . 68

7 Conclusions & Future Work **70**

Bibliography **71**

List of Figures

4.1 Two meshes which are face-wise matching (left) and non-matching (right) along the interface, shown as a dashed line. 30
4.2 Mesh deformation using radial basis function interpolation. The radial basis function is as specified in eq. (4.9) with r equal to the edge length of the outer square. 34
4.3 Mesh deformation using a quasi-static non-linear elasticity method. 35
4.4 An expanding pressure wave on a deforming mesh using a linear deformation (top) and an isoparametric deformation (bottom). In each case the solution was represented using elements of polynomial degree 5. (Pressure). 42
4.5 Spatial convergence in the discrete maximum error at the final simulation time for an expanding pressure wave for meshes of polynomial degree $p = 1$ to 5. The deformation is either linear (P1) or isoparametric (Full P). 43
4.6 Schematic for the pitching and heaving airfoil. 44
4.7 The airfoil at various times. The pivot location of the airfoil is smoothly moved upwards between time $t = 0$ and $t = 1$ and held fixed between $t = 1$ and $t = 2$. (Mach number). 45
4.8 The relative error in angle of attack, $\|\theta(t) - \theta_{exact}(t)\|_\infty / \|\theta_{exact}(t)\|_\infty$, as a function of timestep. The ARK3, ARK4, and ARK5 schemes achieve the expected order of accuracy. Solving the fully-coupled ("FC-") system using the implicit method from the IMEX scheme shows a negligible increase in accuracy despite a large increase in computational cost. A basic staggered weak coupling scheme is shown for comparison. 46
4.9 Flexible cantilever behind a rigid square body. All distances shown are in cm. . 46
4.10 The the cantilever near maximal displacement (Entropy). 47
4.11 The vertical displacement of the cantilever tip as a function of time. 47

5.1 (a) The undeformed fluid (green) and structure (blue) meshes for the 2D Membrane at a 10° angle of attack. (b) The region near the structure is enlarged. . . 50
5.2 The rigid plate and two membranes at time $T = 1.0$ (top), 2.0, 3.0, and 4.0 (bottom). (Entropy). 51
5.3 Lift and drag coefficients as a function of time for a rigid plate and two flexible membranes at 10° angle of attack. 51

5.4 A cross section of the fluid mesh (blue) and entire structure mesh (green) in the reference configuration (left) and typical deformed configuration (right). 52
5.5 A three-dimensional membrane at various times (Mach number on iso-entropy surfaces). The leading edge of the membrane aligns with the fluid and prevents separation. 53
5.6 A rigid plate at various times (Mach number on iso-entropy surfaces). Note the leading edge separation caused by the high angle of attack. 54
5.7 The two-dimensional wing model in non-dimensionalized coordinates. The left and right endpoints of the structure follow a prescribed motion. 55
5.8 The two-dimensional wing (black) at time $T = 17.0$ (top), 18.0, 19.0, and 20.0 (bottom). The structures have varying Young's modulus E and prestretching factor Δ, and range from nearly rigid (left) to quite flexible (right). (Entropy). . 56
5.9 The overhead view of the three-dimensional wing which is uniformly extruded in the z direction. Dirichlet conditions are imposed along the planes given by the solid lines. The leading edge is driven with a sinusoidal flapping motion in the y-z plane, and the root chord is held fixed. 57
5.10 A three-dimensional wing at various times (Mach number on iso-entropy surfaces). 58

6.1 The dimensions of a cross section of the tuning fork, as well as its material parameters. All distances are given in cm. The tuning fork is extruded 0.5 cm so that the cross-section of the tines are square. 60
6.2 The computational mesh for the tuning fork (green) and two different cross sections of the computational mesh for the fluid (blue) in a region near the tuning fork. 61
6.3 Time series data for the relative pressure at three locations each a distance 5.0 cm from the axis of the tuning fork in a plane perpendicular to the axis. The plane is located such that the tines of the tuning fork extend a distance 0.5 cm through the plane. The outer box shows the boundary of computational domain. 63
6.4 The relative pressure in a plane perpendicular to the axis of the tuning fork (as in fig. 6.3a) at various times. For scale, the figures are 20 cm per side which represents almost all of the computational domain in that plane. 64
6.5 The sound pressure level L_p relative to a reference pressure 2×10^{-5} Pa for a range of frequencies, as measured over the last 9 periods of the base frequency as observed at location A. The frequencies of several eigenmodes of the linearized structure are shown for comparison. 65
6.6 The relative sound pressure levels by angle at various distances from the axis of the tuning fork measured in 1° increments, averaged over nine periods of the fundamental mode. Each plot has been normalized to its maximum value. The theoretical curve for a linear quadrupole as given in eq. (6.5) is shown in a solid line. The tines of the tuning fork lie at 0° and 180°. Notice the 5 dB difference in sound pressure level between the two maxima in the extreme near-field. . . . 66
6.7 Kinetic, potential, and total energy of the tuning fork. 67

List of Tables

2.1 Typical parameter values used when modeling air. 7

3.1 The 3rd order implicit-explicit Runge-Kutta coefficients ARK3(2)4L[2]SA. . . . 24
3.2 The number of high-order nodes per element for simplicial (triangles and tetrahedra) and quad/hex meshes of various polynomial orders. 24

4.1 A comparison of the oscillation frequency and maximal vertical tip displacement of the cantilever with values reported in the literature, as reproduced from ref. [34]. The coupling abbreviations stand for partitioned block Gauss-Seidel (P-BGS), partitioned block-Newton (P-BN), and partitioned Newton-Raphson (P-NR). 48

6.1 Significant frequencies and Q factors observed in the time series pressure data at location A (see fig. 6.3a) during $5.0\,\text{ms} \leq t \leq 30.0\,\text{ms}$, as extracted by the filter diagonalization method. 68

List of Algorithms

3.1 Implicit-explicit Runge-Kutta method. 22
4.1 Adaptive Newton-Raphson method. 36
4.2 Time integration scheme for the coupled fluid-structure system. 40

Acknowledgments

I would like to thank my advisors, Professors Per-Olof Persson and Jon Wilkening, for their encouragement and support over these years. As our relationship matured they have treated me more and more like a colleague, allowing me to pursue areas I found interesting and gently reminding me of the importance to share my work through journal papers, conference proceedings, and talks.

I am also grateful to my outside committee member, Prof. Larry Karp, with whom I had several insightful conversations about my research and the numerical methods used in Agriculture and Resource Economics. I must acknowledge two other professors, L. Craig Evans and John Strain, who kindly sat on my qualifying examination and made the process as enjoyable as it could be.

Several colleagues merit special recognition including Trevor Potter, Adam Boocher, Luming Wang, Matthew Zahr, Robert Saye, and Chris Rycroft for their insightful discussions, technical guidance, and overall support.

From my time as an undergraduate at the University of Minnesota I must highlight the advising received from several Mathematics faculty including Peter Webb and Steven Sperber. In addition I am very grateful to E. Dan Dahlberg of the Physics department for letting me work in his research laboratory. The Institute of Technology Honors Group advisors, Pamela Drake and Robert Pepin, left a lasting impression over the four years I worked in their office as an undergraduate tutor. In addition I am eternally grateful for their advice and guidance in applying for scholarships, fellowships, and graduate schools.

I would be remiss not to mention the collaborations with several faculty and students in the course of two Research Experience for Undergraduates programs. The first, at the University of Notre Dame with Prof. Michael Gekhtman and Adam Boocher, led to my first journal publication [15]. The second, at the Claremont Colleges and with Prof. Michael Orrison of Harvey Mudd College and Marie Jameson, gave me valuable experience in numerical linear algebra which would prove useful later.

Before college, my interest in mathematics was nurtured at the University of Minnesota Talented Youth Mathematics Program. Prof. Harvey Keynes deserves special recognition for both his role in founding the program and as an excellent multivariable calculus instructor. In addition, Michael Lawler was instrumental in laying down a strong introduction to calculus and encouraging me to enter the American Mathematics Competitions. Thanks to John Winterhalter of Eden Prairie High School for organizing the AMC exams and coaching the high school math team. Thanks to Tom Kilkelly, Bill Boulger, and Mike Reiners for organizing the Minnesota American Regions Math League team, and to Andrew Niedermaier who served as a role model and often drove me to the practices.

I am exceedingly grateful to my family. To my wife, Corinne Scown, PhD, for her love and support in finishing my research and writing my dissertation. To my parents, Michael and Nola, for their seemingly endless emotional and financial support. Words alone cannot express my gratitude. And to my wonderful younger sister, Kate, whom I have had the pleasure of watching grow and mature into a lovely, confident young woman.

Several open source tools including Python, IPython, and Cython provided the necessary infrastructure for my research. In particular, I am very grateful to Fernando Pérez, the founder of the IPython project, who accepted my contributions and invited me to join the official IPython core development team. In addition, the freely available BLAS, Boost, DistMesh, HDF5, h5py, LAPACK, matplotlib, METIS, mpi4py, MUMPS, NumPy, ScaLAPACK, SciPy, SuiteSparse, and SymPy libraries were invaluable in allowing me to rapidly implement new ideas.

Lastly I would like to acknowledge generous support from the AFOSR Computational Mathematics program under grant FA9550-10-1-0229; the Alfred P. Sloan foundation; the Lawrence Berkeley National Laboratory and the National Energy Research Scientific Computing Center funded by the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231; and the Simons Fellowship.

Chapter 1

Introduction

Many important scientific and engineering problems require predictions of fluid-structure interaction (FSI). For example, unexpected interactions in engineered systems can lead to catastrophic failure or generally undesirable behavior. As a famous example, consider the Tacoma Narrows Bridge which earned the nickname "Galloping Gertie" for the large pitching deformation of the roadway deck observed during windy conditions. Less than 4 months after opening, on November 7, 1940, an aerodynamically induced excitation in a torsional mode of the structure caused the bridge to catastrophically fail [11].

Aeroelastic flutter, a positively reinforced coupling between the aerodynamic forces on an object and the natural mode of the object, can produce large and potentially destructive vibrations in aircraft, turbines, chimneys, and other structures. If during each oscillation the energy transferred to the structure is greater than the energy naturally dissipated, the amplitude of oscillation will increase. If this amplitude continues to grow the mechanical structure may not be able to withstand the internal stresses and may fail. Even if the structure does not fail, flutter may cause undesirable vibrations or noise.

Simulating fluid-structure interactions may also be useful in understanding flapping flight or producing biologically inspired micro aerial vehicles. The wing of a bat, for example, can be modeled as a collection of mostly rigid bones connected by a membrane [8]. This description naturally suggests a fluid-structure interaction model where the structure is driven in a flapping motion and the various aerodynamic properties like lift and drag may be computed from the simulation.

In medicine, a prototypical fluid-structure interaction problem would be the flow of blood in the cardiovascular system [19]. During each contraction of the heart a pressure wave is generated which causes local deformations in the walls of the arteries. A naïve model using rigid walls may fail to predict the proper pressure wave propagation without corrective terms. In theory, one should be able to accurately recover blood flow dynamics by coupling a proper fluid model of blood and elastic structure model of arterial walls, without resorting to ad hoc correction terms. A better understanding of the flow dynamics is thought to help predict and diagnose various cardiovascular pathologies [29].

Numerical acoustical simulations often require a special acoustic boundary condition to

properly model walls. Here one generally specifies the acoustical impedance — a complex number, potentially depending on the frequency of the incident wave — which describes the absorption and reflection of a wave at the surface. This boundary condition is most naturally represented in the frequency-domain, but can be applied in time-domain calculations [16, 31], essentially by modeling variations in the local wall displacement and velocity depending on the incident pressure. This interpretation naturally shows that the proper acoustical behavior could also realized using a coupled fluid-structure system with a proper elastic structure model.

In these systems the interaction between the fluid and structure often involve multiple scales. For example, the sound speeds and characteristic feature sizes in the fluid and structure may be vastly different. In addition, the interaction may involve non-linear effects. These issues often make it challenging to solve even relatively simple problems accurately.

1.1 Previous Work

There have been many approaches suggested for the simulation of fluid-structure interaction problems [32, 56, 70], but most schemes use one of two approaches to solve the coupled system. In the fully coupled (monolithic) approach, the two equations are solved simultaneously. This is straightforward to implement using an explicit time integration scheme [54], but depending on the problem an implicit scheme may be desirable due for any number of reasons, for example, an overly restrictive Courant-Friedrichs-Lewy (CFL) condition. Unfortunately the situation becomes much more complicated when using an implicit scheme. Here one generally needs to use specialized codes to produce a full Jacobian matrix containing off-diagonal blocks corresponding to the fluid-to-structure and structure-to-fluid couplings. Developing a good preconditioner for such matrices is generally difficult, especially since the diagonal blocks, representing the fluid system and structure system individually, often have wildly disparate properties.

An alternative which many resort to using is the so-called *partitioned* method in which an existing fluid solver and an existing structure solver are used in serial, transferring data along the fluid-structure interface from one solver to the other to maintain the coupling. It is still possible to solve the fully-coupled system using a partitioned method. Here one generally uses fixed point iteration, or some variant, until convergence is reached. Unfortunately since this only converges linearly in the number of iterations it is often prohibitively expensive, even for modest problems. In addition, convergence itself may be difficult to achieve.

Instead, the most common solution technique could be described as *weak-coupling* in which for a given timestep the fluid and structure are each solved individually and coupling transferred between them at the timestep. This method is easy to implement but yields only first order accuracy in time. It may be possible to boost this to second order in time using a Strang splitting technique, but higher-order accuracy is generally impossible.

1.2 Overview

In this work we propose using a high-order integration scheme based upon the coefficients of an Implicit-Explicit (IMEX) Runge-Kutta method [40]. The main idea is to integrate the contribution from an off-diagonal block explicitly and the remaining terms implicitly [73]. In practice, we choose the contribution arising from the fluid-to-structure coupling as the term to integrate explicitly. This effectively decouples the system into one where we can call the fluid and structure solvers individually. The actual algorithm is written as a predictor-corrector scheme in which the stage predicted fluid-to-structure coupling is a linear combination of previous stage couplings according to a formula which uses both the implicit and explicit Runge-Kutta coefficients [75]. It is important to note that this naturally allows for stage subiterations. It has been our experience that there is some benefit of increased stability when using additional subiterations, but we emphasize that subiterations are not required to achieve the design accuracy of the method.

There are many other partitioned FSI schemes, many of which employ similar predictor-corrector frameworks to achieve first [25, 55] or second [23, 32] order accuracy. See [28] for a review of ideas used in partitioned schemes.

For the fluid model we use a nodal high-order discontinuous Galerkin (DG) formulation for the compressible Navier-Stokes equations. We handle the deforming domain using an arbitrary Lagrangian-Eulerian (ALE) formulation, as is common in fluid-structure interaction problems [1, 18, 24, 43, 65]. The ALE formulation transforms the system of conservation laws on the deforming real domain into a system of conservation laws on a fixed reference domain, essentially through a change of variables procedure.

The non-linear fluid equations are solved using a Newton-GMRES approach with a block-ILU(0) or block-Jacobi preconditioner [53]. The fluid solver is implemented in parallel using MPI for communication and efficiently scales to several thousand processes [51].

We generally couple the fluid to either a rigid body or a non-linear hyperelastic structure. The rigid body is typically modeled using standard rigid body dynamics (i.e., $F = ma$ and $\tau = I\alpha$) although in some cases certain variables may be constrained. The elastic structure is discretized using a standard nodal high-order continuous Galerkin (CG) finite element method, written in a first order formulation.

The coupling between the fluid and structure is straightforward. As the structure moves it deforms the domain the fluid occupies. This is implemented by moving the boundary nodes of the fluid to conform to the new structure position, and then interpolating the boundary deformation into the interior of the fluid mesh. We use two different methods of interpolation. The first approach uses radial basis functions [9, 13], which works well for small to moderate deformations. The second approach treats the fluid domain as an elastic structure and solves an elastic deformation problem with prescribed boundary positions. This approach is more computationally expensive but may be able to produce higher quality deformed meshes.

The fluid influences the structure through a surface traction, i.e., force per unit area, applied along the fluid-structure boundary. In the case of a rigid structure this traction is

integrated over the entire structure surface to find the net force and torque. Otherwise this traction is computed everywhere along the surface and supplied to the structure solver as part of its boundary conditions. To ensure a high order coupling this transfer of data is done at Gauss integration nodes, not solution nodes.

To simplify these couplings we generally assume that the fluid and structure meshes conform along their common interface, meaning that the meshes share common faces along the interface. In addition we require that the same order polynomials be used to integrate both the fluid and structure, meaning that the solution nodes and Gauss nodes are also shared along the interface.

This work is organized as follows. First we present the equations for both the fluid on the deforming domain and the non-linear structure. Next we present standard discontinuous Galerkin and continuous Galerkin spatial discretizations of the fluid and structure, and discuss standard Runge-Kutta time integration schemes. We then describe our parallel high-order numerical solvers, before moving on to develop the coupling between the fluid and structure systems and construct a high-order time integration technique for the coupled system. We verify the high-order accuracy of the scheme using a test problem of a heaving and pitching NACA airfoil in a laminar flow, subject to a simple smooth heaving motion. In addition we show a standard FSI test problem consisting of a flexible cantilever behind a square bluff body, showing good agreement with tip displacement and oscillation frequency to values found in the open literature.

We then move on to show several uses of this FSI method, broadly separated into membrane and flapping wings applications and acoustics applications. In both two and three dimensions we demonstrate the ability of a compliant thin membrane to align with the incident flow and delay or prevent leading edge separation. We then extend the work to basic models of flapping flight, confirming that the same general principle may hold in certain types of biological flight like that of bats.

Lastly we show the ability of the method to generate acoustic waves in a three-dimensional model of a tuning fork. Here we accurately capture the sound generation and decay of the fundamental and clang modes.

Chapter 2

Governing Equations

2.1 Compressible Navier-Stokes

The compressible Navier-Stokes equations are a non-linear system of equations which can be written in conservation form as:

$$\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x_j}(\rho u_j) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij}) = +\frac{\partial}{\partial x_j}\tau_{ij} \qquad \text{for } i = 1, 2, 3 \tag{2.2}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho u_j E + u_j p) = \frac{\partial}{\partial x_j}(-q_j + u_i \tau_{ij}) \tag{2.3}$$

where the conserved variables are the fluid density ρ, momentum in the j-th spatial coordinate direction ρu_j, and total energy ρE. The viscous stress tensor and heat flux are given by

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) \quad \text{and} \quad q_j = -\frac{\mu}{\Pr}\frac{\partial}{\partial x_j}\left(E + \frac{p}{\rho} - \frac{1}{2}u_k u_k\right). \tag{2.4}$$

Here, μ is the dynamic viscosity and $\Pr = 0.72$ is the Prandtl number which we assume to be constant. For an ideal gas, the pressure p has the form

$$p = \rho R T \tag{2.5}$$

where R is the ideal gas constant and T is the temperature. The temperature is related to the internal energy of the fluid $e = E - u_k u_k/2$ according to the relations

$$\gamma = \frac{C_P}{C_V}, \quad e = C_V T, \quad \text{and} \quad C_P - C_V = R, \tag{2.6}$$

where C_P and C_V are the heat capacities at constant pressure and volume and their ratio γ is the adiabatic gas constant.

From these equations one can easily work out a simple closed form for the pressure

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}u_k u_k\right). \tag{2.7}$$

For air we set the adiabatic gas constant γ to 1.4 which is the theoretical value for a diatomic ideal gas, and which is very nearly equal to empirical values measured over a wide range of conditions.

In later sections we will often write the Navier-Stokes equations in vector notation as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F}_{\text{inv}}(\boldsymbol{u}) - \boldsymbol{\nabla} \cdot \boldsymbol{F}_{\text{vis}}(\boldsymbol{u}, \boldsymbol{\nabla u}) = \boldsymbol{0} \tag{2.8}$$

where the conserved variables, and inviscid and viscous fluxes in the j-th spatial coordinate are

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \quad \boldsymbol{F}_{\text{inv},j}(\boldsymbol{u}) = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + \delta_{ij} p \\ u_j(\rho E + p) \end{bmatrix}, \quad \text{and} \quad \boldsymbol{F}_{\text{vis},j}(\boldsymbol{u}, \boldsymbol{\nabla u}) = \begin{bmatrix} 0 \\ \tau_{ij} \\ -q_j - u_i \tau_{ij} \end{bmatrix}. \tag{2.9}$$

The enthalpy H and speed of sound a are given by

$$H = E + \frac{p}{\rho} \quad \text{and} \quad a = \sqrt{\frac{\gamma p}{\rho}}. \tag{2.10}$$

For visualization we often use the vorticity ω, entropy s, or Mach number M which are calculated as

$$\omega = \nabla \times u_i, \quad s = \frac{p}{\rho^\gamma}, \quad \text{and} \quad M = \frac{\sqrt{u_k u_k}}{a} = \sqrt{\frac{\rho u_k u_k}{\gamma p}}. \tag{2.11}$$

Vorticity measures the local rotation in the flow and is particularly useful in two dimenions when the quantity can be treated as a scalar (i.e., only the z-component is non-zero). Entropy is useful for visualization because it is a quantity which is produced along walls and advected with the flow. Plotting iso-entropy contours in both two and three dimensions is useful for highlighting the characteristics of the flow. Lastly, the Mach number is the ratio of the speed of the fluid to the local speed of sound.

When modeling air at normal conditions we typically use the constants in table 2.1.

2.1.1 Boundary Conditions

We impose several different boundary conditions in our simulations. In the following descriptions let $\vec{v} = [u_1, u_2, u_3]^T$ be the fluid velocity, $\vec{v}_{\text{wall}}$ be the wall velocity, and $\hat{n}$ be an outward pointing unit vector.

Table 2.1: Typical parameter values used when modeling air.

Name	Symbol	Value
Density	ρ	$1.204\,\mathrm{kg/m^3}$
Dynamic Viscosity	μ	$1.83 \times 10^{-5}\,\mathrm{kg/(m{\cdot}s)}$
Speed of Sound	a	$343.0\,\mathrm{m/s}$
Ratio of Specific Heats	γ	1.4
Prandtl number	Pr	0.72

Adiabatic No-Slip

On fluid boundaries with physical objects one generally imposes an adiabatic no-slip condition. This means no relative motion between the wall and the fluid at the interface, i.e.,

$$\vec{v} = \vec{v}_{\text{wall}}. \tag{2.12}$$

In addition, no heat T (i.e., e) or mass ρ is passed into or out of the wall.

Adiabatic Slip

On a problems with a symmetry plane one often chooses to model only half of the domain and impose an adiabatic slip condition on the boundary corresponding to the symmetry plane. This is a relaxed form of the no-slip condition where we allow fluid to move along the boundary but not pass through the boundary, i.e.,

$$\vec{v} \cdot \hat{n} = \vec{v}_{\text{wall}} \cdot \hat{n}. \tag{2.13}$$

Again, no heat or mass is transmitted through the boundary.

Far-Field

Since the computational domain must be finite, it generally necessary to impose far-field boundary conditions on the far boundaries of simulation. Here we generally model uniform "free-stream" flow by imposing the full state of the system. Often we choose free-stream density ρ_∞, velocity $\vec{v}_\infty$ and pressure p_∞. From these values we calculate the full system state as

$$\boldsymbol{u}_\infty = \left[\rho_\infty,\ \rho_\infty \vec{v}_\infty,\ \frac{p_\infty}{\gamma - 1} + \frac{1}{2}\rho_\infty \|\vec{v}_\infty\|^2\right]^T. \tag{2.14}$$

The details of how the various boundary conditions are imposed in the actual numerical simulation will be left until section 3.1.

2.1.2 Isentropic & Isothermal Formulations

As an alternative to solving the full set of compressible Navier-Stokes equations, several approximations may be used to reduce the dimensionality of the system. These reductions are most useful in the case of nearly incompressible flow.

Isentropic

An isentropic process is one in which the entropy s is constant. Since for an ideal gas $s = p/\rho^\gamma$, we can solve for pressure as

$$p = s\rho^\gamma. \tag{2.15}$$

Notice that this approximation allows the elimination of the energy equation (eq. (2.3)) from the compressible Navier-Stokes equations. This effectively reduces the dimension of the system from $D + 2$ variables to $D + 1$ variables in D dimensions. In three spatial dimensions this results in 20% reduction in the number of degrees of freedom in the overall discretized system. In addition, since the number of nonzero elements in the Jacobian matrix in an implicit temporal discretization scales like the square of the number of variables, this accounts for a 36% reduction in the total number of nonzero entries in the Jacobian.

While it is generally notoriously difficult to prove many facts about the Navier-Stokes equations [27], it is known that the solution to the isentropic model converges to the solution of the incompresible Navier-Stokes equations as the Mach number goes to 0, as long as one makes suitable assumptions on the domain, initial data, existence of solutions, and definition of convergence [21, 42].

Isothermal

An isothermal process is one in which the termparature T remains constant. The ideal gas law then shows that locally the pressure must be proportional to the density:

$$p = \rho R T_\infty. \tag{2.16}$$

where R is the specific gas constant and T_∞ is the (constant) temperature. For dry air we use the value

$$R_{\text{dry air}} = 287.058\,\text{J}/(\text{kg}\cdot\text{K}). \tag{2.17}$$

This approximation again allows us to eliminate the energy equation (eq. (2.3)) from the compressible Navier-Stokes equations.

2.2 Arbitrary Lagrangian Eulerian formulation

The deformable fluid domain is handled through an Arbitrary Lagrangian Eulerian (ALE) formulation. Through a change of variables technique, the ALE formulation converts a

system of conservation laws in a time varying physical domain into a system of conservation laws in a fixed reference domain. In writing the equations for the ALE formulation it is typical to denote the reference domain in capital letters ($\boldsymbol{X}$, $\boldsymbol{U}$, $\boldsymbol{F}$, V, etc) and the physical domain in lowercase letters ($\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{f}$, $v(t)$, etc).

Consider, for the moment, a time varying mapping from the reference domain V to the physical domain $v(t)$ in which a point $\boldsymbol{X} \in V$ is mapped to $\boldsymbol{x}(\boldsymbol{X}, t) \in v(t)$. We define the deformation gradient $\boldsymbol{G}$ as the spatial gradient of the mapping, i.e.,

$$\boldsymbol{G} = \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{x}. \tag{2.18}$$

For each reference point $\boldsymbol{X}$ the deformation gradient describes how space is locally deformed in the physical configuration. For example, the determinant

$$g = \det \boldsymbol{G} \tag{2.19}$$

describes local volume deformation, with $g = 1$ corresponding to no volume deformation. Other matrix invariants of $\boldsymbol{G}$ give additional information about the local deformation, a point which we will return to when discussing the non-linear elasticity model. Lastly, we define the mapping velocity

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial t}. \tag{2.20}$$

Following the derivation in [52], we note that a normal vector $\boldsymbol{n}$, elemental area da, and elemental volume dv in physical coordinates are related to a normal vector $\boldsymbol{N}$, elemental area dA, and elemental volume dV in reference coordinates according to

$$\boldsymbol{n}\, da = g\boldsymbol{G}^{-T}\boldsymbol{N}\, dA, \quad \boldsymbol{N}\, dA = g^{-1}\boldsymbol{G}^{T}\boldsymbol{n}\, da, \quad \text{and} \quad dv = g\, dV. \tag{2.21}$$

Next we consider a generic system of conservation laws in the physical domain $(\boldsymbol{x}, t)$,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{\nabla u}) = 0 \tag{2.22}$$

where $\boldsymbol{u}$ is a vector of conserved physical quantities and $\boldsymbol{f}$ is a vector of physical fluxes. We then write the system in integral form as

$$\int_{v(t)} \frac{\partial \boldsymbol{u}}{\partial t}\, dv + \int_{\partial v(t)} \boldsymbol{f} \cdot \boldsymbol{n}\, da = \boldsymbol{0} \tag{2.23}$$

which is obtained by integrating eq. (2.22) over $v(t)$ and applying the divergence theorem to the second term. We may then apply Reynolds transport theorem to the first term, change to the reference coordinates, and reapply Reynolds transport theorem to get

$$\int_V \frac{\partial (g^{-1}\boldsymbol{u})}{\partial t}\, dV - \int_{\partial V} (g\boldsymbol{u}\boldsymbol{G}^{-1}\dot{\boldsymbol{x}}) \cdot \boldsymbol{N}\, dA + \int_{\partial V} (g\boldsymbol{G}^{-}1\boldsymbol{f}) \cdot \boldsymbol{N}\, dA = 0. \tag{2.24}$$

Re-applying the divergence theorem allows us to see this as a conservation law in the reference domain

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{\nabla}_{\boldsymbol{X}} \cdot \boldsymbol{F}(\boldsymbol{U}, \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{U}) = 0 \tag{2.25}$$

in which the conserved quantities and fluxes are

$$\boldsymbol{U} = g\boldsymbol{U} \quad \text{and} \quad \boldsymbol{F} = g\boldsymbol{G}^{-1}\boldsymbol{f} - \boldsymbol{U}\boldsymbol{G}^{-1}\dot{\boldsymbol{x}}. \tag{2.26}$$

We often assume that $\boldsymbol{f}$ contains both inviscid and viscous terms, so that $\boldsymbol{f} = \boldsymbol{f}_{inv}(\boldsymbol{u}) + \boldsymbol{f}_{vis}(\boldsymbol{u}, \boldsymbol{\nabla u})$. In the ALE formulation we ascribe the flux contribution from the moving domain to the inviscid term. That is,

$$\boldsymbol{F}_{inv} = g\boldsymbol{G}^{-1}\boldsymbol{f}_{inv} - \boldsymbol{U}\boldsymbol{G}^{-1}\dot{\boldsymbol{x}} \tag{2.27}$$

$$\boldsymbol{F}_{vis} = g\boldsymbol{G}^{-1}\boldsymbol{f}_{vis} \tag{2.28}$$

Lastly, we note that the gradient in the physical domain is easily computed using the chain rule as

$$\nabla \boldsymbol{u} = (\nabla_{\boldsymbol{X}}(g^{-1}\boldsymbol{U}))\boldsymbol{G}^{-T} = (g^{-1}\nabla_{\boldsymbol{X}}\boldsymbol{U} - \boldsymbol{U}\nabla_{\boldsymbol{X}}(g^{-1}))\boldsymbol{G}^{-T}. \tag{2.29}$$

Before moving one we comment on the so-called geometric conservation law (GCL) [64]. When using the ALE formulation as just described, it is generally not the case that a constant (i.e., free-stream) solution in the physical domain is a solution of the discretized equations in the reference domain. Enforcing the GCL may be done by integrating an auxiliary equation

$$\frac{\partial \bar{g}}{\partial t} - \boldsymbol{\nabla}_{\boldsymbol{X}}(g\boldsymbol{G}^{-1}\dot{\boldsymbol{x}}) = 0 \tag{2.30}$$

and replacing the system of conservation laws eq. (2.25) by

$$\frac{\partial \bar{g} g^{-1}\boldsymbol{U}}{\partial t} + \boldsymbol{\nabla}_{\boldsymbol{X}} \cdot \boldsymbol{F} = \boldsymbol{0}. \tag{2.31}$$

For low order methods the error introduced by violating the GCL can be quite severe, but numerical experiments in [52] show that the issue is less acute for high-order methods. Since we generally restrict ourselves to using high-order methods, for simplicity we are not enforcing the GCL in our results.

For a good description of the DGCL, see [24, 32].

2.3 Rigid Body Dynamics

We quickly remind the reader that for a rigid body the center of mass $\boldsymbol{x}_{\text{cm}}$ obeys Newton's second law of motion,

$$\boldsymbol{F} = m\boldsymbol{a} \tag{2.32}$$

where $\boldsymbol{F}$ is the net force acting on the body, m is the mass of the body, and $\boldsymbol{a} = d^2\boldsymbol{x}_{\text{cm}}/dt^2$ is the resulting acceleration. If the force is transmitted to the object as a surface traction $\boldsymbol{t}$, i.e, force per unit area along the body surface, the net force is easily obtained by integration

$$\boldsymbol{F} = \int_{\partial V} \boldsymbol{t}\, dA. \tag{2.33}$$

A similar law holds for the rotational dynamics about a fixed axis, namely

$$\boldsymbol{\tau} = I\boldsymbol{\alpha} \tag{2.34}$$

where $\boldsymbol{\tau}$ is the net torque acting on the center of mass, I is the moment of inertia about the center of mass (along the fixed axis), and $\boldsymbol{\alpha} = d^2\boldsymbol{\theta}/dt^2$ is the angular acceleration. Again, if a surface traction is applied to the body the net torque is easily calculated as

$$\boldsymbol{\tau} = \int_{\partial V} (\boldsymbol{x} - \boldsymbol{x}_{\text{cm}}) \times \boldsymbol{t}\, dA. \tag{2.35}$$

More complicated rigid body kinematics are easily derived using the Euler-Langrange equations and the Lagrangian $L = T - V$, the difference between the kinetic energy T and potential energy V of the system. For more details see [10].

2.4 Neo-Hookean Elasticity Model

We use a hyperelastic neo-Hookean formulation [36] for modeling deformable structures. Here, the structure position is given by a mapping $\boldsymbol{x}(\boldsymbol{X}, t)$, which for each time t maps a point $\boldsymbol{X}$ in the unstretched reference configuration to its location $\boldsymbol{x}$ in the deformed configuration. From this we compute the mapping velocity and deformation gradient as

$$\boldsymbol{v} = \frac{\partial \boldsymbol{x}}{\partial t}, \quad \text{and} \quad \boldsymbol{F} = \nabla_{\boldsymbol{X}} \boldsymbol{x}(\boldsymbol{X}, t). \tag{2.36}$$

We partition boundary of the structure domain into regions of Dirichlet and Neumann boundary conditions, $\partial V = \Gamma_D \cup \Gamma_N$. On the Dirichlet boundary Γ_D we prescribe the material position $\boldsymbol{x}_D$, often corresponding to no displacement. On the Neumann boundary Γ_N we allow for a general surface traction, i.e., force per unit surface area, which we denote $\boldsymbol{t}$.

The governing equations for the structure are then given by

$$\frac{\partial \boldsymbol{p}}{\partial t} - \nabla \cdot \boldsymbol{P}(\boldsymbol{F}) = \boldsymbol{b} \qquad \text{in } V \tag{2.37}$$

$$\boldsymbol{P}(\boldsymbol{F}) \cdot \boldsymbol{N} = \boldsymbol{t} \qquad \text{on } \Gamma_N \tag{2.38}$$

$$\boldsymbol{x} = \boldsymbol{x}_D \qquad \text{on } \Gamma_D \tag{2.39}$$

where $\boldsymbol{p} = \rho\boldsymbol{v} = \rho\,\partial\boldsymbol{x}/\partial t$ is the momentum, $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{b}$ is an external body force per unit reference volume, and $\boldsymbol{N}$ is a unit normal vector in the reference domain.

For a compressible neo-Hookean material the strain energy density is given by

$$W = \frac{\mu}{2}(\bar{I}_1 - 3) + \frac{\kappa}{2}(J-1)^2 \tag{2.40}$$

where $\bar{I}_1$, the first invariant of the deviatoric part of the left Cauchy-Green deformation tensor, and J, the determinant of the deformation gradient, are calculated as

$$\bar{I}_1 = J^{-2/3} I_1, \quad I_1 = \operatorname{tr}\boldsymbol{B} = \operatorname{tr}(\boldsymbol{F}\boldsymbol{F}^T), \quad J = \det\boldsymbol{F}. \tag{2.41}$$

The constants μ and κ are the shear and bulk modulus of the material. We often find it more convenient to think in terms of Young's modulus E and Poisson's ratio ν which are related to the shear and bulk modulus as

$$\mu = \frac{E}{2(1+\nu)} \quad \text{and} \quad \kappa = \frac{E}{3(1-2\nu)}. \tag{2.42}$$

The first Piola-Kirchhoff stress tensor is computed as

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial W}{\partial \boldsymbol{F}} = \mu J^{-2/3}\left(\boldsymbol{F} - \frac{1}{3}\operatorname{tr}(\boldsymbol{F}\boldsymbol{F}^T)\boldsymbol{F}^{-T}\right) + \kappa(J-1)J\boldsymbol{F}^{-T}. \tag{2.43}$$

For two-dimensional problems we use a plane strain formulation in which we treat the stretch in the third dimension as constant [14], thinking of the problem as modeling the cross-section of an infinitely long prismatic structure. Here we use a slightly modified strain energy density

$$W = \frac{\mu}{2}(\bar{I}_1 - 2) + \frac{\kappa}{2}(J-1)^2 \tag{2.44}$$

where we have replaced the 3 in eq. (2.40) by 2. Another way to see this is to note that in three dimensions $I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$ where λ_i are the three eigenvalues of $\boldsymbol{F}$. In the plane strain formulation we posit $\lambda_3 = 1$ and thus $I_1 = \lambda_1^2 + \lambda_2^2 + 1$. For posterity, we present the Piola-Kirchhoff stress tensor for the two-dimensional plane strain formulation:

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial W}{\partial \boldsymbol{F}} = \mu J^{-2/3}\left(\boldsymbol{F} - \frac{1}{3}(\operatorname{tr}(\boldsymbol{F}\boldsymbol{F}^T) + 1)\boldsymbol{F}^{-T}\right) + \kappa(J-1)J\boldsymbol{F}^{-T}. \tag{2.45}$$

2.4.1 Quasi-Static Formulation

In some instances, particularly with regard to mesh deformation methods as will be discussed in section 4.1.2, we are interested in quasi-static solutions of the non-linear elasticity equations, i.e., solutions where the material velocity is infinitely small. This is easily imposed by

setting $\partial \boldsymbol{p}/\partial t = 0$. The structure eqs. (2.37) to (2.39) then become

$$-\nabla \cdot \boldsymbol{P}(\boldsymbol{F}) = \boldsymbol{b} \qquad \text{in } V \tag{2.46}$$

$$\boldsymbol{P}(\boldsymbol{F}) \cdot \boldsymbol{N} = \boldsymbol{t} \qquad \text{on } \Gamma_N \tag{2.47}$$

$$\boldsymbol{x} = \boldsymbol{x}_D \qquad \text{on } \Gamma_D. \tag{2.48}$$

Here again $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor as given in eq. (2.45) and eq. (2.43) for two and three dimensions.

Chapter 3

Discretization

3.1 Fluid Spatial (Discontinuous Galerkin)

The structure equations as described in section 2.2 are discretized using a high-order discontinuous Galerkin formulation with tetrahedral mesh elements and nodal basis functions. The inviscid fluxes are computed using Roe's method [57], and the numerical fluxes for the viscous terms are chosen according to the compact DG method [48]. Below, we summarize this discretization for the ALE system of conservation laws eq. (2.25). For simplicity, we change the notation and use lower-case symbols for the solution $\boldsymbol{u}$, and we omit the subscripts on the derivative operators. We also split the fluxes into an inviscid component $\boldsymbol{F}^i(\boldsymbol{u})$ and a viscous component $\boldsymbol{F}^v(\boldsymbol{u}, \nabla\boldsymbol{u})$, corresponding to terms in the left-hand side and the right-hand side of eqs. (2.1) to (2.3), respectively.

Following standard procedure for DG discretization of second-derivatives [6], we first introduce the auxiliary gradient variables $\boldsymbol{q} = \nabla\boldsymbol{u}$, and write eq. (2.25) as the system of first order equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}^i(\boldsymbol{u}) - \nabla \cdot \boldsymbol{F}^v(\boldsymbol{u}, \boldsymbol{q}) = 0, \tag{3.1}$$

$$\nabla\boldsymbol{u} = \boldsymbol{q}. \tag{3.2}$$

We introduce a computational mesh and denote its elements by $\mathcal{T}_h = \{K\}$. Furthermore, we introduce the finite element spaces $\mathcal{V}_h^p$ and Σ_h^p as:

$$V_h^p = \{\boldsymbol{v} \in [L^2(\Omega)]^5 \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^5 \ \forall K \in \mathcal{T}_h\}, \tag{3.3}$$

$$\Sigma_h^p = \{\boldsymbol{\tau} \in [L^2(\Omega)]^{5\times 3} \mid \boldsymbol{\tau}|_K \in [\mathcal{P}_p(K)]^{5\times 3} \ \forall K \in \mathcal{T}_h\}, \tag{3.4}$$

where $\mathcal{P}_p(K)$ is the space of polynomial functions of degree at most $p \geq 1$ on K. To obtain a form that is appropriate for discretization using the CDG method, we multiply the system of equations eqs. (3.1) and (3.2) by test functions $\boldsymbol{v}, \boldsymbol{\tau}$ and integrate by parts. Our semi-discrete DG formulation is then expressed as: find $\boldsymbol{u}_h \in V_h^p$ and $\boldsymbol{q}_h \in \Sigma_h^p$ such that for all

$K \in \mathcal{T}_h$, we have

$$\int_K \frac{\partial \boldsymbol{u}_h}{\partial t} \cdot \boldsymbol{v}\, dx + \int_K \left(\boldsymbol{F}^i(\boldsymbol{u}_h) - \boldsymbol{F}^v(\boldsymbol{u}_h, \boldsymbol{q}_h)\right) : \nabla \boldsymbol{v}\, dx - \oint_{\partial K} \left(\widehat{\boldsymbol{F}^i(\boldsymbol{u}_h)} - \boldsymbol{F}^{\widehat{v}(\boldsymbol{u}_h, \boldsymbol{q}_h)}\right) \cdot \boldsymbol{v}\, ds \quad = \boldsymbol{0}, \quad \forall \boldsymbol{v} \in [\mathcal{P}_p(K)]^5, \tag{3.5}$$

$$\int_K \boldsymbol{q}_h : \boldsymbol{\tau}\, dx + \int_K \boldsymbol{u}_h \cdot (\nabla \cdot \boldsymbol{\tau})\, dx - \oint_{\partial K} (\hat{\boldsymbol{u}}_h \otimes \boldsymbol{n}) : \boldsymbol{\tau}\, ds \quad = \boldsymbol{0}, \quad \forall \boldsymbol{\tau} \in [\mathcal{P}_p(K)]^{5\times 3}. \tag{3.6}$$

To complete the description we need to specify the numerical fluxes for all element boundaries ∂K. The inviscid fluxes $\widehat{\boldsymbol{F}^i(\boldsymbol{u}_h)}$ are computed using Roe's method [57], and the modification for our ALE formulation described in [52]. For the viscous fluxes $\widehat{\boldsymbol{F}_h^v}$, $\hat{\boldsymbol{u}}_h$, we use a formulation based on the CDG method [48], which is a slight modification of the LDG method [17] to obtain a compact and sparser stencil with improved stability properties.

First, define a *switch function* $S_K^{K'} \in \{-1, 1\}$ for each internal face e that element K shares with a neighboring element K'. We require that $S_K^{K'} = -S_{K'}^K$, but unlike the standard LDG method no other restrictions are imposed. Here we use the simple *natural switch*, which is positive if the global element number of K is greater than that of K', and negative otherwise. The numerical fluxes are then defined as follows:

- In eq. (3.6), $\hat{\boldsymbol{u}}_h$, is defined by standard "up-winding" according to the switch function:

$$\hat{\boldsymbol{u}}_h = \begin{cases} \boldsymbol{u}'_h & \text{if } S_K^{K'} = +1 \\ \boldsymbol{u}_h & \text{if } S_K^{K'} = -1, \end{cases} \tag{3.7}$$

  where $\boldsymbol{u}'_h$ is the numerical solution defined by the neighboring element K' on the face. This defines the gradients $\boldsymbol{q}_h$ for each element K.

- In eq. (3.5), the numerical fluxes $\widehat{\boldsymbol{F}_h^v}$ are defined by first introducing the "face gradients" $\boldsymbol{q}_h^e$ for each face e of K, using a slight modification of eq. (3.6):

$$\int_K \boldsymbol{q}_h^e : \boldsymbol{\tau}\, dx + \int_K \boldsymbol{u}_h \cdot (\nabla \cdot \boldsymbol{\tau})\, dx - \oint_{\partial K} (\hat{\boldsymbol{u}}_h^e \otimes \boldsymbol{n}) : \boldsymbol{\tau}\, ds = \boldsymbol{0}, \quad \forall \boldsymbol{\tau} \in [\mathcal{P}_p(K)]^{5\times 3} \tag{3.8}$$

  with

$$\hat{\boldsymbol{u}}_h^e = \begin{cases} \hat{\boldsymbol{u}}_h & \text{on face } e, \text{ from equation eq. (3.7)}, \\ \boldsymbol{u}_h & \text{otherwise.} \end{cases} \tag{3.9}$$

  These are then used to define the numerical fluxes $\widehat{\boldsymbol{F}_h^v}$ on face e:

$$\widehat{\boldsymbol{F}_h^v}^e = C_{11}(\boldsymbol{u}'_h - \boldsymbol{u}_h) + \begin{cases} \boldsymbol{F}^v(\boldsymbol{u}_h^e, \boldsymbol{q}_h^e) \cdot \boldsymbol{n} & \text{if } S_K^{K'} = +1 \\ \boldsymbol{F}^v(\boldsymbol{u}_h^{e\,\prime}, \boldsymbol{q}_h^{e\,\prime}) \cdot \boldsymbol{n} & \text{if } S_K^{K'} = -1 \end{cases} \tag{3.10}$$

where $\boldsymbol{u}_h^e{}'$, $\boldsymbol{q}_h^e{}'$ are the solutions / face gradients from the neighboring element K' on face e. Note that these fluxes can be seen as "down-winding" according to the switch function. The parameter C_{11} is used for additional stabilization, here we will use a value of $C_{11} = 10/h_{\min}$ where $h_{\min}$ is the height of the element with respect to face e.

For more details on the CDG scheme and its properties, including the compact sparsity pattern of the stencils, see [48]. At a boundary face, we impose either far field or no-slip conditions weakly through the fluxes, see [47].

We use standard finite element procedures for the discretization. We define a set of equidistributed nodes $\boldsymbol{x}_j$, $j = 1, \ldots, N_p$, within each element K, where for simplex elements $N_p = \binom{p+D}{D}$ in D spatial dimensions. We then determine the shape functions as the Lagrange interpolation functions $\phi_i(\boldsymbol{x}) \in \mathcal{P}_p(K)$ such that $\phi_i(\boldsymbol{x}_j) = \delta_{ij}$. Using these, the solution in each element can be written in terms of its discrete expansion coefficients $\boldsymbol{u}_i$ as:

$$\boldsymbol{u}_h(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{u}_i \phi_i(\boldsymbol{x}) \tag{3.11}$$

and similarly for the auxiliary variable $\boldsymbol{q}_h$, the test functions $\boldsymbol{v}, \boldsymbol{\tau}$, and the time-derivatives $\partial \boldsymbol{u}_h / \partial t$. We evaluate all integrals in eqs. (3.5) and (3.6) using high-order Gaussian quadrature rules, and setting the test function expansion coefficients to the identity matrix and eliminating the local $\boldsymbol{q}_h$ variables, we obtain the semi-discrete form of our equations:

$$\boldsymbol{M}^f \frac{d\boldsymbol{u}^f}{dt} = \boldsymbol{r}^f(\boldsymbol{u}^f), \tag{3.12}$$

for solution vector $\boldsymbol{u}^f$, mass matrix $\boldsymbol{M}^f$, and residual function $\boldsymbol{r}^f(\boldsymbol{u}^f)$.

3.2 Structure Spatial (Continuous Galerkin)

The structure equations as described in section 2.4 are discretized as follows. The domain is represented using an unstructured simplicial mesh $\mathcal{T}_h$, and curved boundaries are fit using isoparametric elements. On this mesh, we define the space of continuous piecewise polynomials of degree p:

$$V_h^p = \{\boldsymbol{v} \in [\mathcal{C}_0(\Omega)]c^3 \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^3 \ \forall K \in \mathcal{T}_h\}, \tag{3.13}$$

We also define the subspaces of functions in V_h^p that satisfy the non-homogeneous Dirichlet boundary conditions:

$$V_{h,D}^p = \{\boldsymbol{v} \in V_h^p \mid \boldsymbol{v}|_{\Gamma_D} = \boldsymbol{x}_D\}, \tag{3.14}$$

as well as the homogeneous Dirichlet boundary conditions:

$$V_{h,0}^p = \{\boldsymbol{v} \in V_h^p \mid \boldsymbol{v}|_{\Gamma_D} = 0\}. \tag{3.15}$$

By multiplying eq. (2.37) by an arbitrary test function $\boldsymbol{z} \in V_{h,0}^p$, integrating over the domain V, and applying Green's theorem, we obtain our finite element formulation: find $\boldsymbol{x}_h \in V_{h,D}^p$ such that for all $\boldsymbol{z} \in V_{h,0}^p$,

$$\int_V \rho \frac{\partial^2 \boldsymbol{x}_h}{\partial t^2} \boldsymbol{z} \, dX = -\int_V \boldsymbol{P}(\boldsymbol{F}(\boldsymbol{x}_h)) : \nabla \boldsymbol{z} \, dX + \oint_{\Gamma_N} \boldsymbol{t}(\boldsymbol{x}_h) \cdot \boldsymbol{z} \, dS + \int_V \boldsymbol{b} \cdot \boldsymbol{z} \, dX \, . \tag{3.16}$$

The system of equations eq. (3.16) is implemented using standard finite element techniques. The discrete solution vector $\boldsymbol{X}$ and the test functions are represented at the nodes using nodal basis functions. The integrals are computed using high-order Gauss integration rules. The computed elemental residuals are assembled into a global discrete residual vector $\boldsymbol{R}(\boldsymbol{X})$, to give the nonlinear ODE

$$\boldsymbol{M} \frac{d^2 \boldsymbol{X}}{dt^2} = \boldsymbol{R}(\boldsymbol{X}) \tag{3.17}$$

which we immediately convert to a first-order system by introducing the velocity $\boldsymbol{V} = d\boldsymbol{X}/dt$. The discrete positions and velocities are combined into a single solution vector $\boldsymbol{u}^s = [\boldsymbol{X}; \boldsymbol{V}]$, corresponding residual vector $\boldsymbol{r}^s(\boldsymbol{u}^s) = [\boldsymbol{V}; \boldsymbol{R}(\boldsymbol{X})]$, and mass matrix $\boldsymbol{M}^s = \mathrm{diag}(\boldsymbol{I}, \boldsymbol{M})$, to obtain the semi-discrete form of our equations:

$$\boldsymbol{M}^s \frac{d\boldsymbol{u}^s}{dt} = \boldsymbol{r}^s(\boldsymbol{u}^s). \tag{3.18}$$

3.2.1 Quasi-Static Formulation

The quasi-static non-linear elasticity equations as given in section 2.4.1 are discretized in a similar fashion. Again, the domain is represented using an unstructured simplicial mesh $\mathcal{T}_h$ and curved boundaries are fit using isoparametric elements. We repeat the process of defining V_h^p the space of continuous piecewise polynomials of degree p, and $V_{h,D}^p$ (resp. $V_{h,0}^p$) the subspaces of functions which satisfy the non-homogeneous (resp. homogeneous) Dirichlet boundary conditions.

Following a similar procedure of multiplying eq. (2.46) by an arbitrary test function $\boldsymbol{z} \in V_{h,0}^p$, integrating over the domain V, and applying Green's theorem, we obtain our finite element formulation: find $\boldsymbol{x}_h \in V_{h,D}^p$ such that for all $\boldsymbol{z} \in V_{h,0}^p$,

$$0 = \int_V \boldsymbol{P}(\boldsymbol{F}(\boldsymbol{x}_h)) : \nabla \boldsymbol{z} \, dX + \oint_{\Gamma_N} \boldsymbol{t}(\boldsymbol{x_h}) \cdot \boldsymbol{z} \, dS + \int_V \boldsymbol{b} \cdot \boldsymbol{z} \, dX. \tag{3.19}$$

The system of equations in eq. (3.19) is implemented using standard finite element techniques. The discrete solution vector $\boldsymbol{X}$ and the test functions are represented at the nodes using nodal basis functions. The integrals are computed using high-order Gauss integration rules. The computed elemental residuals are assembled into a global discrete residual vector $\boldsymbol{R}(\boldsymbol{X})$ giving the non-linear system

$$\boldsymbol{R}(\boldsymbol{X}) = 0. \tag{3.20}$$

The system is generally solved using the standard Newton-Raphson method, where given an initial guess $\boldsymbol{X}^{(0)}$ we iterate by

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} - \left[\frac{d\boldsymbol{R}}{d\boldsymbol{X}}(\boldsymbol{X}^{(k)})\right]^{-1} \boldsymbol{R}(\boldsymbol{X}^{(k)}) \tag{3.21}$$

Here the Jacobian matrix $\boldsymbol{K} = d\boldsymbol{R}/d\boldsymbol{X}$ is computed for each element and assembled into a global matrix. The prescribed displacement at the boundary nodes is enforced by elimination of the corresponding variables from the system of equations.

Alternatively, we have found that it is often convenient, especially in the implementation of our parallel solver, to assemble the global residual leaving the boundary nodes in the system. The displacement of these nodes is enforced strongly, by setting the corresponding equations of the global residual to $\boldsymbol{X}_i - \boldsymbol{x}_{D,i}$ for each node i in the Dirichlet boundary. (Here $\boldsymbol{x}_{D,i}$ is the prescribed location of that node $\boldsymbol{X}_i$ as specified in the problem). Observe that solving $\boldsymbol{R}(\boldsymbol{X}) = 0$ would give $\boldsymbol{X}_i = \boldsymbol{x}_{D,i}$ as desired. Clearly the corresponding rows in the Jacobian matrix $\boldsymbol{K}$ become the identity.

3.3 Temporal Discretization (Runge-Kutta Methods)

We use the *method of lines* to evolve the fluid and structure partial differential equations in time, by first discretizing in space as described in the previous sections and then integrating the resulting ordinary differential equation (ODE) using standard techniques.

In this section, we will generically write the ODE as

$$M\frac{du}{dt} = R(t, u) \tag{3.22}$$

$$u(t_0) = u_0 \tag{3.23}$$

where $u = [u_1, \ldots, u_N]^T$ is the vector of discretized spacial variables, M is the mass matrix, R the residual, and u_0 is the initial condition at time t_0. We will also assume some minimal requirements on the ODE to ensure wellposedness, namely M is non-singular and R is nice (e.g., Lipschitz).

The most basic ways to integrate eq. (3.22) in time are the explicit *forward Euler*

$$Mu^{(n+1)} = Mu^{(n)} + \Delta t R(t_n, u^{(n)}) \tag{3.24}$$

and implicit *backward Euler*

$$Mu^{(n+1)} = Mu^{(n)} + \Delta t R(t_{n+1}, u^{(n+1)}) \tag{3.25}$$

methods.

Of these, forward Euler is referred to as an explicit method since the expression for $u^{(n+1)}$ can be evaluated directly. On the other hand, backward Euler is referred to as an implicit

method since $u^{(n+1)}$ appears on both sides of the equation and hence must be solved for, often using the Newton-Raphson method.

The Newton-Raphson method is an iterative procedure for finding a root $f(x) = 0$ from an approximate root x_0 via a series of iterations

$$x_{k+1} = x_k - \left[\frac{df}{dx}(x_k)\right]^{-1} f(x_k) \tag{3.26}$$

until a tolerance is reached, i.e., $f(x_n) < tol$.

The Newton-Raphson method, as applied in the case of the backward Euler method, is: given an approximate solution $u_0^{(n+1)}$, repeat

$$u_{k+1}^{(n+1)} = u_k^{(n)} - \left[M - \Delta t \frac{dR}{du}(t_{n+1}, u_k^{(n+1)})\right]^{-1} \left(M u_k^{(n+1)} - M u^{(n)} - \Delta t R(t_{n+1}, u_k^{(n+1)})\right) \tag{3.27}$$

until numerical convergence is reached, i.e.,

$$\|M u_k^{(n+1)} - M u^{(n)} - \Delta t R(t_{n+1}, u_k^{(n+1)})\| < tol. \tag{3.28}$$

Note that the term in the square brackets is a matrix of the form $A = M - \alpha J$, where $\alpha = \Delta t$ and $J = dr/du$; and the term in the parenthesis is a vector b. Here it's clear that each step of the Newton method requires solving a linear system $Ax = b$.

For small, two-dimensional problems, a direct method of solving this resulting linear system is often viable. For example, one can use Gaussian Elimination (with partial pivoting) to factorize A as

$$A = PLU \tag{3.29}$$

where P is a permutation matrix, L is a unit lower triangular matrix, and U is an upper triangular matrix.

Given this factorization, the solution process for $Ax = b$ becomes

$$x = U^{-1}(L^{-1}(P^{-1}b)) \tag{3.30}$$

where each term is easy to evaluate via permutation, forward substitution, and backward substitution. It goes without saying that we suggest using existing libraries which already implement the factorization and solve algorithms like LAPACK [5] and ScaLAPACK [12] for dense matrices, and MUMPS [3, 4] and UMFPACK [20] for sparse matrices.

Before moving on, let us take a minute to rewrite the forward Euler method:

$$Mk_1 = R(t_n, u^{(n)}) \tag{3.31}$$

$$u^{(n+1)} = u^{(n)} + \Delta t k_1 \tag{3.32}$$

and backward Euler method:

$$Mk_1 = R(t_n + \Delta t, u^{(n)} + \Delta t k_1) \tag{3.33}$$
$$u^{(n+1)} = u^{(n)} + \Delta t k_1. \tag{3.34}$$

Here we have introduce auxiliary variable k_1 which records the slope at a specific time. This idea of recording intermediate slopes naturally extends to multiple stages in Runge-Kutta methods. Consider, for example, the classical Runge-Kutta method, RK4:

$$Mk_1 = R(t_n, u^{(n)}) \tag{3.35}$$
$$Mk_2 = R(t_n + \Delta t/2, u^{(n)} + \Delta t/2 k_1) \tag{3.36}$$
$$Mk_3 = R(t_n + \Delta t/2, u^{(n)} + \Delta t/2 k_2) \tag{3.37}$$
$$Mk_4 = R(t_n + \Delta t, u^{(n)} + \Delta t k_3) \tag{3.38}$$
$$u^{(n+1)} = u^{(n)} + \Delta t(k_1/6 + k_2/3 + k_3/3 + k_4/6). \tag{3.39}$$

This is a 4-stage, 4-th order, explicit method.

We can generalize this framework to allow for other coefficients as:

$$Mk_i = R(t_n + c_i \Delta t, u^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij} k_j) \quad \text{for } i = 1, \ldots, s \tag{3.40}$$

$$u^{(n+1)} = u^{(n)} + \Delta t \sum_{i=1}^{s} b_i k_i \tag{3.41}$$

where s is the number of stages. We call the s-by-s matrix a_{ij} the coefficients, the vector b_i the weights, and the vector c_i the nodes.

These Runge-Kutta methods are generally written as a Butcher tableau:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}. \tag{3.42}$$

For example, forward and backward Euler may be written as

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \quad \text{and} \quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}, \tag{3.43}$$

and the RK4 scheme as:

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & & 1/2 & & \\ 1 & & & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{3.44}$$

where missing entries should be interpreted as 0.

Certain properties of the Runge-Kutta scheme may be read directly from the coefficients. For example, one can determine if the scheme is explicit (ERK) or implicit (IRK) depending on the matrix of coefficients a_{ij}. The method is explicit if the coefficients are strictly lower-triangular. If this is the face, we can first solve for k_1, then k_2, etc, each by explicit evaluation (and possibly a mass matrix solve).

Otherwise the scheme is called implicit and will require a non-linear iterative solve procedure like the one discussed for the backward Euler method. Note that if the coefficient matrix is dense each k_i will depend on all of the other k_j, meaning that we will likely need to use one big Newton-Raphson non-linear solve on all of the k_i simultaneously. At each step this will require the solve of a (Ns)-by-(Ns) linear system which may be prohibitively expensive.

To get around this we can use a diagonally implicit Runge-Kutta (DIRK) scheme where the coefficient matrix a_{ij} is lower-triangular. Here it's clear that k_1 only depends on k_1; k_2 depends on k_1 and k_2; and k_i depends only on k_1 through k_i. Because of this, we can first solve for k_1 using Newton-Raphson on a smaller N-by-N system, then solve for k_2, then k_3, etc. In other words, we now require s non-linear solves of an N-by-N system, rather than 1 non-linear solve of an (Ns)-by-(Ns) system.

In the class of DIRK schemes, we call those whose non-zero diagonal coefficients are all equal *singly diagonally implicit* (SDIRK) schemes. If in addition the first stage is explicit (i.e., $a_{11} = 0$), we call the scheme *explicit first stage, singly diagonally implicit* (ESDIRK).

As a rather trivial example of an ESDIRK scheme consider the trapezoidal rule:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}. \tag{3.45}$$

3.3.1 Implicit-Explicit (IMEX) Runge-Kutta Schemes

Consider for the moment a convection-diffusion type partial differential equations

$$u_t = uu_x + \nu \Delta u \tag{3.46}$$

which, after discretizing the spatial derivatives, gives an ordinary differential equation

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{u}). \tag{3.47}$$

Here $\boldsymbol{f}$ corresponds to the convection term uu_x and $\boldsymbol{g}$ corresponds to the diffusion term $\nu\Delta u$. We know that the properties of $\boldsymbol{f}$ and $\boldsymbol{g}$ are very different. For example, $\boldsymbol{f}$ is non-linear and the ODE $d\boldsymbol{u}/dt = \boldsymbol{f}(\boldsymbol{u})$ is generally not very stiff. On the other hand, $\boldsymbol{g}$ is linear and the ODE $d\boldsymbol{u}/dt = \boldsymbol{g}(\boldsymbol{u})$ is generally stiff. Here it would be natural to use an explicit scheme to integrate $\boldsymbol{f}$ and an implicit scheme to integrate $\boldsymbol{g}$.

Implicit-Explicit Runge-Kutta methods, as introduced in ref. [7], propose using an explicit Runge-Kutta method to integrate $\boldsymbol{f}$ together with a paired diagonally implicit Runge-Kutta method to integrate $\boldsymbol{g}$. As is traditional, we use the regular notation to denote the coefficients of the diagonally implicit scheme (a_{ij}, b_i, c_i) and a hat to denote the coefficients of the explicit scheme ($\hat{a}_{ij}$, $\hat{b}_i$, $\hat{c}_i$). In general, and in every case we consider, the nodes are common between the two schemes, i.e., $c = \hat{c}$. Since the first stage of an explicit scheme has $\hat{c}_1 = 0$, this means that the first stage of the implicit scheme is also typically explicit. In particular, most IMEX Runge-Kutta schemes are a pairing of an explicit (ERK) method and an ESDIRK method.

The implicit-explicit Runge-Kutta scheme is given by the definitions of the stage solution:

$$\boldsymbol{u}_i^{(n)} = \boldsymbol{u}^{(n)} + \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij} \hat{\boldsymbol{k}}_j + \Delta t \sum_{j=1}^{i} a_{ij} \boldsymbol{k}_j \qquad \text{for } i = 1, \ldots, s \tag{3.48}$$

and slopes:

$$M\hat{\boldsymbol{k}}_i = \boldsymbol{f}(\boldsymbol{u}_i^{(n)}) \quad \text{and} \quad M\boldsymbol{k}_i = \boldsymbol{g}(\boldsymbol{u}_i^{(n)}) \qquad \text{for } i = 1, \ldots, s. \tag{3.49}$$

The solution is advanced to the next time step via

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \Delta t \sum_{i=1}^{s} \hat{b}_i \hat{\boldsymbol{k}}_i + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i. \tag{3.50}$$

In terms of implementation, one can trace through the equations to see that one can first solve for $\boldsymbol{k}_1$, then evaluate $\hat{\boldsymbol{k}}_1$, then solve for $\boldsymbol{k}_2$, then evaluation $\hat{\boldsymbol{k}}_2$, and so on. This corresponds to, for each stage, first doing an implicit solve of $\boldsymbol{g}$ followed by an implicit evaluation of $\boldsymbol{f}$. The algorithm is written out fully in algorithm 3.1.

Algorithm 3.1 Implicit-explicit Runge-Kutta method.

Input: Ordinary differential equation $M(d\boldsymbol{u}/dt) = \boldsymbol{f}(t, \boldsymbol{u}) + \boldsymbol{g}(t, \boldsymbol{u})$.
Input: Numerical solution $\boldsymbol{u}^{(n)}$ at time t_n.
Input: Timestep $\Delta t > 0$.
for stages $i = 1, \ldots, s$ **do**
Define the stage solution $\boldsymbol{u}_i^{(n)} = \boldsymbol{u}^{(n)} + \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij} \hat{\boldsymbol{k}}_j + \Delta t \sum_{j=1}^{i} a_{ij} \boldsymbol{k}_j$.
Solve for $\boldsymbol{k}_i$ in $M\boldsymbol{k}_i = \boldsymbol{g}(t_n + c_i \Delta t, \boldsymbol{u}_i^{(n)})$. # Implicit solve of $\boldsymbol{g}$.
Solve for $\hat{\boldsymbol{k}}_i$ in $M\hat{\boldsymbol{k}}_i = \boldsymbol{f}(t_n + c_i \Delta t, \boldsymbol{u}_i^{(n)})$. # Explicit evaluation of $\boldsymbol{f}$.
end for
Set $\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \Delta t \sum_{i=1}^{s} \hat{b}_i \hat{\boldsymbol{k}}_i + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i$.
Output: Numerical solution $\boldsymbol{u}^{(n+1)}$ at time $t_{n+1} = t_n + \Delta t$.

There are many examples of implicit-explicit Runge-Kutta schemes. For example, *forward backward Euler*:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1 & 0 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & & \\ 1 & 0 & 1 \\ \hline & 0 & 1 \end{array} \tag{3.51}$$

where the tableau on the left is the explicit scheme ($\hat{a}_{ij}$, $\hat{b}_i$, $\hat{c}_i$) and the tableau on the right the implicit scheme (a_{ij}, b_i, c_i). Note here that unlike many other examples $\hat{b} \neq b$.

A second order example is the trapezoidal rule [41]:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & & \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}. \tag{3.52}$$

Many popular IMEX coefficients are due to Kennedy & Carpenter, who in ref. [40] provided schemes of orders 3, 4, and 5 which have many desirable properties including for the ESDIRK coefficients L-stability, stage order 2 and stiff accuracy. The schemes are named ARK3(2)4L[2]SA, ARK4(3)6L[2]SA, and ARK5(4)8L[2]SA, according to the naming convention ARK$q(p)sS[q_{SO}]X$ where q is the order, p is the order of an embedded method, s is the number of stages, S is a characterization of the stability, q_{so} is the stage-order of the implicit method, and X is any other notable trait. This means, for example, that the 3rd order method has 4 stages; the 4th order method has 6 stages; and the 5th order method has 8 stages. The coefficients for the 3rd order method are reproduced in table 3.1. See ref. [40] for the coefficients of the other schemes. In this document we will generally abbreviate the scheme names, calling them ARK3, ARK4, and ARK5 respectively.

3.4 Implicit Solvers

The systems of equations produced by the DG discretization of the fluid system are typically very large. We often use polynomials of degree $p = 3$ or higher which gives a large number of degrees of freedom per element and solution component. These numbers are summarized in table 3.2.

Recall that in three dimensions the Navier-Stokes equations have 5 solution components per node. This means that a $p = 3$ tetrahedral element will have 100 degrees of freedom per element. At typical mesh consisting of tens or hundreds of thousands elements would then have millions of degrees of freedom, as for example in section 5.2.

3.4.1 Parallel Newton-Krylov Fluid Solvers

The IMEX Runge-Kutta scheme as discussed in section 3.3.1 requires both explicit residual evaluations and implicit non-linear solves. For the following descriptions we assume the fluid

Table 3.1: The 3rd order implicit-explicit Runge-Kutta coefficients ARK3(2)4L[2]SA.

0				
$\frac{1767732205903}{2027836641118}$	$\frac{1767732205903}{2027836641118}$			
$\frac{3}{5}$	$\frac{5535828885825}{10492691773637}$	$\frac{788022342437}{10882634858940}$		
1	$\frac{6485989280629}{16251701735622}$	$\frac{-4246266847089}{9704473918619}$	$\frac{10755448449292}{10357097424841}$	
	$\frac{1471266399579}{7840856788654}$	$\frac{-4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$

(a) Explicit Runge-Kutta (ERK) coefficients.

0				
$\frac{1767732205903}{2027836641118}$	$\frac{1767732205903}{4055673282236}$	$\frac{1767732205903}{4055673282236}$		
$\frac{3}{5}$	$\frac{2746238789719}{10658868560708}$	$\frac{640167445237}{6845629431997}$	$\frac{1767732205903}{4055673282236}$	
1	$\frac{1471266399579}{7840856788654}$	$\frac{-4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$
	$\frac{1471266399579}{7840856788654}$	$\frac{-4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$

(b) Explicit first stage, singly diagonally implicit Runge-Kutta (ESDIRK) coefficients.

Table 3.2: The number of high-order nodes per element for simplicial (triangles and tetrahedra) and quad/hex meshes of various polynomial orders.

	2D		3D	
Polynomial Order	Triangles	Quadrilaterals	Tetrahedra	Hexahedra
$p = 2$	6	9	10	27
$p = 3$	10	16	20	64
$p = 4$	15	25	35	125
$p = 5$	21	36	56	216
$p = 6$	28	49	84	343
$p = 7$	36	64	120	512

system has been discretized in space to give an ordinary differential equation

$$\boldsymbol{M}^f \frac{d\boldsymbol{u}^f}{dt} = \boldsymbol{r}^f(\boldsymbol{u}^f). \tag{3.53}$$

Of course the right hand side may rely additionally on some external fields, e.g., body forces, mesh deformation in an ALE formulation, etc, but for brevity these are not shown.

Explicit stages

The explicit evaluation is relatively straightforward. Essentially, given the current fluid state $\boldsymbol{u}^f$ solve for $\boldsymbol{k}^f$ in

$$\boldsymbol{M}^f \boldsymbol{k}^f = \boldsymbol{r}^f(\boldsymbol{u}^f). \tag{3.54}$$

Here the right hand side is trivially evaluated on an element by element basis. Here we parallelize the computation by decomposing the domain the into n_{proc} partitions, each containing some subset of the elements. For the flux computations along each element face the solver needs to know the state of the element across that face. Computationally this means that each process needs to maintain up-to-date values of neighboring elements one layer outside of the local partition.

Clearly we can overlap the communication of data with residual computations, so the algorithm employed is:

- Evaluate the residual on each element which shares a face with a neighboring partition.
- Set up asynchronous send and receive operations for the element-wise residuals with neighboring partitions.
- Evaluate the residuals on remaining elements.
- Wait for the send and receive operations to complete.

To decompose the domain we first we build the element adjacency graph, where vertices correspond to elements and edges occur when two elements share a common face. We then use the METIS software [38] to create the partitioning. In general the software attempts to create partitions of roughly equal size with minimal surface area (since surface area corresponds to the amount of data which must be sent during the residual evaluation).

Since the mass matrix $\boldsymbol{M}^f$ is block diagonal, the inverse can be applied in parallel on a per-element basis.

Implicit stages

An implicit stage of the Runge-Kutta method requires a non-linear solve for $\boldsymbol{k}^f$ in an equation of the form

$$\boldsymbol{M}^f \boldsymbol{k}^f = \boldsymbol{r}^f(\boldsymbol{u}^f + \alpha \boldsymbol{k}^f) \tag{3.55}$$

where at stage i we have $\alpha = \Delta t a_{ii}$ and $\boldsymbol{u}^f = \boldsymbol{u}_0^f + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j^f$. To complete the non-linear solve we use a Newton-GMRES method. The outer Newton solve requires linear solutions to equations of the form

$$(\boldsymbol{M}^f - \alpha \boldsymbol{K}) \boldsymbol{\delta k}^f = \boldsymbol{r} \tag{3.56}$$

where $\boldsymbol{K} = d\boldsymbol{r}^f / d\boldsymbol{u}^f$ is the Jacobian. Here we solve the linear system using an iterative preconditioned GMRES method.

Since the linear system is quite large, one would prefer to avoid computing and storing the sparse matrix $\boldsymbol{M}^f - \alpha \boldsymbol{K}$ and instead compute the required matrix-vector products on the fly using a so-called matrix free method. However these methods are often difficult to precondition.

Even worse, for these high order methods the Jacobian matrices ($d\boldsymbol{r}^f / d\boldsymbol{u}^f$) tend to be less sparse than their low order counterparts. For a typical problem of the compressible Navier-Stokes equations in three dimensions which is discretized using polynomials of degree $p = 3$ on a mesh of a hundred thousand tetrahedra, a simple calculation shows that the resulting system will have 10 million degrees of freedom and each Jacobian will have about 3 billion entries requiring 24 GB of storage. It is clear that parallel computers are needed, both for storing these matrices and to perform the computations.

In our highly parallel 3DG implementation [51], we compute and store the Jacobian in a specialized block storage format. Here we have a series of diagonal blocks, one for each element, and a series of off-diagonal blocks, two for each interior face (one in the lower triangular part and one in the upper triangular part). In the parallel partitioning scheme, each processor owns all of the rows corresponding to the degrees of freedom associated with the nodes in that partition. Note that it does not store rows associated with neighboring partitions.

The matrix-vector products required for each GMRES iteration are computed in the obvious fashion. Much like the explicit residual evaluation, some communication is required after each matrix vector product to transfer element data across the faces on the boundary of each partition.

There are many ways to precondition the linear system. We generally use one of two preconditioners:

- *Block Jacobi:* This preconditioner relies on approximating the matrix by its diagonal blocks, i.e., $m_{ij} = a_{ij}$ if i and j are degrees of freedom which lie in the same element and $m_{ij} = 0$ otherwise. This preconditioner is useful for ensuring that the equations have a similar scaling. In parallel this preconditioner scales perfectly since it is a purely local operation.

- *Block ILU(0):* Here we create an approximate LU decomposition of the matrix by using the standard Gaussian elimination algorithm on each block of the matrix, but insist that no additional fill is introduced along the way. That is, the block sparsity pattern of $L + U$ is the same as the sparsity pattern of the original matrix. Gaussian

elimination is a fundamentally sequential operation so this cannot be used in parallel. As a compromise we can do block ILU(0) on the rows corresponding to each partition separately. In this way, no communication is required. However, as the number of the partitions approaches the number of elements, i.e., as the number of elements per partition goes to one, this method becomes equivalent to block Jacobi.

More complicated p-multigrid and $ILU(k)$ schemes have been implemented, but experience for a wide range of problems leads us to generally use block Jacobi or block ILU(0). For more details see [53].

In addition to choosing the preconditioner, we can also optimize the original partitioning by providing edge weights in the adjacency graph as suggested in [53]. This can cause the partitioning to vaguely follow streamlines of the solution and can reduce the number of GMRES iterations by a small factor. In addition we order the elements using a Minimum Discarded Fill (MDF) algorithm when using the block ILU(0) preconditioner.

The resulting simulation time for a particular problem is heavily dependent upon the number of processors used, the problem, the timestep, and the tolerances in the Newton and Krylov solvers. Even the order of the chosen Runge-Kutta method can have a large impact. For a large discussion on these points alone, see ref. [72]. For the results presented in this work used anywhere between a handful of cores on a local machine for low fidelity 2D simulations to over 2048 cores for large 3D simulations using the Hopper and Edison systems at the National Energy Research Scientific Computing Center (NERSC). The longest simulations took approximately 24 hours to run, often broken into smaller 6 hour or 1 hour increments to get through the batch queuing system faster.

3.4.2 Sparse Direct Structure Solvers

The system of equations produced by the CG discretization of the structure is generally much smaller than the that of the fluid, both because of structure domain is physically smaller and because the CG discretization avoids the duplicate nodes which would appear in a DG discretization. Nonetheless, we still find it expedient to use a parallel code, again based on MPI. The structure domain is decomposed using the METIS software [38], and the discretization and matrix assembly are each done in parallel. In order to solve the linear systems arising from Newton's method, we use the MUMPS [3, 4] software package providing the matrix in distributed coordinate form.

Explicit Stages

In the explicit stages of the Runge-Kutta scheme we require computing the stage derivative $\boldsymbol{k}^s$ from the stage state $\boldsymbol{u}^s$ as

$$\boldsymbol{M}^s \boldsymbol{k}^s = \boldsymbol{r}^s(\boldsymbol{u}^s). \tag{3.57}$$

We can compute this in an almost identical way as in the DG fluid case. Each process computes the local element-wise residual on the elements within that partition. The local

results are then assembled together into the global residual via addition using the so-called *stamping method*. One key difference when compared to a discontinuous Galerkin method is the mass matrix is no longer block diagonal. There are many ways to do the requisite mass matrix solve, including iterative methods which can be quite competitive since the mass matrix is symmetric positive definite and generally well conditioned. Here, however, we choose to prefactorize the mass matrix using the MUMPS [3, 4] multifrontal parallel sparse direct solver.

Implicit Stages

In the implicit stages of the Runge-Kutta scheme we compute the stage derivative as

$$\boldsymbol{M}^s \boldsymbol{k}^s = \boldsymbol{r}^s(\boldsymbol{u}^s + \alpha \boldsymbol{k}^s) \tag{3.58}$$

where again at stage i we have $\boldsymbol{u}^s = \boldsymbol{u}_0^s + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j^s$ and $\alpha = \Delta t a_{ii}$. We use a Newton-Raphson procedure to complete the non-linear solve, which at each iteration requires the solution of a linear system of the form

$$(\boldsymbol{M}^s - \alpha \boldsymbol{K}^s)\boldsymbol{\delta k}^s = \boldsymbol{b} \tag{3.59}$$

where $\boldsymbol{K}^s = d\boldsymbol{r}^s/d\boldsymbol{u}^s$ is the Jacobian. For convenience we again use the MUMPS solver to complete the linear solve, providing the matrix to the software package in a distributed coordinate format. That is, each process provides their local portion of the matrix as a list of entries $(i,\ j,\ A_{ij})$ where i and j are the global coordinates of the entry and A_{ij} is the value. This assembly process is done element by element, each process providing entries corresponding to elements in that region. Duplicate entries which occur along element boundaries are automatically summed by MUMPS as in the stamping method.

Since the sparsity pattern of the resulting matrix does not change between subsequent solves, we can have MUMPS precompute and reuse a symbolic factorization. The numeric factorization phase must of course be repeated since the matrix entries will have changed.

We remark that this direct solver is not particularly competitive, especially in 3D, and that other means of assembling and solving the linear system are likely to be more efficient. Nonetheless, we chose this implementation because it was easy to add to the existing 3DG software and provides adequate performance for problems of interest. Many applications discussed in this work have structures which, despite being fully three-dimensional, behave more as a two-dimensional object because they are just one element thick in the third dimension. In this regime direct solvers can be competitive with iterative solvers.

In terms of implementation, the CG structure code reuses the generic DG framework with internal boundary fluxes disabled. The resulting element-wise residuals and Jacobians are assembled into global residuals and Jacobians using the stamping method,

$$r_{cg} = W^T r_{dg} \quad \text{and} \quad K_{cg} = W^T K_{dg} W, \tag{3.60}$$

where W is the natural operator which takes a CG solution and returns a DG solution by duplicating the values at repeated nodes.

Chapter 4

Fluid-Structure Interaction

4.1 Coupling

In a fluid-structure interaction problem, a primary consideration is the coupling between the fluid and the structure. Generally we treat the fluid as exerting a surface traction, i.e., force per unit area, on the boundary of the structure. The structure, in response to the fluid and its own internal dynamics deforms the domain in which the fluid is modeled.

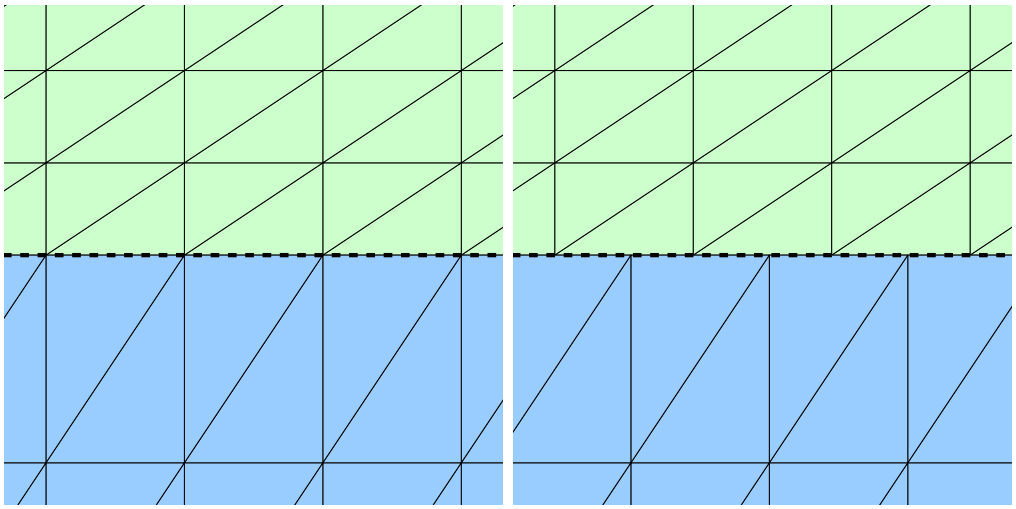
In situations where the structure is deformable and modeled using a finite element discretization, we require that the fluid mesh and structure mesh along the fluid-structure interface be face-wise matching. As shown in fig. 4.1, two meshes $\mathcal{T}_1$ and $\mathcal{T}_2$ are face-wise matching if each face along their interface Γ_{12} is a proper face of each mesh. That is,

$$\partial\mathcal{T}_1 \cap \Gamma_{12} = \partial\mathcal{T}_2 \cap \Gamma_{12}. \tag{4.1}$$

In addition, we require that the fluid and structure be discretized using the same polynomial order.

This approach offers two key benefits which we exploit. First, boundary data is easily passed between the fluid and structure meshes, at either the solution nodes or Gauss integration nodes, by a simple pre-computed lookup table and without any interpolation. Second, the deformed fluid mesh is easily constructed to be exactly conformal to a given deformed structure position.

The main drawback of this approach is that we are no longer free to choose the meshes for the fluid and structure independently. This adds an additional level of complexity to the process of mesh generation, but is generally surmountable. In addition, element size constraints essentially leak across the boundary. For example, if the discretized fluid requires many tiny elements near the boundary to resolve fine features, the structure may be required to have an undesirably large number of elements. Note that this constraint applies only to the mesh size in directions tangent to the interface. For example, one can create a boundary layer of highly anisotropic elements in the fluid mesh without imposing additional constraints on the structure mesh.



(a) Matching interface (b) Non-matching interface

Figure 4.1: Two meshes which are face-wise matching (left) and non-matching (right) along the interface, shown as a dashed line.

4.1.1 Fluid-to-structure coupling

We begin by describing the coupling from the fluid to the solid, which differs depending on the specifics of the structure model.

For example, if the structure is modeled as a rigid body we need only compute the total force and torque which the fluid applies to the structure. Recall that these quantities may be obtained by integrating the surface traction, i.e., momentum flux, over the entirety of the fluid-structure interface. Specifically, we obtain a total force

$$\boldsymbol{F} = \oint_{\partial V} \boldsymbol{t}\, dA \tag{4.2}$$

and torque

$$\boldsymbol{\tau} = \oint_{\partial V} (\boldsymbol{x} - \boldsymbol{x}_0) \times \boldsymbol{t}\, dA \tag{4.3}$$

where $\boldsymbol{x}_0$ is a point about which the torque is measured.

Numerically, we compute these integrals using Gaussian integration. For each face e in the fluid-structure interface Γ_s, we compute the position $\boldsymbol{x}_i^{\{e\}}$ and traction $\boldsymbol{t}(\boldsymbol{x}_i^{\{e\}})$ for each of the $i = 1, \ldots, n_g$ Gauss nodes. These quantities are then summed to produce the total force:

$$\boldsymbol{F} = \sum_{e \in \Gamma_s} \sum_{i=1}^{n_g} w_i^{\{e\}} \boldsymbol{t}(\boldsymbol{x}_i^{\{e\}}). \tag{4.4}$$

The numerical calculation of the torque is done analogously.

If the structure is not a rigid body and is instead allowed to deform, the the fluid to structure coupling consists of a traction applied along the boundary of the structure domain. The traction, computed as the momentum flux through the boundary of the fluid mesh, is computed using the fluid state variables and transfered to the structure mesh. Since we have assumed that the faces are shared between fluid and structure meshes along the interface, this transfer can be done without any interpolation. To ensure the highest possible accuracy this transfer is done pointwise at the Gauss integration nodes, not the solution nodes.

4.1.2 Structure-to-fluid coupling

The structure-to-fluid coupling is a deformation of the fluid mesh in response to a change in the structure position. We represent the deformed fluid mesh and mapping velocity on each element of the fluid mesh in an isoparametric fashion, i.e., by using polynomials of the same order as the fluid discretization. This means that the deformed fluid mesh may *exactly* conform to the deformed structure by setting the positions of the boundary fluid nodes to the deformed position of the structure. For deformations much smaller than the size of an element no additional work is required. However larger deformations may cause the mesh to invert if the interior nodes are not moved as well.

There are several methods of deforming meshes in response to moving boundaries. Here we will discuss two possible options, radial basis function interpolation and quasi-static elasticity deformation.

Radial Basis Function Interpolation

Radial basis function (RBF) interpolation for mesh deformation has been well studied [9, 13]. In general this method of deforming the mesh is straightforward to implement, reasonable to parallelize, and works well for small to moderate mesh deformations. On the other hand, the resulting quality of the deformed mesh may be quite poor (and possibly inverted) for severe deformations or poor selection of the various parameters.

Radial basis function interpolation seeks to create a global approximation of a function from values given pointwise in some regions of the domain. The basic premise is that the value at a specific point should strongly influence the area near that point and weakly influence areas distant from the point. Here the rate of decay of influence depends only on the radial distance between the source point and the measurement point. In addition, RBF interpolation usually includes a polynomial term which we will think of as capturing global translations and rotations.

In this context, we seek an interpolant giving the deformed fluid mesh position $\boldsymbol{x}$ as a function of the position $\boldsymbol{X}$ in the reference fluid mesh, and which is of the form

$$\boldsymbol{x}(\boldsymbol{X}) = \sum_{j=1}^{n} \boldsymbol{\alpha}_j \phi_j(\|\boldsymbol{X} - \boldsymbol{X}_j\|_2 / r_j) + \boldsymbol{p}(\boldsymbol{X}) \tag{4.5}$$

where $\boldsymbol{X}_j$ are a set of control points, ϕ_j radial basis functions, r_j characteristic radii, and $\boldsymbol{p}$ a linear polynomial. We generally restrict ourselves to the case where all of the radial basis functions and characteristic radii are equal, the indices from those terms to indicate that they do not vary.

The coefficients $\boldsymbol{\alpha}_j$ and coefficients of the polynomial $\boldsymbol{p}$ are found by imposing the value of $\boldsymbol{x}$ at the control points $\boldsymbol{X_j}$:

$$\boldsymbol{x}_j = \boldsymbol{x}(\boldsymbol{X}_j) \quad \text{for } j = 1, \ldots, N \tag{4.6}$$

and additionally requiring

$$\sum_{j=1}^{N} \boldsymbol{\alpha}_j \boldsymbol{q}(\boldsymbol{X}_j) = 0 \tag{4.7}$$

for all polynomials $\boldsymbol{q}$ of degree less than or equal to the degree of $\boldsymbol{p}$, i.e. 1.

This gives rise to a linear system

$$\begin{bmatrix} M & \phi \\ \phi^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x} \\ 0 \end{bmatrix} \tag{4.8}$$

where $M_{ij} = \phi(\|\boldsymbol{X}_j - \boldsymbol{X}_i\|_2 / r)$ and $\phi = [1, X]$. Note that if there are n_b boundary nodes in dimension d, this gives a linear system of size $(n_b + d + 1)$-by-$(n_b + d + 1)$. There are many ways to solve such a system, including both iterative techniques [26] and direct methods. Here the control points $\boldsymbol{X}_j$ are the nodes on the boundary of the fluid mesh, both along the fluid-structure interface and at far boundaries.

Clearly the properties of the coefficient interpolation matrix will depend heavily on the radial basis function and problem. For example, in our work we have considered compactly supported radial basis functions ϕ, like the C^2 function

$$\phi(r) = \begin{cases} (1-r)^4(4r+1) & \text{if } 0 \le r \le 1 \\ 0 & \text{if } 1 \le r. \end{cases} \tag{4.9}$$

This leads to some sparsity in the resulting matrix as the radial basis function is zero whenever $\|\boldsymbol{X}_j - \boldsymbol{X}_i\|_2 \ge r$. Other common radial basis functions are not compactly supported, including the inverse quadratic

$$\phi(r) = \frac{1}{1 + r^2}. \tag{4.10}$$

Finally we note that the radial basis function need not even have positive weight at the origin. For example, the thin plate spline

$$\phi(r) = \begin{cases} r^2 \log r & \text{if } r > 0 \\ 0 & \text{if } r = 0 \end{cases} \tag{4.11}$$

is supported entirely away from the origin.

Since we need to solve for the mesh deformation many times per simulation, we have found it expedient to use a direct method to prefactorize the coefficient matrix in eq. (4.8) to minimize the amount of time later spent during the simulation performing the linear solve. Note that this is possible in our case because the control points all lie on the boundary of the fluid domain, a co-dimension one surface. This means that there are comparatively fewer such nodes and in general the prefactorization procedure becomes a relatively small portion of the overall simulation time.

Nevertheless we generally use the parallelized LU factorization routines in ScaLAPACK [12] to both further reduce computation time and alleviate memory pressure by spreading the matrix over several compute nodes.

After the coefficients $\boldsymbol{\alpha}_j$ and polynomial $\boldsymbol{p}$ have been computed the interior nodes of the deformed mesh are easily computed using eq. (4.5). Depending on the actual form of the radial basis function, the quantity $\phi(\|\boldsymbol{X} - \boldsymbol{X}_j\|_2/r)$ may be somewhat expensive to compute. If sufficient memory is available this can be avoided by precomputing the values for each mesh node $\boldsymbol{X}$ and storing the result as a two-dimensional matrix of shape n-by-n_b where n is the number of (local) mesh nodes and n_b is the number of (global) boundary nodes. The final interpolant can then be computed via an efficient matrix vector product.

In our framework, the deformed positions of the boundary nodes on the fluid-structure interface are specified by the displacement of the structure at that node. Nodes along other boundaries, e.g., far-field boundaries, can be set arbitrarily, however we generally observed the best results when those nodes where held fixed in their reference position.

Observe that for a fixed (interior) node $\boldsymbol{X}$ the radial basis function interpolant in eq. (4.5) is linear in the displacement of the boundaries $\boldsymbol{x}_j$. This means that the mapping velocity,

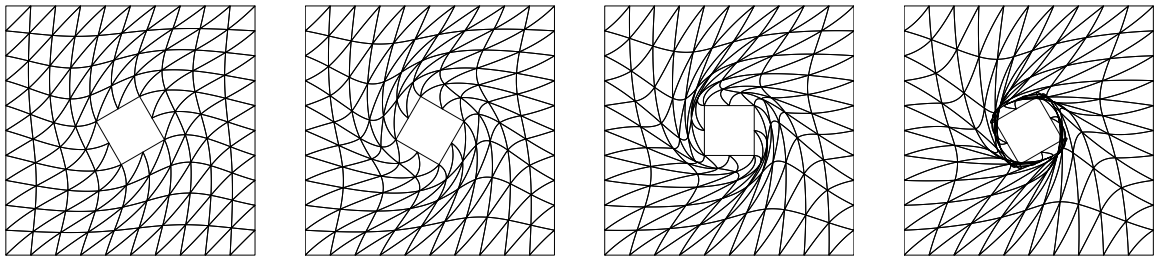
$$\boldsymbol{v}(\boldsymbol{X}) = \sum_{j=1}^{N} \frac{d\boldsymbol{\alpha}_j}{dt} \phi(\|\boldsymbol{X} - \boldsymbol{X}_j\|_2/r_j) + \frac{d\boldsymbol{p}}{dt}(\boldsymbol{X}), \tag{4.12}$$

may be seen as the field given by an interpolation process of the boundary velocities instead of the boundary positions. In fact, given the boundary positions and velocities, it is often easy to perform both interpolations simultaneously to take advantage of additional efficiencies present in the BLAS-3 routines.

As a test problem, consider a square domain with a square removed from the center:

$$\Omega = [0.0, 1.0]^2 \setminus [0.4, 0.6]^2. \tag{4.13}$$

The domain is triangulated in a structured fashion using isoparametric elements of polynomial degree 2. We fix the outer boundary of the domain and rotate the inner boundary about the middle, $[0.5, 0.5]^T$, by and angle θ. We select the C^2 compactly supported radial basis function in eq. (4.9) with characteristic radius 1, which gave the best results for this deformation for several different RBF interpolants and radii examined. The resulting deformed mesh for rotations of 30°, 60°, 90°, and 120° are shown in fig. 4.2.



(a) 30° (b) 60° (c) 90° (d) 120°

Figure 4.2: Mesh deformation using radial basis function interpolation. The radial basis function is as specified in eq. (4.9) with r equal to the edge length of the outer square.

Here we see that this mesh deformation method does a very good job with the small deformation (30°), but has some difficulty with larger deformations. In particular, some elements have already inverted (i.e., the determinant of the local Jacobian mapping is negative) by 90°.

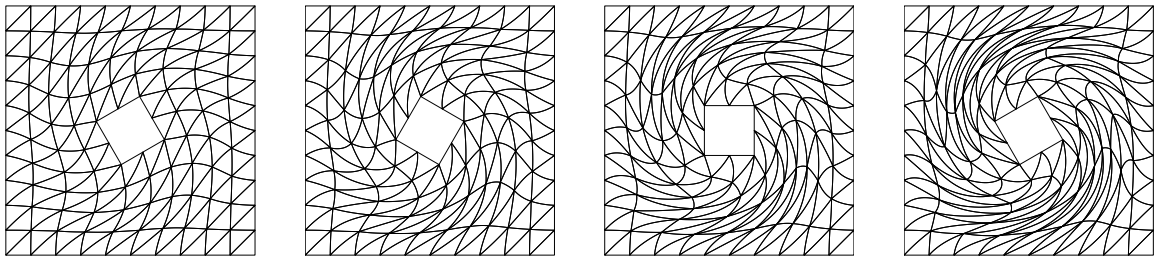
In addition one can easily show that a non-inverting deformation of a 180° rotation of the inner square is not possible using this technique for any choice of radial basis function interpolant. To see this, recall that the deformed position of any node depends linearly on the positions of the boundary nodes. Since a $+180°$ and $-180°$ rotation of the inner square would lead to the same locations of the boundary nodes, the RBF interpolant is unable to distinguish between these two cases. In particular, a curve connecting the left outer boundary to the left inner boundary in the undeformed mesh would have to pass both under the square in the $+180°$ rotation and under the square in the $-180°$ rotation, which is not possible.

One way of avoiding this issue is to continually redefine the reference mesh as the mesh at the previous step. In this way we can construct large global deformations out of only small deformations, however it is still difficult to guarantee a high element quality and the additional cost of recomputing the RBF interpolation matrix may make the method uncompetitive.

Quasi-Static Non-Linear Elasticity

Another method of mesh deformation is by a quasi-static non-linear elasticity model. In analogy to previously proposed linear elasticity method for mesh deformation [62], here we imagine that the entire fluid domain is a non-linear elastic structure. We impose Dirichlet boundary conditions corresponding to the desired boundary deformations and solve the non-linear system of equations as given in section 3.2.1.

There are two main benefits of this method over radial basis function interpolation. First, the method is generally able to handle larger and more severe deformations. Second, we have much greater and more intuitive control over the resulting deformation by varying the physical parameters in the strain energy density eq. (2.40). For example, as Poisson's ratio



(a) 30° (b) 60° (c) 90° (d) 120°

Figure 4.3: Mesh deformation using a quasi-static non-linear elasticity method.

ν increases to 0.5 the material becomes incompressible and this may help preserve higher element qualities at the expense of making the resulting non-linear system harder to solve. Also we can vary Young's modulus E throughout the domain, choosing low values in regions able to withstand large deformation and large values in regions where the deformation should be minimized. One obvious thing to do here is make E large near the domain boundary and small in the interior. This could be done using a distance function or by a proxy such as element size or height. The former case allows better control and often gives better results but requires more effort as a distance function must be created from scratch unless one can be reused from the mesh generation process, say from using a tool like DistMesh [49, 50].

In fig. 4.3 we repeat the experiment of rotating and inner square inside of a fixed box, this time using the non-linear elasticity deformation method. Here we set $\nu = 0.40$ and a spatially varying E according to

$$E(x) = 1 + \frac{100}{1 + (d(x)/d_0)^2} \tag{4.14}$$

where $d_0 = 0.05$ and

$$d(x) = \max\left\{0.0, \min\left(\text{dist}(x, \Gamma_{in}) - d_0, \text{dist}(x, \Gamma_{out}) + 2d_0\right)\right\} \tag{4.15}$$

where Γ_{in} and Γ_{out} are the inner and outer boundaries. This somewhat complicated expression for E was chosen to create high values near the inner boundary, moderate values near the far boundary, and low values in an intermediate region. This causes more deformation to occur in the intermediate region, which is desirable. As the figure shows, the resulting mesh still has not inverted, even at a rotation of 120°, but the element quality does become quite poor for the larger rotations.

Because the deformation equations are non-linear, the system may exhibit multiple solutions for a given configuration of the boundary. In particular, the zero that we find is going to be dependent upon the initial approximate zero in the Newton-Raphson procedure. In particular this means that we are able to construct deformed meshes corresponding to $+180°$ and $-180°$ rotations of the inner boundary using essentially a homotopy of intermediate rotations (e.g. $+30°$, $+60°$, ..., $+180°$).

In fact, this gives an easy way to increase the robustness of the solver. We can add a parameter α, varying between 0 and 1, to our non-linear system in eq. (3.20). Here $\alpha = 0$ corresponds to a problem with a known solution and $\alpha = 1$ corresponds to the system that we are interested in solving. For $\alpha = 0$ an obvious choice is the result of a previous non-linear solve, or if none available the undeformed reference configuration. Intermediate values of α correspond to intermediate problems, e.g., a linear interpolation of the positions of the Dirichlet boundaries:

$$\boldsymbol{x}_D(\alpha) = (1.0 - \alpha)\boldsymbol{x}_{D,0} + \alpha\boldsymbol{x}_{D,1} \tag{4.16}$$

where $\boldsymbol{x}_{D,0}$ are the Dirichlet boundary conditions for the known $\alpha = 0$ solution and $\boldsymbol{x}_{D,1}$ are the Dirichlet boundary conditions for the desired $\alpha = 1$ solution.

Algorithm 4.1 Adaptive Newton-Raphson method.

Input: System of equations $\boldsymbol{R}_\alpha(\boldsymbol{X}) = 0$ with known solution $\boldsymbol{X}_0$ for $\alpha = 0$.

```
X ← X_0                                          # Initial configuration.
α ← 0, Δα ← 1                                    # Start at 0, attempt full step.
while α < 1 do
    X_old ← X, α_old ← α                         # Backup old solution.
    α ← min(α + Δα, 1)
    Solve R_α(X) = 0 using Newton's method       # Terminate early if NaNs of Infs.
    if Newton convergence then
        Δα ← 2Δα                                 # Accept step, increase stepsize.
    else
        X ← X_old, α ← α_old                     # Restore solution.
        Δα ← Δα/2                                # Decrease stepsize.
        if Δα < Δα_min then
            Solver failed.
        end if
    end if
end while
```

Output: $\boldsymbol{X}$, a solution to $\boldsymbol{R}_\alpha(\boldsymbol{X}) = 0$ for $\alpha = 1$.

With this formulation any homotopy technique may be used to advance the known solution at $\alpha = 0$ to the desired solution at $\alpha = 1$. In particular we have found it convenient to use an adaptive process where α is slowly increased by $\Delta\alpha$, the increment chosen so that the Newton convergence is well behaved. The details of the algorithm are given in algorithm 4.1. This strategy can certainly be improved upon, but it does contain the essential ingredients of adaptive parameter stepping based upon feedback from the Newton convergence.

One immediate difficulty when using this non-linear elasticity deformation method is the inability to easily compute the deformed mesh mapping velocity $d\boldsymbol{x}/dt$ from the boundary velocity data. Clearly one could use a finite difference approach. For example if we call $\boldsymbol{x}$ the

deformed mesh arising from the boundary data $\boldsymbol{x}_D$, the mapping velocity may be generically approximated to second order accuracy as:

$$\dot{\boldsymbol{x}}(\boldsymbol{x}_D) = \frac{\boldsymbol{x}(\boldsymbol{x}_D + \epsilon\dot{\boldsymbol{x}}_D) - \boldsymbol{x}(\boldsymbol{x}_D - \epsilon\dot{\boldsymbol{x}}_D)}{2\epsilon} + \mathcal{O}(\epsilon^2) \tag{4.17}$$

for small ϵ where we have used a dot to indicate a time derivative d/dt. Higher order approximations could be used if we need more accuracy, but this quickly becomes a cumbersome process.

Instead, we propose using the chosen time integration scheme itself as a means of computing the mesh velocities. Specifically, if we are given the mesh position $\boldsymbol{x}_i$ at each of the stages $i = 1, \ldots, s$ of an implicit Runge-Kutta scheme a_{ij}, we say that the mapping velocities $\dot{\boldsymbol{x}}_i$ are *stage consistent* if

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + \Delta t \sum_{j=1}^{s} a_{ij}\dot{\boldsymbol{x}}_j \quad \text{for } i = 1, \ldots, s \tag{4.18}$$

where $\boldsymbol{x}_0$ is the mesh position at the start of the Runge-Kutta step. In other words, we define the stage consistent mesh velocities to be those velocities which when integrated using the scheme result in the given mesh positions.

In the case when A is of full rank, e.g., a fully implicit or diagonally implicit Runge-Kutta scheme, some algebraic manipulation gives the stage mesh velocities as a linear combination of the mesh positions:

$$\dot{\boldsymbol{x}}_i = \sum_{j=1}^{s} (A^{-1})_{ij} \frac{\boldsymbol{x}_j - \boldsymbol{x}_0}{\Delta t} \quad \text{for i=1,. . . ,s.} \tag{4.19}$$

Note that for a DIRK scheme A and A^{-1} are lower triangular so the mesh velocity at stage j depends only on stages $i = 1, \ldots, j$. This preserves an obvious time dependency and may be desirable if the stage mesh position is calculated on-the-fly from the current stage variables.

In the case of an ESDIRK scheme A is of rank $s - 1$ and the situation becomes more complicated. Here it is natural to require that a mesh velocity be given at the beginning of the time step, $\dot{\boldsymbol{x}}_0$. Since $c_1 = 0$, to avoid a multi-valued mesh velocity we set $\dot{\boldsymbol{x}}_1 = \dot{\boldsymbol{x}}_0$. This then closes eq. (4.19) and allows us to solve for all of the later stage velocities $\dot{\boldsymbol{x}}_j$. All of the ESDIRK schemes we consider have the first same as last property, so we take the mesh velocity at the final stage $\dot{\boldsymbol{x}}_s$ of one time step as the initial mesh velocity of the subsequent time step.

4.2 Temporal Integrator

Consider for the moment our system of fluid $\boldsymbol{u}^f$ and structure $\boldsymbol{u}^s$ variables written as a coupled first order system of ordinary differential equations $\boldsymbol{M}\, d\boldsymbol{u}/dt = \boldsymbol{r}(\boldsymbol{u})$ where

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}^f \\ \boldsymbol{u}^s \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}^f(\boldsymbol{u}^f;\, \boldsymbol{x}(\boldsymbol{u}^s)) \\ \boldsymbol{r}^s(\boldsymbol{u}^s;\, \boldsymbol{t}(\boldsymbol{u}^f)) \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}^f & \\ & \boldsymbol{M}^s \end{bmatrix}. \tag{4.20}$$

Note that we have highlighted the dependence of the fluid on the structure arising from the ALE mesh motion $\boldsymbol{x}$, and the structure on the fluid via the surface traction $\boldsymbol{t}$.

In addition, observe that the discretized structure equation may be separated into two terms

$$\boldsymbol{r}^s(\boldsymbol{u}^s;\, \boldsymbol{t}(\boldsymbol{u}^f)) = \boldsymbol{r}^{ss}(\boldsymbol{u}^s) + \boldsymbol{r}^{sf}(\boldsymbol{t}(\boldsymbol{u}^f)) \tag{4.21}$$

where the first gives the structure dynamics in the absence of an applied surface traction and the second accounts for the additional dynamics from the applied surface traction. Since the second term is linear in $\boldsymbol{t}$, if $\tilde{\boldsymbol{t}}$ is any other surface traction, we may write the structure equation as

$$\boldsymbol{r}^s(\boldsymbol{u}^s;\, \boldsymbol{t}(\boldsymbol{u}^f)) = \boldsymbol{r}^s(\boldsymbol{u}^s;\, \tilde{\boldsymbol{t}}) + \boldsymbol{r}^{sf}(\boldsymbol{t}(\boldsymbol{u}^f) - \tilde{\boldsymbol{t}}) \tag{4.22}$$

Here we will generally think of $\tilde{\boldsymbol{t}}$ as a predicted value of $\boldsymbol{t}(\boldsymbol{u}^f)$ and refer to it as a predicted fluid-to-structure coupling.

Using this formulation, we may split eq. (4.20) as

$$\boldsymbol{M}\frac{d\boldsymbol{u}}{dt} = \begin{bmatrix} \boldsymbol{r}^f(\boldsymbol{u}^f;\, \boldsymbol{x}(\boldsymbol{u}^s)) \\ \boldsymbol{r}^s(\boldsymbol{u}^s;\, \tilde{\boldsymbol{t}}) \end{bmatrix} + \begin{bmatrix} \\ \boldsymbol{r}^{sf}(\boldsymbol{t}(\boldsymbol{u}^f) - \tilde{\boldsymbol{t}}) \end{bmatrix} \tag{4.23}$$

where we intend to integrate the first term implicitly and the second term explicitly. Observe here how the predicted fluid-structure coupling allows us to complete the implicit solve in two phases, first calling a structure solver to compute the stage value of $\boldsymbol{u}^s$ and then calling a fluid solver to compute the stage value of $\boldsymbol{u}^f$.

Our scheme differs slightly from the standard IMEX formulation in that we avoid evaluating the explicit terms $\boldsymbol{r}^{sf}$ but instead update the stage flux for the structure equation using the corrected value of the coupling $\boldsymbol{t}(\boldsymbol{u}^f)$. Note that this naturally allows per stage Gauss-Seidel iterations where we treat the corrected fluid-to-structure coupling as the new predicted value and repeat the stage calculations. It has been reported that using one or more Gauss-Seidel iterations can improve the stability of the overall method [73], but we emphasize that these iterations are generally not required to achieve the design accuracy of the method.

Here we use the predictor suggested in ref. [75], namely the predicted value $\tilde{\boldsymbol{t}}$ at stage i is a linear combination of the corrected values $\boldsymbol{t}$ at previous stages

$$\tilde{\boldsymbol{t}}_i = \sum_{j=1}^{i-1} \frac{\hat{a}_{ij} - a_{ij}}{a_{ii}} \boldsymbol{t}_j \tag{4.24}$$

where a_{ij} (resp. $\hat{a}_{ij}$) are the coefficients from the implicit (resp. explicit) Runge-Kutta integration scheme. This predictor is easily derived by assuming a linear ODE $du/dt = Au$ which we split into parts intended to be integrated implicitly and explicitly:

$$\frac{du}{dt} = \underbrace{(A - C)u}_{\text{implicit}} + \underbrace{Cu}_{\text{explicit}}. \tag{4.25}$$

The stage solution at stage i is

$$u^{(i)} = u^{(0)} + \Delta t \sum_{j=1}^{i} a_{ij} k_j + \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij} \hat{k}_j \tag{4.26}$$

$$= u^{(0)} + \Delta t \sum_{j=1}^{i} a_{ij} (A - C) u^{(j)} + \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij} C u^{(j)} \tag{4.27}$$

$$= \underbrace{u^{(0)} + \Delta t \sum_{j=1}^{i} a_{ij} A u^{(j)}}_{\text{implicit scheme}} + \Delta t \left(-C a_{ii} u^{(i)} + \sum_{j=1}^{i-1} (\hat{a}_{ij} - a_{ij}) C u^{(j)} \right) \tag{4.28}$$

which recognize as being the implicit scheme plus some additional terms. Here the additional terms inspire the choice of a predictor:

$$\widetilde{Cu^{(i)}} = \sum_{j=1}^{i-1} \frac{\hat{a}_{ij} - a_{ij}}{a_{ii}} C u^{(j)}. \tag{4.29}$$

To see how this actually is a predictor we solve for the stage solution and substitute to see

$$(I - \Delta t a_{ii} (A - C)) u^{(i)} = u^{(0)} + \Delta t \sum_{j=1}^{i-1} a_{ij} (A - C) u^{(j)} + \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij} C u^{(j)} \tag{4.30}$$

$$= u^{(0)} + \Delta t \sum_{j=1}^{i-1} a_{ij} A u^{(j)} + \Delta t a_{ii} \widetilde{Cu^{(i)}} \tag{4.31}$$

where $\widetilde{Cu^{(i)}}$ now is obviously a predicted coupling.

To better understand the coupling we now suppose our system consists of both fluid variables and structure variables

$$A = \begin{bmatrix} A_{ff} & A_{fs} \\ A_{sf} & A_{ss} \end{bmatrix} \tag{4.32}$$

with the coupling C chosen to be

$$C = \begin{bmatrix} 0 & 0 \\ A_{sf} & 0 \end{bmatrix}. \tag{4.33}$$

Then the matrix $I - \Delta t a_{ii} (A - C)$ is block upper-triangular and can be backsolved by first doing a structure solve and then a fluid solve.

If the system is non-linear, as is the case with our fluid and structure models, the algorithm and analysis becomes more tedious but the ideas remain the same. In general we use

Algorithm 4.2 Time integration scheme for the coupled fluid-structure system.

Input: Structure $\boldsymbol{u}^s_n$ and fluid $\boldsymbol{u}^f_n$ values at time t_n.
Input: Timestep $\Delta t > 0$.
Input: Paired implicit (a_{ij}, b_i, c_i) and explicit ($\hat{a}_{ij}$, $\hat{b}_i$, $\hat{c}_i$) Runge-Kutta coefficients.

$\boldsymbol{t}_{n,1} \leftarrow \boldsymbol{t}(\boldsymbol{u}^f_{n,1})$ # Fluid-to-structure coupling.
$\boldsymbol{k}^s_{n,1} \leftarrow M_s^{-1}\boldsymbol{r}^s(\boldsymbol{u}^s_{n,1}, \boldsymbol{t}_{n,1})$ # Structure residual.
$\boldsymbol{k}^f_{n,1} \leftarrow M_f^{-1}\boldsymbol{r}^f(\boldsymbol{u}^f_{n,1}, \boldsymbol{x}(\boldsymbol{u}^s_{n,1}))$ # Fluid residual.
for stage $i = 2, \ldots, s$ **do**
  $\tilde{\boldsymbol{t}}_{n,i} \leftarrow \sum_{j=1}^{i-1} \frac{\hat{a}_{ij} - a_{ij}}{a_{ii}} \boldsymbol{t}_{n,j}$ # Predict fluid-to-structure coupling.
  Solve for $\boldsymbol{k}^s_{n,i}$ in $M_s\boldsymbol{k}^s_{n,i} = \boldsymbol{r}^s(\boldsymbol{u}^s_{n,i}, \tilde{\boldsymbol{t}}_{n,i})$
    where $\boldsymbol{u}^s_{n,i} = \boldsymbol{u}^s_n + \Delta t \sum_{j=1}^{i} a_{ij}\boldsymbol{k}^s_{n,j}$ # Implicit structure solve.
  Solve for $\boldsymbol{k}^f_{n,i}$ in $M_f\boldsymbol{k}^f_{n,i} = \boldsymbol{r}^f(\boldsymbol{u}^f_{n,i}, \boldsymbol{x}(\boldsymbol{u}^s_{n,i}))$
    where $\boldsymbol{u}^f_{n,i} = \boldsymbol{u}^f_n + \Delta t \sum_{j=1}^{i} a_{ij}\boldsymbol{k}^f_{n,j}$ # Implicit fluid solve.
  $\boldsymbol{t}_{n,i} \leftarrow \boldsymbol{t}(\boldsymbol{u}^f_{n,i})$ # Correct fluid-to-structure coupling.
  $\boldsymbol{k}^s_{n,i} \leftarrow M_s^{-1}\boldsymbol{r}^s(\boldsymbol{u}^s_{n,i}, \boldsymbol{t}_{n,i})$ # Re-evaluate structure residual.
end for
$\boldsymbol{u}^s_{n+1} \leftarrow \boldsymbol{u}^s_n + \Delta t \sum_{i=1}^{s} b_i\boldsymbol{k}^s_{n,i}$ # Advance structure.
$\boldsymbol{u}^f_{n+1} \leftarrow \boldsymbol{u}^f_n + \Delta t \sum_{i=1}^{s} b_i\boldsymbol{k}^f_{n,i}$ # Advance fluid.

Output: Structure $\boldsymbol{u}^s_{n+1}$ and fluid $\boldsymbol{u}^f_{n+1}$ values at time $t_{n+1} = t_n + \Delta t$.

the same formula for the predictor (since that term enters the structure equation linearly) but replace the linear structure and fluid solves with their non-linear counterparts.

At a minimum it is easy to see that the algorithm, as presented in algorithm 4.2, is first order accurate by linearizing around the initial conditions and noting that it becomes equivalent to the linear algorithm just discussed. Numerical experiments, however, indicate that the design order is achieved.

The algorithm easily lends itself to subiterations, repeating the structure and fluid solvers. However this is quite expensive and not required to achieve the design accuracy [74]. There have been reports of some increased stability when using subiterations [73] and we did experience this to some extent. However, we found the method was generally stable in the regime of time accurate coupling.

4.3 Validation

4.3.1 ALE / Expanding Pressure Wave

We begin by testing the Arbitrary Lagrangian-Eulerian formulation with a specified analytic mesh deformation, and compare spatial convergence for several deformation strategies. As

a non-trivial test problem, we consider a small Gaussian perturbation in the density and pressure of an otherwise constant state.

As the domain we choose $\Omega = [0,1]^2$ with far-field boundary conditions on the left, bottom, and right walls and an adiabatic no-slip condition on the top wall. The spatially varying fluid density, momentum and pressure are initialized as

$$\rho = \rho_\infty(1 + d_0 \exp(\|x - x_0\|_2^2/r_0^2) \tag{4.34}$$

$$\rho\vec{u} = 0 \tag{4.35}$$

$$p = p_\infty(1 + d_0 \exp(\|x - x_0\|_2^2/r_0^2)) \tag{4.36}$$

with non-dimensionalized far-field density $\rho_\infty = 1$. The far-field pressure p_∞ is calculated from eq. (2.10) using the non-dimensionalized sound speed $a_\infty = 5$. The perturbation parameters where chosen as $d_0 = 0.1$, $r_0 = 0.1$, and $x_0 = [0.5, 0.7]^T$.

The fluid is modeled using the compressible Navier-Stokes equations, eqs. (2.1) to (2.3), with dynamic viscosity $\mu = 1/1000$. The background mesh was optionally deformed using an analytic mapping

$$x(X, Y, t) = X + A\sin(2\pi X)\sin(2\pi Y)\sin(2\pi f t) \tag{4.37}$$

$$y(X, Y, t) = Y + A\sin(2\pi X)\sin(2\pi Y)\sin(4\pi f t) \tag{4.38}$$

with amplitude $A = 0.05$ and frequency $f = 20$.

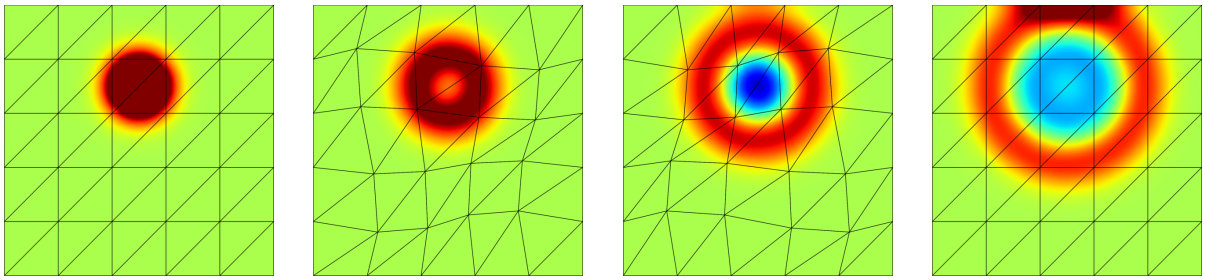
The system was integrated until a final time of $T = 1/20$ which is sufficient time for the pressure wave to propagate and hit the adiabatic no-slip wall on the top of the domain. We used the explicit RK4 scheme with a sufficiently small Δt so that the error in the solution at time T was dominated by errors in the spatial discretization and not the temporal discretization.

Note that T is the period of mesh deformation, so that the mesh starts in an undeformed configuration at time $t = 0$ and returns to an undeformed configuration at time $t = T$. This allows us to measure the accuracy of the ALE mapping by comparing the numerical solution of the problem at $t = T$ to one obtained on a non-deforming mesh.

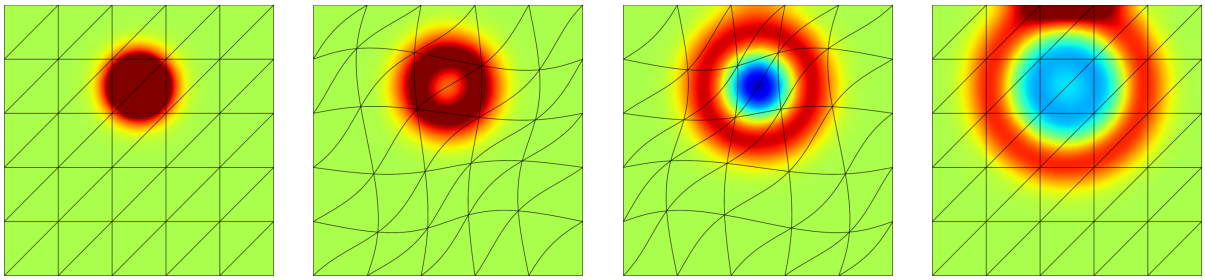
The domain Ω is discretized using triangular elements into a mesh $\mathcal{T}_h$ by first dividing the domain into a regular grid of squares of side length h and then dividing each of those squares in half along the diagonal.

The solution on each element is represented using polynomials of degree p where we investigate $p = 1$ through 5. Numerically the mesh deformation is represented on each element using either a linear $p = 1$ representation or an isoparametric representation. A time series of the solution on two meshes is shown in fig. 4.4.

Here we observe that both deformation strategies are easily able to accurately capture the radiating pressure wave. Notice that when we represent the mesh deformation using $p = 1$ elements the resulting map $\boldsymbol{x}(\boldsymbol{X}, t)$ is piecewise linear and hence the ALE formulation in section 2.2 simplifies significantly as the deformation gradient $\boldsymbol{G}$ and mapping determinant g are both constant. This also simplifies calculation of the viscous derivative as an entire term $\nabla_{\boldsymbol{X}}(g^{-1})$ vanishes.



(a) $t = 0$ (b) $t = T/3$ (c) $t = 2T/3$ (d) $t = T$



(e) $t = 0$ (f) $t = T/3$ (g) $t = 2T/3$ (h) $t = T$

Figure 4.4: An expanding pressure wave on a deforming mesh using a linear deformation (top) and an isoparametric deformation (bottom). In each case the solution was represented using elements of polynomial degree 5. (Pressure).

However, a $p = 1$ mesh deformation representation is unlikely to be able to accurately capture complicated boundary motions. On the other hand, the isoparametric $p = 5$ mesh deformation representation can much more accurately capture complicated motions but is more computationally expensive. A mixed approach is possible, where high order deformations are used on elements near the boundaries and low order deformations are used elsewhere, but this is not explored in this work.

The relative accuracy of using a $p = 1$ deformation instead of an isoparametric can be discussed. Here we naïvely expect the $p = 1$ mapping to produce slightly better results since the resulting map from real coordinates $\boldsymbol{x}$ to the solution $\boldsymbol{u}$, which requires inverting the mesh deformation, is slightly less complicated. This intuition is reflected in a numerical convergence plot which is shown in fig. 4.5.

Here we measure the error in the solution at $t = T$ in the discrete maximum norm for a non-deforming fixed mesh, a $p = 1$ deformation, and an isoparametric deformation ('Full P') for elements of order $p = 1$ though 5. In general we observe convergence at the expected $p + 1$ rate for the fixed mesh and both deformation strategies. For small p the difference in accuracy between the three methods is difficult to ascertain, however for larger p the difference is more acute. At $p = 5$ there is a notable difference in accuracy between the three methods, with the fixed mesh being the most accurate and the isoparametric deformation being the least accurate.

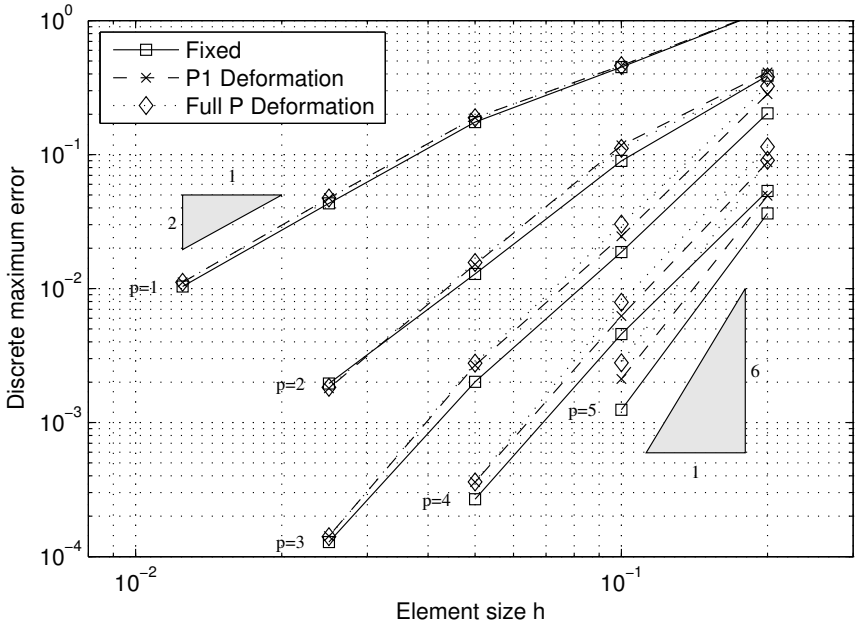


Figure 4.5: Spatial convergence in the discrete maximum error at the final simulation time for an expanding pressure wave for meshes of polynomial degree $p = 1$ to 5. The deformation is either linear (P1) or isoparametric (Full P).

From this experiment we can generally recommend using a linear representation of the mesh deformation if possible. If not, the isoparametric deformation still performs adequately and is able to represent a much larger class of deformations. Mixed approaches should be feasible and represent a possible compromise.

4.3.2 FSI / Pitching and Heaving Airfoil

To validate the high-order convergence of the integration scheme in time, we considered a simple test problem consisting of a pitching and heaving NACA 0012 airfoil. The airfoil is allowed to rotate around a fixed pivot in the interior of the airfoil, as shown in fig. 4.6. Since the airfoil is treated as a rigid body, the structure variables consist only of the pitching angle θ and the angular velocity ω. The fluid is assigned no-slip boundary conditions on the interface with the airfoil, which contributes a torque τ about the pivot of the airfoil.

The position of the pivot follows a prescribed vertical motion $y(t)$ between $t = 0$ and $t = 1$ which is C^3 and satisfies $y(0) = 0$ and $y(1) = 1/4$. In addition the airfoil is subjected to a torsional restoring force with torsional spring constant k. The equations of motion of

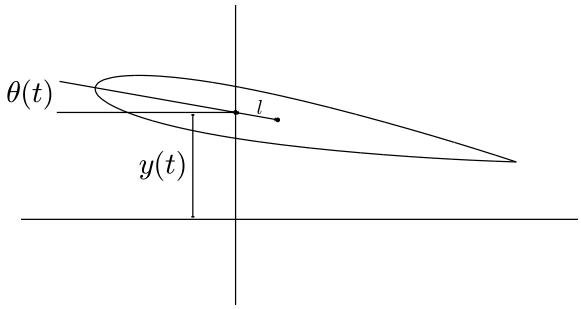


Figure 4.6: Schematic for the pitching and heaving airfoil.

the airfoil written as a first order system are

$$\frac{\partial \theta}{\partial t} = \omega \tag{4.39}$$

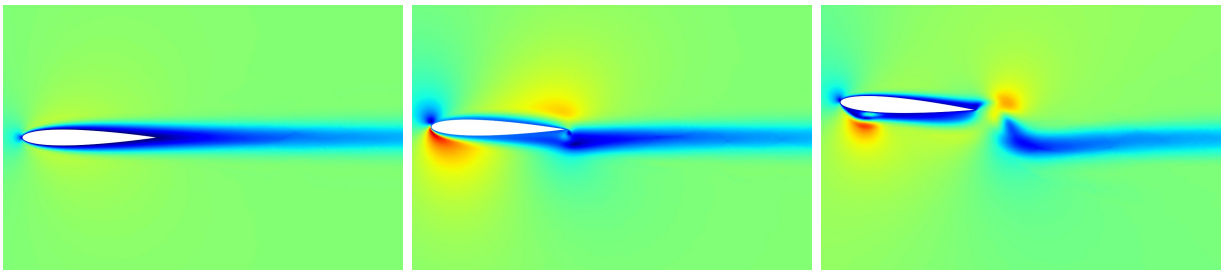
$$I\frac{\partial \omega}{\partial t} = -k\theta - \tau - lm\cos(\theta)y''(t) \tag{4.40}$$

where I is the moment of inertia (around the pivot), l is the distance from the pivot to the center of mass, and m is the total mass of the airfoil.

The non-dimensionalized constants chosen for this problem were $I = 1$, $k = 0.1$, $l = 0.2$, and $M = 1$. The airfoil has chord length 1, and the pivot located along the midline a distance 1/3 from the leading edge. The far field fluid has velocity $\boldsymbol{u} = [1, 0]^T$, density 1, Mach number 0.2, and Reynolds number 1000. To initialize the system at time $t = 0$ we let the structure be at rest and solve for the steady state solution of the fluid. The evolution of the system is shown in fig. 4.7.

To validate the temporal convergence of the scheme we measured the relative error in the angle of attack $\theta(t)$ as compared to the solution of the same system using an explicit fourth order Runge-Kutta method with a suitably small timestep. A plot of the observed relative error as a function of timestep for the third, fourth, and fifth order ARK coefficients is shown in fig. 4.8. Note that in each case, the scheme exhibits convergence at the designed rate. For comparison we also solved the fully-coupled (monolithic) fluid-structure system using the implicit coefficients of the ARK scheme, by performing many Gauss-Seidel subiterations until achieving numerical convergence. This resulted in a negligible increase in accuracy despite a tremendous increase in computational cost.

We emphasize that a naïve staggered method would generally achieve first, or at most second, order accuracy in time. Here, we are able to achieve up to 5th order accuracy in time while reusing the same fluid and and structure solvers that would be used in a staggered scheme.



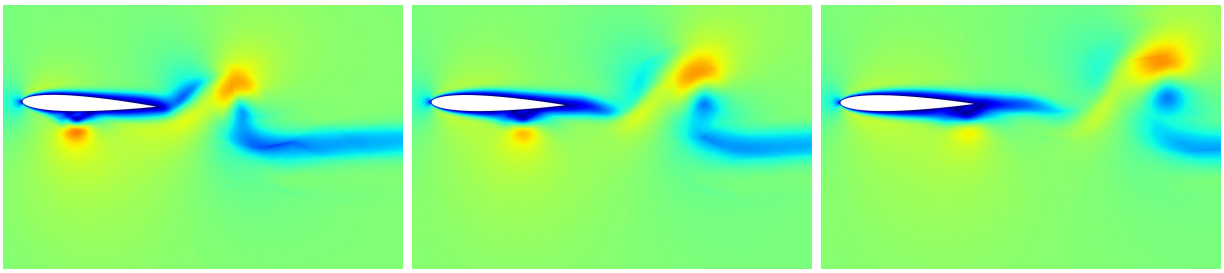


Figure 4.7: The airfoil at various times. The pivot location of the airfoil is smoothly moved upwards between time $t = 0$ and $t = 1$ and held fixed between $t = 1$ and $t = 2$. (Mach number).

4.3.3 FSI / Cantilever

Next we consider a variation on a standard fluid-structure interaction benchmark [68], which consists of a flexible cantilever behind a rigid square body as shown in fig. 4.9.

The cantilever and square body are assigned no-slip boundary conditions. We impose a far field boundary condition at the far walls, with the far field conditions corresponding to uniform flow to the right at 51.3 cm/s. To approximate incompressible flow, we assign a far field Mach number of 0.2. The flow has Reynolds number Re = 333 based on the dimension of the bluff body (1 cm).

We model the cantilever using the neo-Hookean formulation as described in section 2.4, instead of the St. Venant-Kirchhoff model as the test problem describes. We are careful to assign the same elastic moduli, which are specified as Young's modulus $E = 2.5 \times 10^5$ Pa and Poisson's ratio $\nu = 0.35$. The shear and bulk moduli are then calculated as $\mu = E/(2(1+\nu))$ and $\kappa = E/(3(1-2\nu))$.

The fluid domain was triangulated using 6576 degree 3 elements, for a total of 65,760 high-order nodes. The cantilever was triangulated using 64 degree 3 elements. The system was integrated in time using the ARK3 coefficients and a fixed time step of 1×10^{-3} s. One Gauss-Seidel iteration was performed at each integration stage to increase the stability of the coupling.

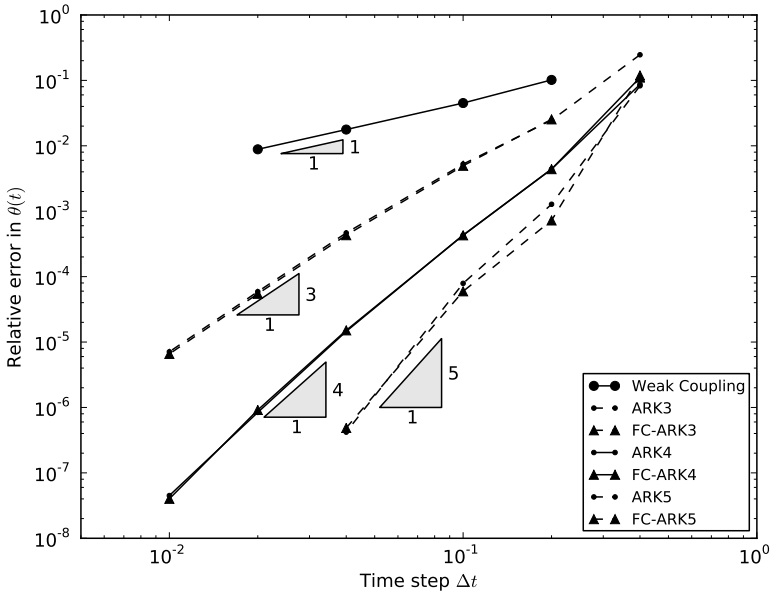
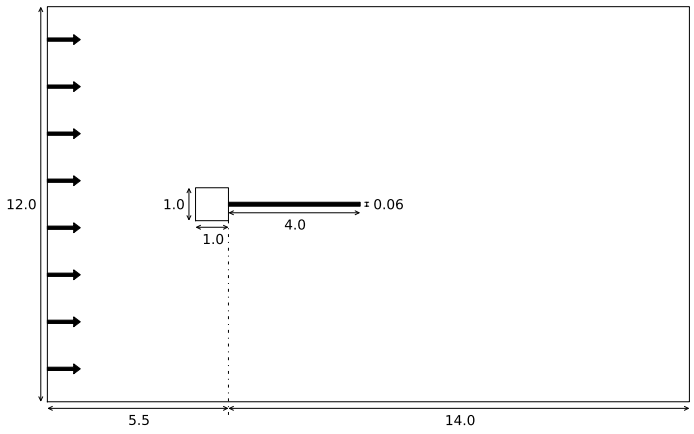


Figure 4.8: The relative error in angle of attack, $\|\theta(t) - \theta_{exact}(t)\|_\infty / \|\theta_{exact}(t)\|_\infty$, as a function of timestep. The ARK3, ARK4, and ARK5 schemes achieve the expected order of accuracy. Solving the fully-coupled ("FC-") system using the implicit method from the IMEX scheme shows a negligible increase in accuracy despite a large increase in computational cost. A basic staggered weak coupling scheme is shown for comparison.



Cantilever
$\rho_s = 100\,\mathrm{kg/m^3}$
$\nu_s = 0.35$
$E = 2.5 \times 10^5\,\mathrm{Pa}$

Fluid & Flow
$\rho_f = 1.18\,\mathrm{kg/m^3}$
$\nu_f = 1.54 \times 10^{-5}\,\mathrm{m^2/s}$
$v_f = 0.513\,\mathrm{m/s}$
$\mathrm{Re} = 333$
$\mathrm{Ma} = 0.2$

Figure 4.9: Flexible cantilever behind a rigid square body. All distances shown are in cm.

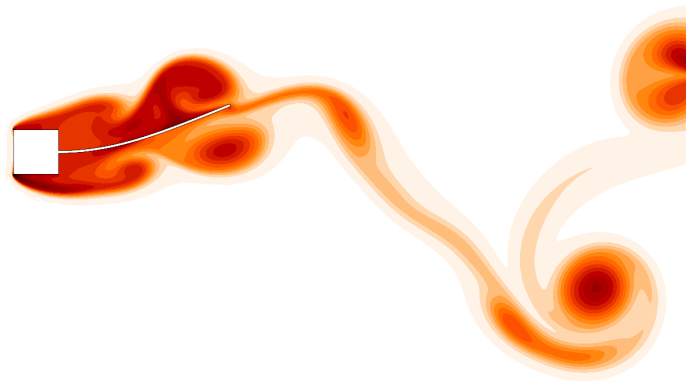


Figure 4.10: The the cantilever near maximal displacement (Entropy).

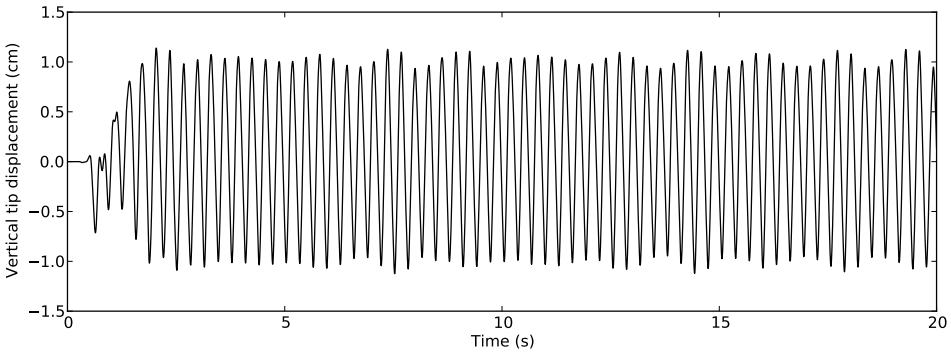


Figure 4.11: The vertical displacement of the cantilever tip as a function of time.

The Reynolds number considered is high enough that the flow separates at the bluff body and produces a von Kármán vortex street. This causes the cantilever to begin oscillating and after a period of a few seconds the fluid-cantilever system settles into a nearly periodic state. Figure 4.11 shows the vertical displacement of the tip as a function of time.

The observed tip vertical amplitude and oscillation frequency are compared to the existing literature in table 4.1. Our observed a maximal tip amplitude of 1.12 cm and oscillation frequency of 3.18 Hz show good agreement with values obtained in the literature, which were computed using different fluid, structure, and temporal discretizations.

Table 4.1: A comparison of the oscillation frequency and maximal vertical tip displacement of the cantilever with values reported in the literature, as reproduced from ref. [34]. The coupling abbreviations stand for partitioned block Gauss-Seidel (P-BGS), partitioned block-Newton (P-BN), and partitioned Newton-Raphson (P-NR).

Author	Fluid	Structure	Coupling	f(Hz)	$d_{\max}$(cm)
Kassiotis et al. [39]	FVM	FEM	P-BGS	2.98	1.05
Wood et al. [69]	FVM	FEM	P-BGS	2.94	1.15
Yvin [71]	FVM	FEM	P-BGS	3.16	1.20
Olivier et al. [46]	FVM	FVM	P-BGS	3.17	0.95
Habchi et al. [34]	FVM	FVM	P-BGS	3.25	1.02
Walhorn et al. [66]	Stabilized FEM	FEM	P-BGS	3.14	1.02
Wall and Ramm [68]	Stabilized FEM	FEM	P-BGS	2.99	1.22
Matthies and Steindorf [45]	FVM	FEM	P-BN	3.13	1.18
Dettmer and Perić [22]	Stabilized FEM	FEM	P-NR	3.03	1.25
Present study	DG FEM	FEM	IMEX	3.18	1.12

Chapter 5

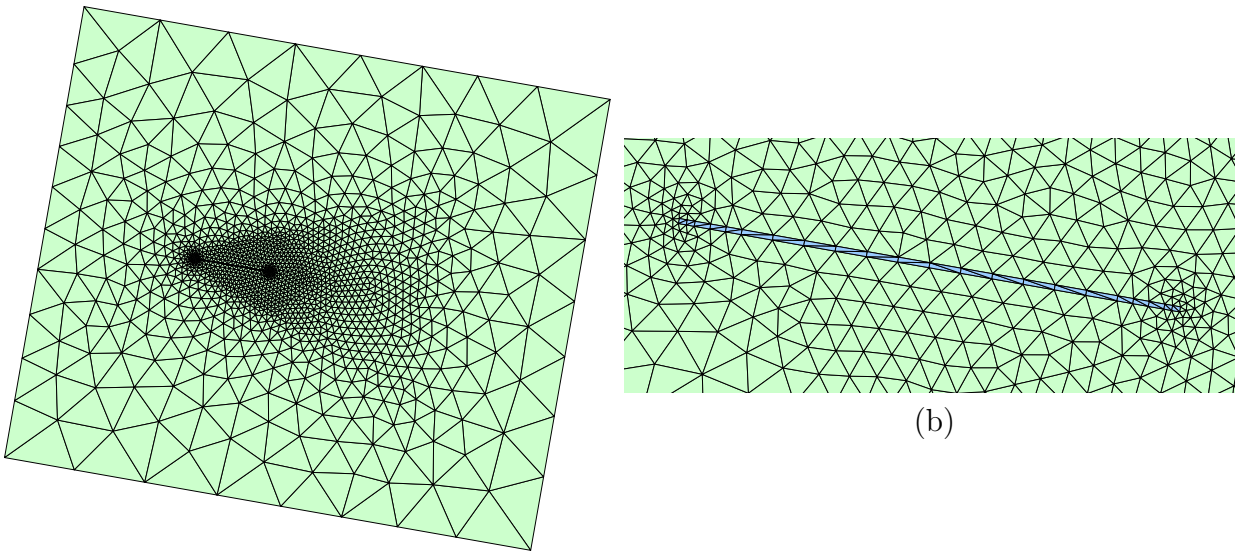
Membranes & Flapping Wings

There has been recent interest in better understanding the mechanics of animal flight. One particularly interesting and little understood regime of animal flight are bats and other similar mammals which have thin compliant membrane-like wings. Here the pattern of flight is quite complicated; bats, for example, are able to articulate numerous joints in the wing. In addition, the compliant skin membrane is thought to play a large role in the resulting aerodynamic properties of the wing. Better understanding the role the membrane wings play animal flight is thought likely to improve the performance of micro air vehicles [60].

Because of the relative difficulty in working with live animals and the inability to isolate the effect of a particular kinematic parameter, considerable effort has been put into other means of studying bat flight including building a robotic replica [8]. In the same vein as ref. [63] we propose using numerical simulations of more canonical geometries (squares and triangles) of membranes whose boundaries are either held fixed or driven in an idealized flapping motion [30].

In this chapter we demonstrate that the techniques in this work are suitable for simulating simple motions of thin membranes at moderate Reynolds number. We begin by investigating a thin two-dimensional membrane whose leading and trailing edges are held fixed to create a constant angle of attack with an incoming fluid. We then extend this work to a fully three-dimensional model where tip vortices create a non-uniform flow in the span-wise direction. In each case we demonstrate that the compliance of the membrane is successful in delaying or eliminating leading edge separation which would be present if the membrane was instead perfectly rigid.

We then extend this work to flapping flight, first in the two-dimensional case where the leading and trailing edges of the membrane are instead driven in a sinusoidal pattern. A similar ability of the leading edge to align with the incident flow is observed. In three dimensions we model a flapping wing as a triangular membrane which is stretched between a fixed body and an oscillating leading edge, with an unconstrained trailing edge. The dihedral angle of the leading edge varies sinusoidally and we observe that the membrane experiences both inertial effects from the flapping motion and aeromechanical effects from the fluid.



(a) (b)

Figure 5.1: (a) The undeformed fluid (green) and structure (blue) meshes for the 2D Membrane at a 10° angle of attack. (b) The region near the structure is enlarged.

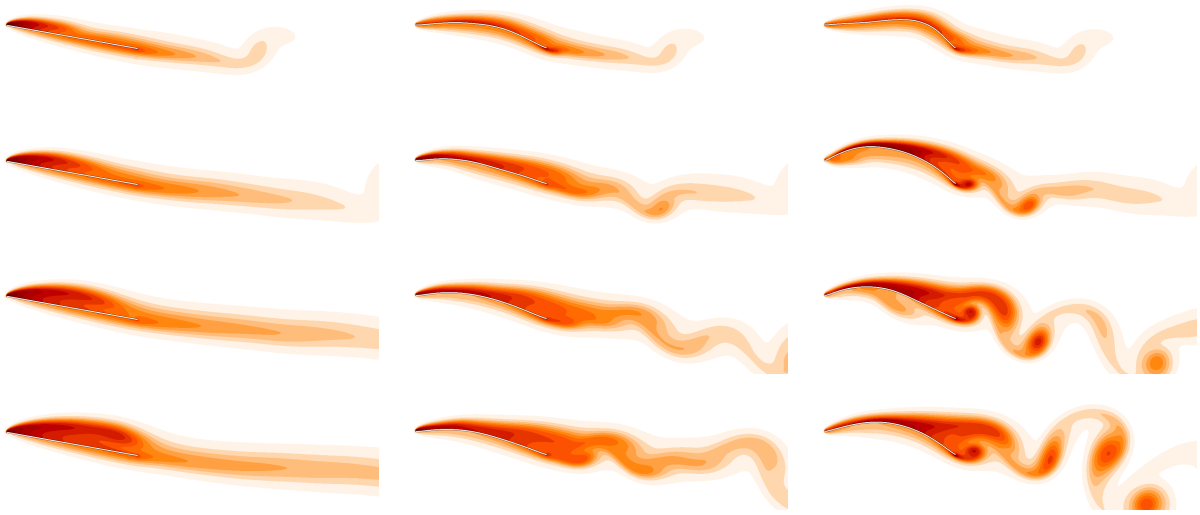
5.1 Membrane, 2D

First we considered a thin rectangular membrane with length 1 and height 0.01 in uniform incoming flow at a 10° angle of attack. This structure was modeled using the standard volumetric equations as described in section 2.4, with highly anisotropic elements. We applied no-slip conditions on the membrane boundary and far field boundary conditions on the far fluid domain boundaries. The far field flow was set to unit density, unit velocity in the x direction, Mach number 0.2, and Reynolds number 1000. We assigned Dirichlet boundary conditions of no displacement to the front and rear faces of the membrane.

The membrane was set to a non-dimensionalized density of $\rho = 40.0$, Poisson's ratio of $\nu = 0.3$. We explored two different Young's moduli of $E = 1 \times 10^3$ and $E = 5 \times 10^3$. For comparison we also investigated a fixed, rigid plate.

The membrane was discretized using 44 degree 3 triangular elements, and the fluid was discretized using 2575 degree 3 triangular elements. See fig. 5.1. The system was integrated in time using the ARK3 coefficients and a timestep of 2×10^{-3}.

The lift and drag coefficients as a function of time for the rigid plate and two membranes are plotted in fig. 5.3. In each case the coefficients were computed using the characteristic planform area, i.e., 1.



Rigid Plate Membrane ($E = 5 \times 10^3$) Membrane ($E = 1 \times 10^3$).

Figure 5.2: The rigid plate and two membranes at time $T = 1.0$ (top), 2.0, 3.0, and 4.0 (bottom). (Entropy).

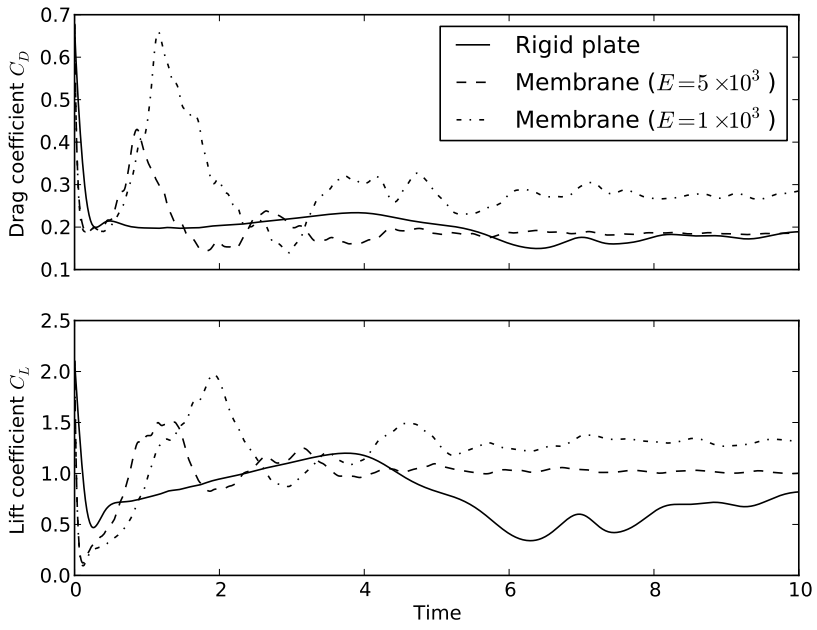


Figure 5.3: Lift and drag coefficients as a function of time for a rigid plate and two flexible membranes at 10° angle of attack.

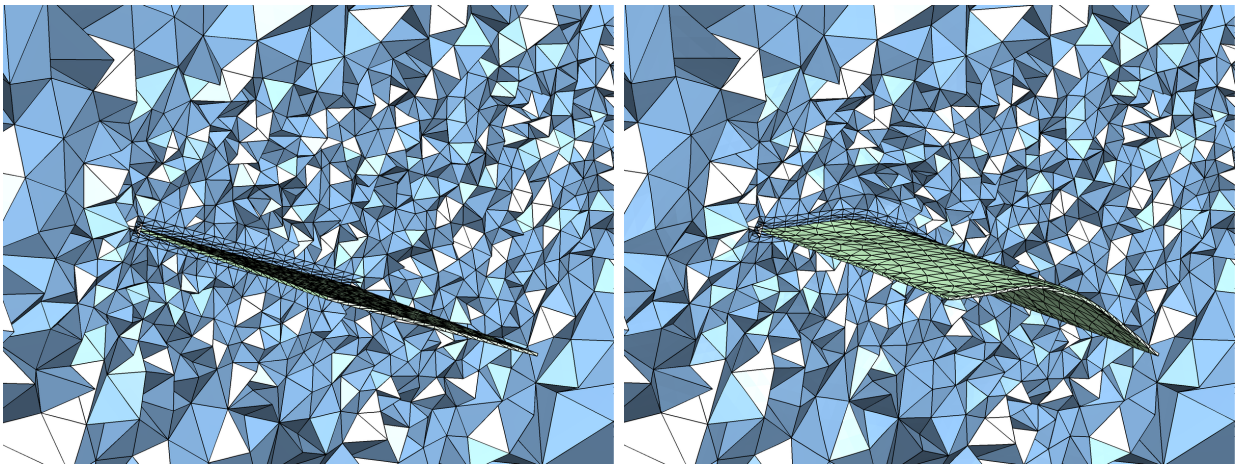


Figure 5.4: A cross section of the fluid mesh (blue) and entire structure mesh (green) in the reference configuration (left) and typical deformed configuration (right).

5.2 Membrane, 3D

In three dimensions we considered an extruded form of the 2D membrane module from section 5.1. The membrane has length and width 1 and height 0.01 and is placed in uniform flow at an atan(5/12) $\approx 22.6°$ angle of attack. The fluid was assigned no-slip conditions at the membrane boundary and far field conditions at the far boundaries. The far field flow was set to unit density $\rho = 1.0$, unit velocity $\boldsymbol{u} = [1.0, 0, 0.]^T$, Mach number 0.2 and Reynolds number 2000. The membrane was assigned Dirichlet boundary conditions on the leading and trailing faces. The physical parameters of the membrane were chosen as density $\rho = 100.0$, Young's modulus $E = 1 \times 10^3$, and Poisson's ratio $\nu = 0.35$.

The membrane was discretized using 1317 highly anisotropic degree 3 elements. The fluid mesh had 108,358 degree 3 elements, for a total of about 2.17 million high-order nodes or almost 11 million degrees of freedom. A cross-section of the fluid mesh is shown in fig. 5.4.

A timestep of 1×10^{-3} was used and the system was solved until $T = 3.0$. The Mach number is shown on iso-entropy surfaces for several time steps in fig. 5.5. Here we see that the leading edge of the membrane aligns with the incoming fluid and successfully prevents leading edge separation. In addition we see that the fluid curls around the sides of the membrane and exhibits a classic roll-up behavior. For comparison, in fig. 5.6 we show the behavior of a fluid when the membrane is replaced by a fixed rigid plate of the same dimensions.

5.3 Flapping Wing, 2D

We model a two-dimensional flapping wing as a thin rectangular structure of chord length 1 and height 0.06 whose endpoints are heaved according to a prescribed vertical motion. The leading ("left") and trailing ("right") edges follow a sinusoidal motion with a phase lag of

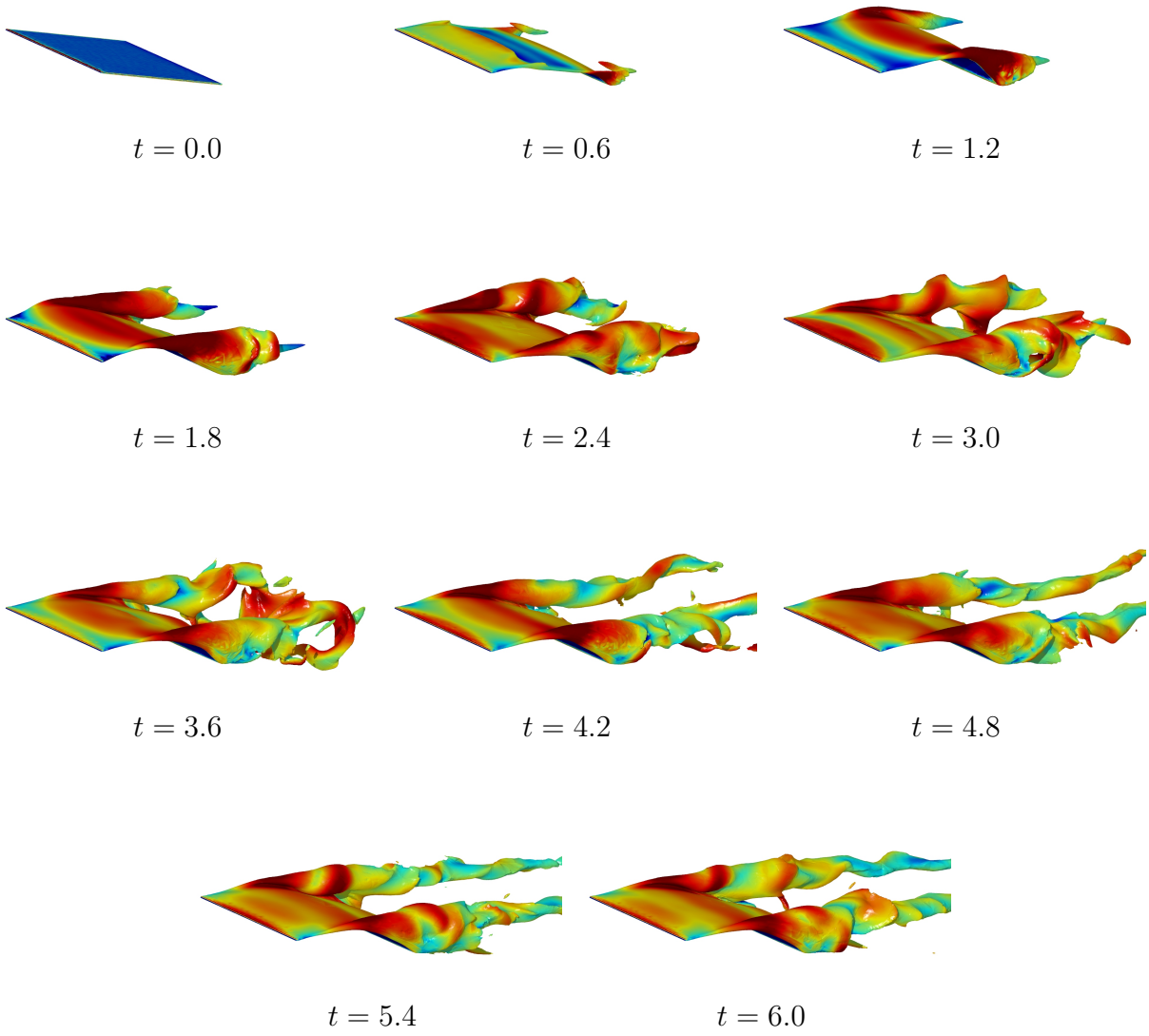


Figure 5.5: A three-dimensional membrane at various times (Mach number on iso-entropy surfaces). The leading edge of the membrane aligns with the fluid and prevents separation.

60°. Specifically,

$$y_l(t) = A\sin(2\pi ft) \tag{5.1}$$
$$y_r(t) = A\sin(2\pi ft - \phi) \tag{5.2}$$

where $A = 0.5$, $f = 0.2$, and $\phi = \pi/3$. A schematic of the model along with the material parameters is shown in fig. 5.7.

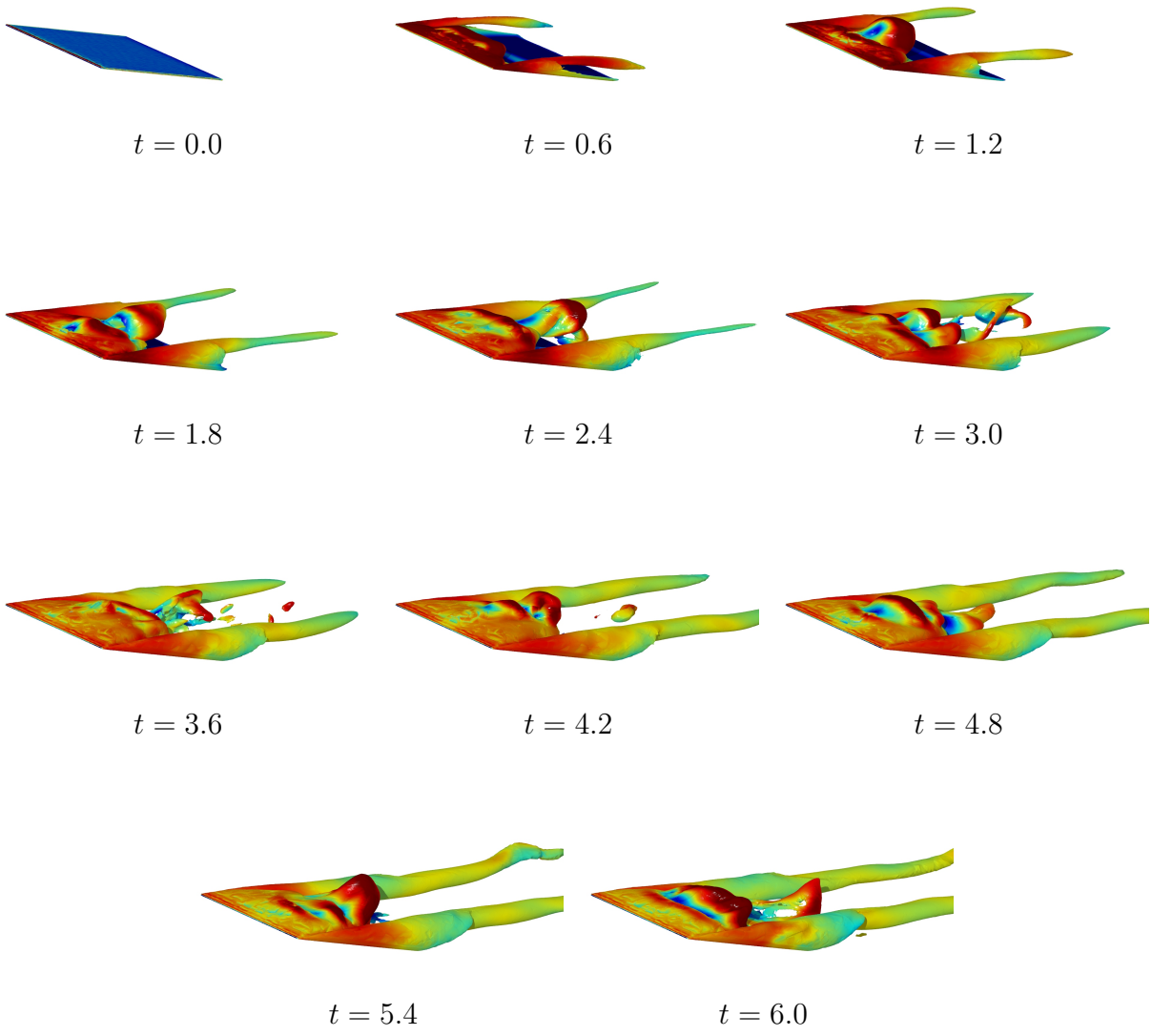
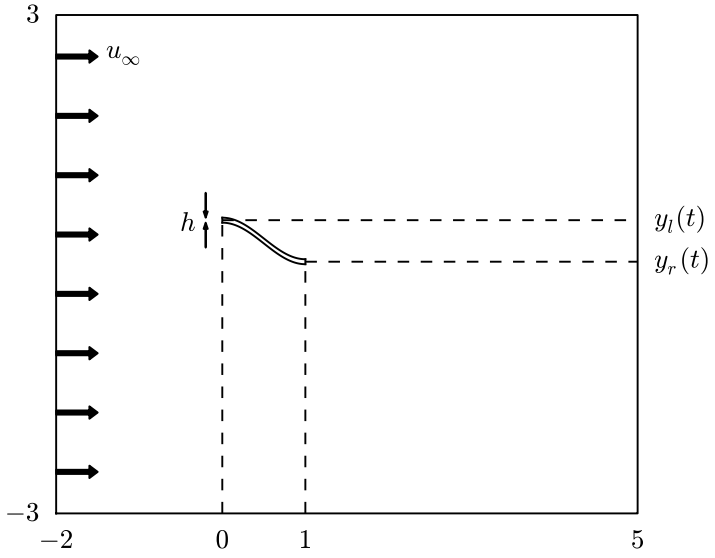


Figure 5.6: A rigid plate at various times (Mach number on iso-entropy surfaces). Note the leading edge separation caused by the high angle of attack.

For consistency in the initialization of the fluid, the structure was forced into a 0° angle of attack at time $t = 0$ by multiplying the y_l- and y_r-coordinates by a smooth C^3 blending function which was 0 at $t = 0$ and 1 at $t = 1/(4f)$. This creates a somewhat violent initial motion and in some cases introduces transient modes in the structure which decay after a few periods of heaving.

The structure was always started with no velocity and in a rectangular configuration with



Structure
$h = 0.06$
$l = 1.0$
$\rho_s = 200$
$\nu_s = 0.40$
$E_s = 1 \times 10^3$ or 5×10^3

Fluid & Flow
$\rho_\infty = 1.0$
$u_\infty = 1.0$
$\mathrm{Ma}_\infty = 0.2$
$\mathrm{Re} = 2000$

Figure 5.7: The two-dimensional wing model in non-dimensionalized coordinates. The left and right endpoints of the structure follow a prescribed motion.

height $h = 0.06$ and length $l = 1$. In some cases the structure was prestretched by an amount Δ by changing the reference dimensions of the structure to $l_{ref} = l/\Delta$ and $h_{ref} = h\Delta^{2\nu}$, where ν is Poisson's ratio. This is not a stationary configuration for the structure, but the initial transients decay quickly and do not cause any difficulties in the simulation.

The domain was triangulated into 44 elements each discretized using polynomials of degree 3 for a total of 268 nodes or 1072 degrees of freedom. The fluid was discretized using 2575 degree 3 polynomial elements for a total of 25,750 nodes or 103,000 degrees of freedom. A timestep of $\Delta t = 5 \times 10^{-3}$ was used to integrate the system until time $T = 20.0$.

We investigated several different material parameters and prestretching factors, and show the results of the simulations at several times in fig. 5.8. Note that the varying parameters greatly impact the ability of the structure to align with the incident flow. Each simulation took approximately 3 hours on 48 CPU cores.

5.4 Flapping Wing, 3D

In three dimensions we consider a thin prismatic triangular structure of root chord length 1, tip-to-tip span length 3, thickness 0.01, and 0° sweep, as shown in fig. 5.9. The leading edge

$E = 5 \times 10^3$	$E = 1 \times 10^3$	$E = 1 \times 10^3$
$\Delta = 1.5$	$\Delta = 1.2$	$\Delta = 1.1$

Figure 5.8: The two-dimensional wing (black) at time $T = 17.0$ (top), 18.0, 19.0, and 20.0 (bottom). The structures have varying Young's modulus E and prestretching factor Δ, and range from nearly rigid (left) to quite flexible (right). (Entropy).

of the wing is driven in a sinusoidal skewing motion in the y-z plane, namely

$$x(t, X, Y, Z) = X \tag{5.3}$$
$$y(t, X, Y, Z) = \cos(\theta(t))Y \tag{5.4}$$
$$z(t, X, Y, Z) = \sin(\theta(t))|Y| + Z \tag{5.5}$$

where the dihedral angle $\theta(t)$ follows a sinusoidal pattern

$$\theta(t) = \theta_0 \cos(2\pi f t), \tag{5.6}$$

with amplitude $\theta_0 = \pi/12$ and frequency $f = 0.2$.

To introduce some additional rigidity into the structure, it was prestretched by a factor

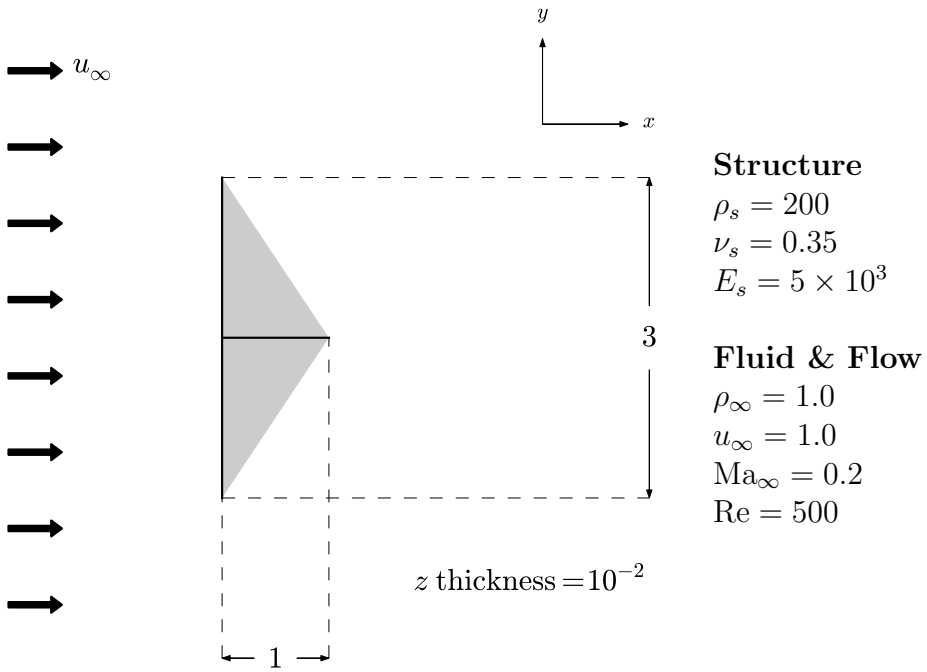


Figure 5.9: The overhead view of the three-dimensional wing which is uniformly extruded in the z direction. Dirichlet conditions are imposed along the planes given by the solid lines. The leading edge is driven with a sinusoidal flapping motion in the y-z plane, and the root chord is held fixed.

Δ by changing the dimensions of the reference structure to

$$\mathrm{span}_{ref} = \mathrm{span}/\Delta \tag{5.7}$$
$$\mathrm{chord}_{ref} = \mathrm{chord}/\Delta \tag{5.8}$$
$$\mathrm{thickness}_{ref} = \mathrm{thickness}/\Delta^{-\nu}. \tag{5.9}$$

We used a prestretching factor $\Delta = 1.05$. To avoid strong initial transients in the structure we began the simulation by first allowing the structure to relax into a steady state configuration at the top of the flapping stroke and in the absence of the fluid.

The structure was discretized using 498 degree 2 tetrahedral elements for a total of 1197 nodes and 7182 degrees of freedom. The fluid was discretized using 22,683 degree 2 tetrahedral elements for a total of 226,830 nodes or 1,134,150 degrees of freedom. A timestep of $\Delta t = 5 \times 10^{-3}$ was used to integrate until the time $T = 5.5$ which corresponds to a little more than one complete stroke.

The simulation was run on 192 CPU cores and took approximately two hours to complete. See fig. 5.10.

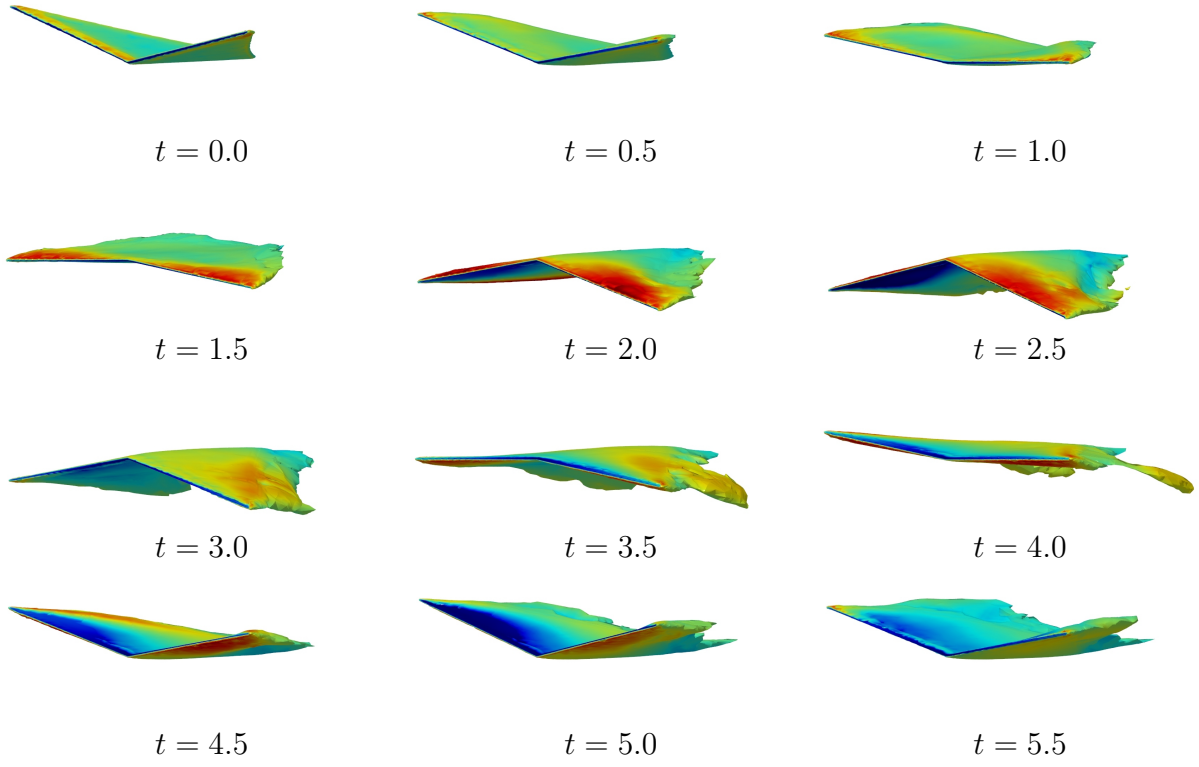


Figure 5.10: A three-dimensional wing at various times (Mach number on iso-entropy surfaces).

Chapter 6

Acoustics

6.1 Tuning Fork

The tuning fork and its properties have long been of interest. Helmholtz, for instance, observed that the sound generation is not directionally uniform near the tuning fork. Instead, in ref. [35] (page 161), Helmholtz observes:

> On striking a tuning-fork and slowly revolving it about its longitudinal axis close to the ear, it will be found that there are four positions in which the tone is heard clearly; and four intermediate positions in which it is inaudible. The four positions of strong sound are those in which either one of the prongs, or one of the side surfaces of the fork, is turned towards the ear.

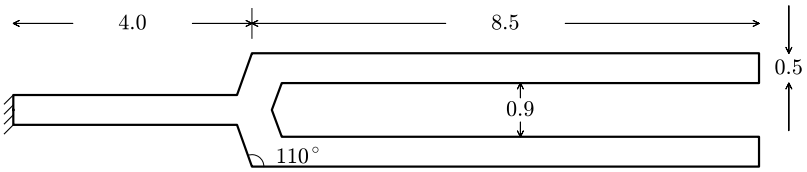
Helmholtz continues to explain that the sound pattern is due to an interference effect between sound generated from each of the tines. Further work refined this observation to posit that the radiated sound field is that of a linear quadrupole, i.e., a sum of two dipole sources of opposite phase whose axes lie on a single line [61], a result which has been generally validated by experimental measurement [59].

This observation shows that modeling a tuning fork fundamentally requires a three-dimensional simulation, as a two-dimensional slice would either fail to capture this directivity pattern or would be unable to properly model the tuning fork itself.

Here we seek to computationally reproduce these measurements of the near-field sound directivity pattern. In addition we study the decay rates for various modes in the tuning fork and demonstrate that a high-order method can naturally capture these rates without any assumptions beyond the standard physical parameters for air and steel. In particular, we observe proper levels of damping without any damping terms in the structure model itself.

6.1.1 The Model

The dimensions of the tuning fork considered are shown in fig. 6.1. The two tines have a square cross section with dimensions 0.5 cm by 0.5 cm and are approximately 8.5 cm long.



$E = 200\,\text{GPa}$
$\nu = 0.29$
$\rho = 7.8\,\text{gm/cm}^3$

Figure 6.1: The dimensions of a cross section of the tuning fork, as well as its material parameters. All distances are given in cm. The tuning fork is extruded 0.5 cm so that the cross-section of the tines are square.

They connect to a stem which is 0.5 cm by 0.5 cm in cross section and 4.0 cm long. The tines are separated by a distance of 0.9 cm. While these dimensions are typical for a tuning fork, it is important to note that this model is not based upon a physical tuning fork and in particular the fundamental mode does not correspond to a standard musical pitch.

The tuning fork is modeled after steel, using the physical parameters density $\rho = 7800\,\text{kg/m}^3$, Young's Modulus $E = 200\,\text{GPa}$, and Poisson's ratio $\nu = 0.29$. In our work we have chosen to hold the tuning fork by rigidly clamping the square face at the base of the stem.

Approximations of a tuning fork using a beam model [58] predict symmetric in-plane modes with frequencies of

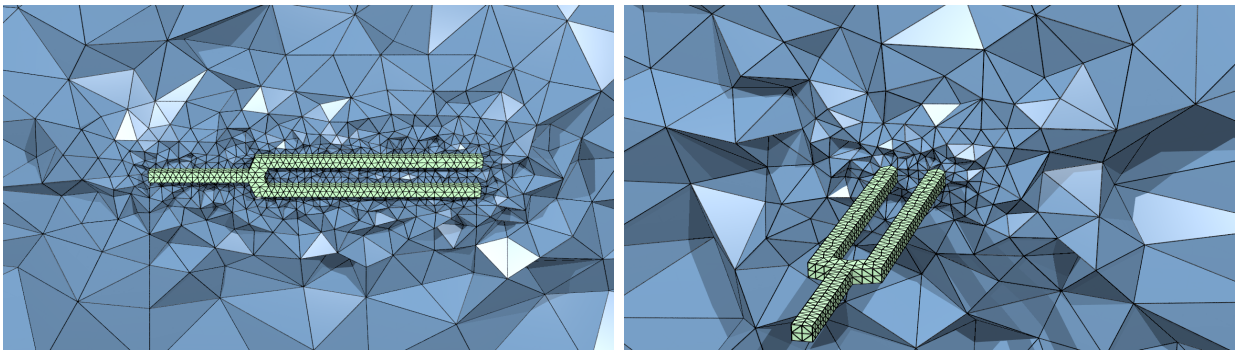
$$f_n = \frac{\pi K}{8L^2}\sqrt{\frac{E}{\rho}}\left[1.194^2,\, 2.988^2,\, 5^2,\, \ldots\, (2n-1)^2\right] \tag{6.1}$$

where L is the length of the tines and K is the radius of gyration ($1/\sqrt{12} \times 0.5\,\text{cm}$ in our case). Using the physical parameters for steel, this gives approximate values of the first two frequencies:

$$f_1 \approx 566.3\,\text{Hz} \quad \text{and} \quad f_2 \approx 3546\,\text{Hz}. \tag{6.2}$$

As is customary, we will refer to the first mode as the *fundamental* or principal mode. This is the dominant mode when the tuning fork is struck and corresponds to the pitch that is heard. In this mode the two tines move in a symmetrical fashion — at any moment either both towards each other or both away from each other. The second mode is called the *clang* mode and corresponds to a symmetric mode where the tips of the tines move towards each other while the middle of the tines move away from each other, and vice versa.

In addition to symmetric in-plane modes, there are also a few other natural classes of modes. Asymmetric in-plane modes are ones where the tines of the tuning fork move in the



(a) Along the tuning fork axis. (b) Perpendicular to the tuning fork axis.

Figure 6.2: The computational mesh for the tuning fork (green) and two different cross sections of the computational mesh for the fluid (blue) in a region near the tuning fork.

same direction. Here it is difficult to have a theoretical formula for the frequencies because the stem also plays a large role in the motion. Out-of-plane modes are ones where the tines of the tuning fork leave the plane, either in a symmetrical or asymmetrical fashion.

Before moving on we finally note that we model the tuning fork as immersed in air. The air is assigned typical values: density $\rho = 1.24\,\mathrm{kg/m^3}$, speed of sound $c = 343.0\,\mathrm{m/s}$, and dynamic viscosity $1.836 \cdot 10^{-5}\mathrm{kg/(m\,s)}$. The simulation domain is a box which extends 10 cm from the tuning fork in each of the Cartesian directions. Note that this domain is almost entirely near-field, as the wavelengths for the two symmetric in-plane modes (eq. (6.2)) are:

$$\lambda_1 = 60.5\,\mathrm{cm} \quad \text{and} \quad \lambda_2 = 9.67\,\mathrm{cm}. \tag{6.3}$$

These lengths are both on the order of, or larger than, the size of the computational domain.

The fluid is assigned no-slip boundary conditions on the interface with the tuning fork and characteristic far-field boundary conditions at the far boundaries. We note that the characteristic boundary conditions are not perfectly absorbing, i.e., waves may be reflected. Future work could include using a perfectly matched layer at the boundary of the box to prevent internal reflections.

6.1.2 Results

Finally, we present results for a single three-dimensional tuning fork simulation. We created two unstructured tetrahedral meshes, one for the fluid and and one for the structure. The structure mesh contained about 2,200 tetrahedra which for our polynomial degree $p = 3$ gives about 13,600 high-order nodes, or 82,000 degrees of freedom. The fluid mesh consisted of approximately 23,200 tetrahedra, which for our polynomial degree $p = 3$ gives 464,000 high-order nodes, or 2,320,000 degrees of freedom (see fig. 6.2).

The tuning fork was initialized by linearly skewing the tines apart from each other so that at the tip the interior spacing increased by 0.014 cm and the exterior spacing increased

by 0.029 cm. We note that this is a highly nonphysical excitation, but was intended to validate the robustness of the solver and ensure that many of the symmetrical modes of the tuning fork would be excited. The tines were then released and the system integrated in time using the algorithm in section 4.2. A timestep of $\Delta t = 50\,\mu$s was used and the system was solved until $T = 30$ ms for a total of 600 time steps. This timestep corresponds to a sampling frequency of 20 kHz allowing us to resolve frequencies below 10 kHz. We note that this timestep is approximately a factor of 20 times above CFL for the fluid based upon the sound speed and the size of the smallest elements.

Because of rather severe initial transients due to the highly deformed configuration of the structure, a timestep of $\Delta t/5$ was used for the first 5 timesteps.

Each time step took approximately one minute on 768 processors, for a total simulation time of approximately 10 hours.

Pressure Time Series

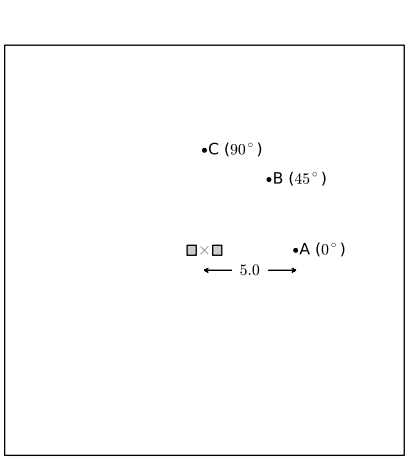
We measured the pressure at several locations surrounding the tuning fork, in each case recording the value relative to the baseline pressure $p_0 = 1.012 \times 10^5$ Pa. In fig. 6.3 we present a time series for the pressure at three locations, each in a plane perpendicular to the axis of the tuning fork intersecting the tuning fork 0.5 cm away from the tips of the tines. The locations shown are all a distance of 5.0 cm from the axis of the tuning fork, making angles of 0°, 45°, and 90° with the axis which passes through both tines. Observe that the high frequency modes decay quickly over the first 10 ms or so, leaving a signal which is almost entirely composed of the principal frequency.

We also present cross sectional visualizations of the pressure at two representative sequences of frames in fig. 6.4. The first sequence, 4.00 ms to 4.10 ms, is before the initial transients have decayed. In this sequence we can see that a high frequency mode, most likely the clang mode, is dominant. In the second sequence, 23.60 ms to 24.00 ms, we see approximately one quarter period of the fundamental mode.

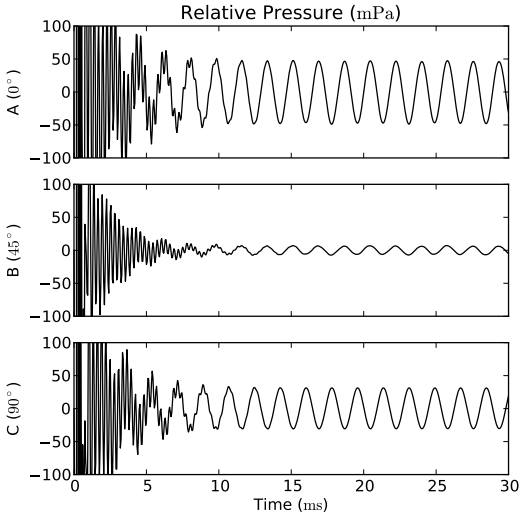
Recall that the sound pressure level L_p, measured in dB above a standard reference level, is calculated as

$$L_p = 10 \log_{10} \left(\frac{{p_{\text{rms}}}^2}{{p_{\text{ref}}}^2} \right), \tag{6.4}$$

where p_{rms} is the root mean square of the signal (relative to the baseline pressure) and p_{ref} is a reference pressure typically set to 2×10^{-5} Pa [2]. By taking a Fourier transform of the last 9 periods of the pressure signal at location A, we show the sound pressure level for various frequency in fig. 6.5. In addition we linearized the tuning fork model around the reference configuration (in the absence of air) and show several computed eigenfrequencies with a description of their corresponding eigenmodes. Due to the comparatively short length of time simulated, the resolution from the Fourier transform is somewhat lacking especially in the low frequency regime. We will return to this point later in section 6.1.2.



(a) Measurement locations.



(b) Time series.

Figure 6.3: Time series data for the relative pressure at three locations each a distance 5.0 cm from the axis of the tuning fork in a plane perpendicular to the axis. The plane is located such that the tines of the tuning fork extend a distance 0.5 cm through the plane. The outer box shows the boundary of computational domain.

Angular Dependence

The directionality of the sound field radiated by a tuning fork may also be measured. The tuning fork is thought to be well modeled by a linear quadrupole, i.e., two dipoles of opposite phase whose dipole axes lie along a single line. A formula for the resulting pressure field is derived as [59, 61]

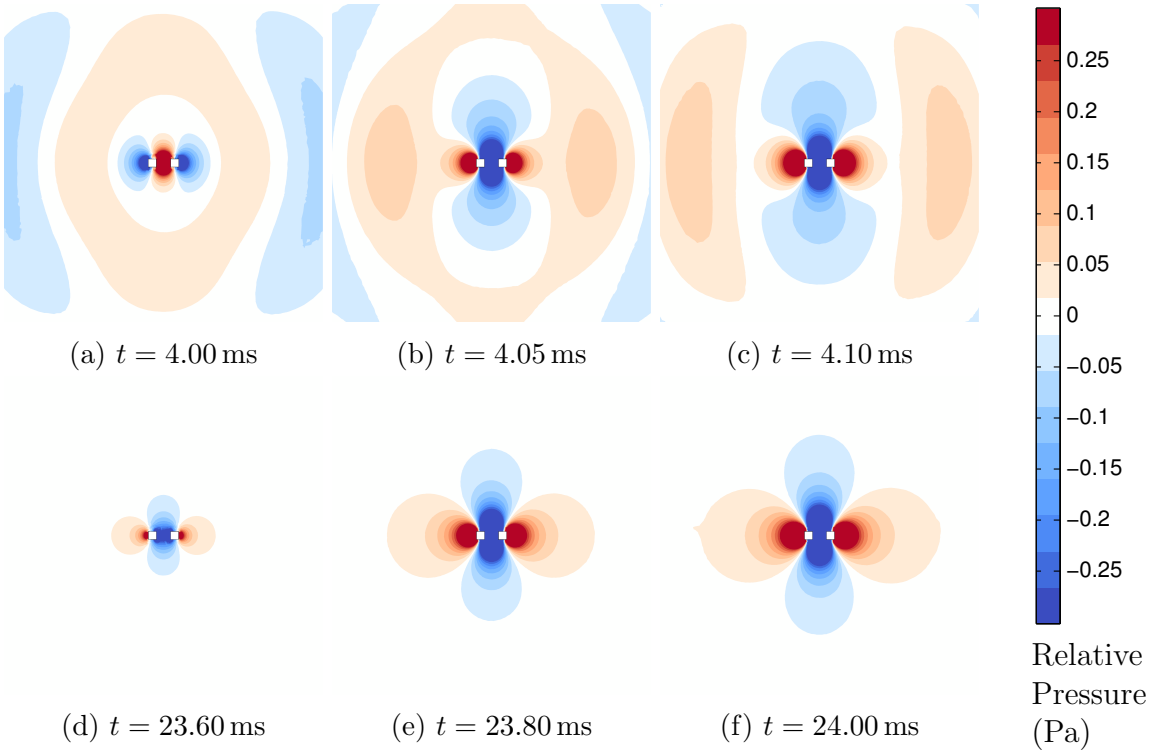
$$p(r,\theta) = \frac{A}{r}\left[(1-3\cos^2\theta)\left(\frac{ik}{r} - \frac{1}{r^2} + \frac{k^2}{3}\right) - \frac{k^2}{3}\right] \tag{6.5}$$

where A is a normalization constant, r is the distance from the linear quadrupole source, $k = 2\pi/\lambda$ is the wave number, and i indicates an out-of-phase term. The product kr is generally used to separate the so-called near-field $kr \ll 1$ from the far-field $kr \gg 1$.

From this formula we compute the angular and radial dependence on the sound pressure level for an idealized linear quadrupole:

$$L_p = 10\log_{10}\left(\frac{\|p(r,\theta)\|^2}{{p_{\text{ref}}}^2}\right). \tag{6.6}$$

In fig. 6.6, we compare this idealized angular dependence to measured sound pressure levels at a variety of distances from the axis of the tuning fork. In each case the measurements were done in the same plane as our previous measurements (see fig. 6.3a). As is typical, we



(a) $t = 4.00\,\text{ms}$ (b) $t = 4.05\,\text{ms}$ (c) $t = 4.10\,\text{ms}$

(d) $t = 23.60\,\text{ms}$ (e) $t = 23.80\,\text{ms}$ (f) $t = 24.00\,\text{ms}$

Figure 6.4: The relative pressure in a plane perpendicular to the axis of the tuning fork (as in fig. 6.3a) at various times. For scale, the figures are 20 cm per side which represents almost all of the computational domain in that plane.

have normalized each plot to the maximum value so that the maximum sound pressure level is shown as 0 dB.

Perhaps the most striking aspect of the directivity plots is the sharp decrease in sound pressure levels between regions of maxima. For instance, we observe an SPL drop of over 40 dB for 4 specific angles when measuring 2.5 cm away from the axis of the tuning fork. Next observe that we accurately capture the expected 5 dB drop in the maximum sound pressure level between the 0°–180° and the 90°–270° axes.

Also notable is the relatively good agreement between the measured sound pressure levels and the linear quadrupole source behavior, especially at the larger radii of 7.5 cm and 10.0 cm. For smaller radii the system is likely no longer well-modeled by an idealized linear quadrupole as the finite size effects of the actual tuning fork are likely to play a larger role. We note that our observed disparity between measurements and the linear quadrupole at 2.5 cm has a similar character to previous experimental measurements (c.f., fig. 9b in ref. [59]) wherein the lobes at 90° and 270° are observed to be wider than those of an idealized linear quadrupole, and the lobes at 0° and 180° are observed to be narrower.

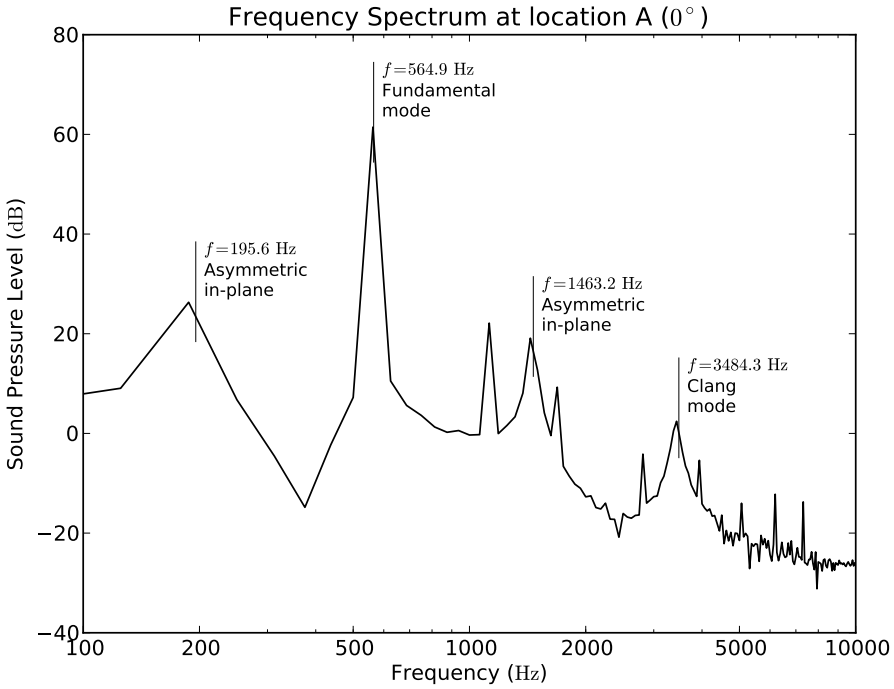


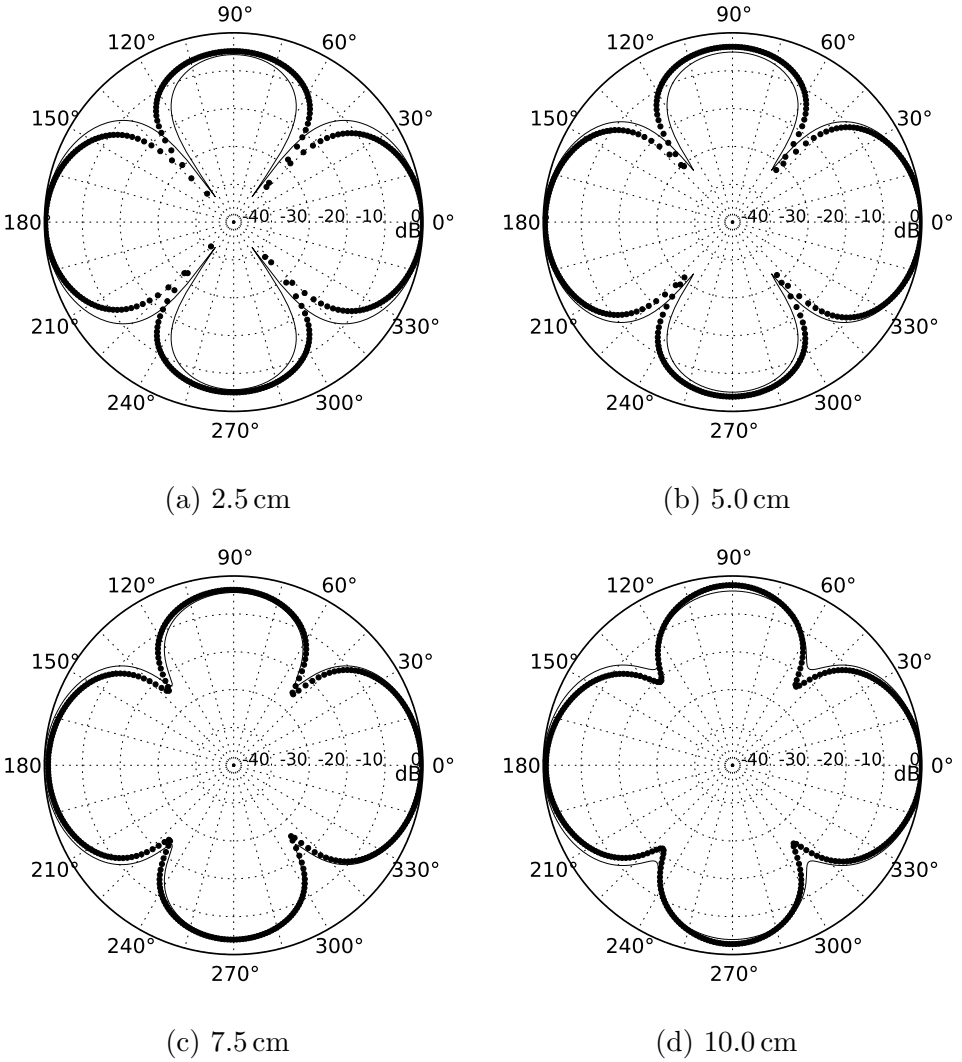
Figure 6.5: The sound pressure level L_p relative to a reference pressure $2 \times 10^{-5}\,\mathrm{Pa}$ for a range of frequencies, as measured over the last 9 periods of the base frequency as observed at location A. The frequencies of several eigenmodes of the linearized structure are shown for comparison.

Quality Factor

A major quantity of interest in a resonator system is the Q factor or quality factor. There are two equivalent definitions for the Q factor, one based on energy storage and losses and another based on resonance bandwidth. Here we consider the former, defining the Q factor as

$$Q = 2\pi \frac{E}{\Delta E} \tag{6.7}$$

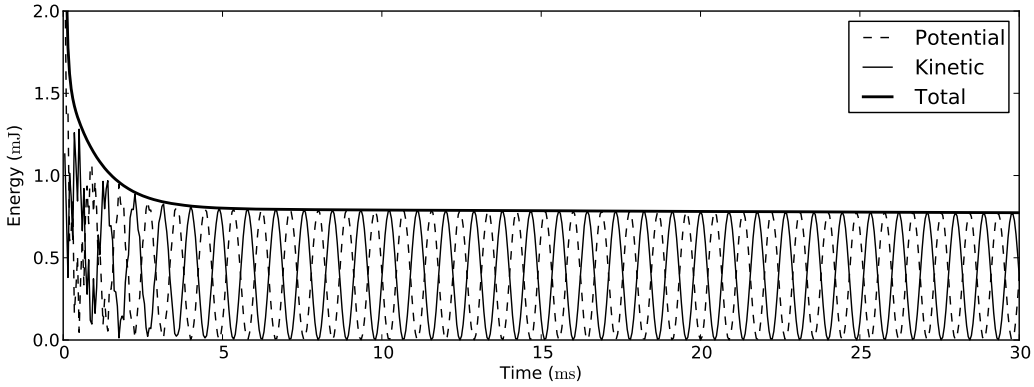
where E is the total energy stored in the resonator and ΔE is the energy dissipated per cycle. Since $\Delta E \ll E$, a bit of algebra shows that we can equivalently define the Q factor as the number of periods required for the energy to decay by $e^{-2\pi}$. In other words, $Q = 2\pi f \tau$ if the energy signal decays like $e^{-t/\tau}$. Since we have seen that the tuning fork emits an almost entirely pure signal at the fundamental frequency after 10 ms, we will consider 1 cycle to be one period of the fundamental mode.



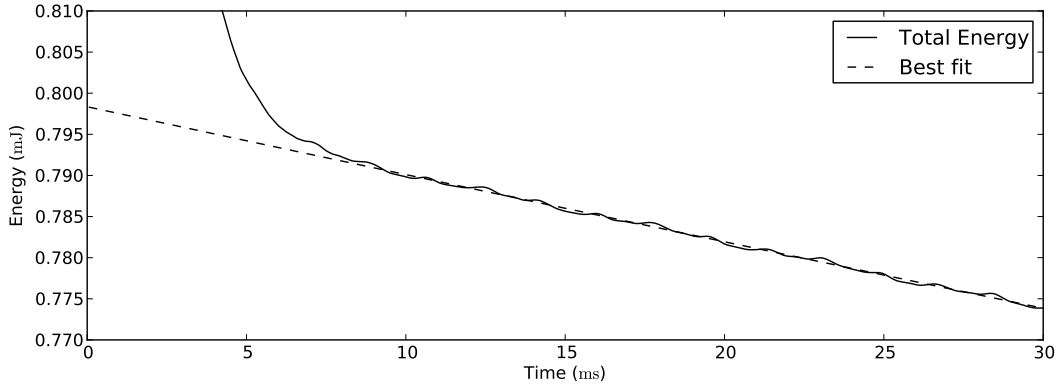
(a) 2.5 cm (b) 5.0 cm

(c) 7.5 cm (d) 10.0 cm

Figure 6.6: The relative sound pressure levels by angle at various distances from the axis of the tuning fork measured in 1° increments, averaged over nine periods of the fundamental mode. Each plot has been normalized to its maximum value. The theoretical curve for a linear quadrupole as given in eq. (6.5) is shown in a solid line. The tines of the tuning fork lie at 0° and 180°. Notice the 5 dB difference in sound pressure level between the two maxima in the extreme near-field.



(a) Energy time series.



(b) Zoomed in view with a best-fit curve.

Figure 6.7: Kinetic, potential, and total energy of the tuning fork.

We calculate the total energy of the tuning fork as

$$E = \underbrace{\int_V W \, dx}_{\text{potential}} + \underbrace{\int_V \frac{1}{2}\rho v^2 \, dx}_{\text{kinetic}} \tag{6.8}$$

where V is the reference configuration, ρ is the reference density, v is the material velocity, and W is the strain energy density as defined in eq. (2.40).

We show the potential, kinetic, and total energy contained within the tuning fork as a function of time in fig. 6.7a. Note the large initial losses due to the decay of high-frequency transients followed by a region of little decay. By changing the scale of the vertical axis we can better highlight the slow decay of the energy in the tuning fork, as shown in fig. 6.7b. Here we have fit an exponential decay curve to the total energy for the values after 10 ms. We see that the best fit curve quite closely approximates the decay in energy over many cycles, with only some small intra-cycle deviations as the tuning fork does not emit energy at a constant rate.

Table 6.1: Significant frequencies and Q factors observed in the time series pressure data at location A (see fig. 6.3a) during $5.0\,\text{ms} \leq t \leq 30.0\,\text{ms}$, as extracted by the filter diagonalization method.

Frequency (Hz)	Q Factor	Notes
196.0	453.1	Asymmetric in-plane
562.2	3414.0	Fundamental mode
1459.0	194.8	Asymmetric in-plane
3424.0	22.8	Clang mode

The best fit exponential has the form $E \approx A\exp(-t/\tau)$ where we find $A = 0.798\,\text{mJ}$ and $\tau = .96\,\text{s}$. Since the fundamental frequency is $f = 564\,\text{Hz}$ we easily calculate the Q factor to be 3400 which is in the range expected for a tuning fork.

Filter Diagonalization & Harmonic Inversion

Another way to measure the Q factor is by running the pressure time series through a so-called harmonic inversion process. Here we approximate the pressure by a sum of decaying exponential functions

$$p(t) \approx \sum_k d_k e^{-i\omega_k t} \tag{6.9}$$

with complex-valued parameters d_k and ω_k, where ω_k encodes the resonant frequency and Q factor of the k-th mode.

There are many such ways to decompose a signal into such a series. For example, the Fourier transform (fig. 6.5) is already such a series, however its numerical stability comes at the expense of poor frequency resolution as the ω_k are fixed with a linear spacing of $O(1/T)$ where T is the duration of the time series.

Here we employ the filter diagonalization method [33, 44, 67], using the freely available `Harminv` software [37]. We use the pressure time series data from location A (see fig. 6.3a) as the input signal and specify a frequency window of 100 Hz to 10,000 Hz. The method identifies the fundamental frequency $f = 562.6\,\text{Hz}$ with corresponding Q factor 3414.0. In addition, several other modes are well resolved and are shown in table 6.1. These modes include the clang mode and two asymmetric in-plane modes, each of which has a much smaller Q factor than the fundamental mode. Note that the identified frequencies are in good agreement with the modes predicted by the linear eigenvalue analysis as shown in fig. 6.5.

6.1.3 Conclusions

In this section we have demonstrated how high-order fluid-structure interaction methods can accurately capture the dynamics of a tuning fork, providing accurate predictions of

frequencies, angular sound pressure level distributions, Q factors, and damping rates.

Future work includes more realistic initial conditions (e.g., an impulsive hit with a mallet), a larger computational domain for far-field measurements, improved absorbing boundary conditions on the far walls, and the addition of a resonance box. In addition more work could be done to explore the higher symmetric modes as well as the asymmetric and out-of-plane modes.

Lastly we mention that techniques similar to the ones used in this work could be used to simulate a variety of other instruments including gongs, xylophones, and marimbas.

Chapter 7

Conclusions & Future Work

We have presented a high-order accurate scheme for fluid-structure interaction problems. By using a predictor for the fluid-to-structure coupling, the method allows the reuse of existing domain specific fluid and structure solvers while still maintaining a high-order of time accuracy. The accuracy of the method in space and in time were verified using grid convergence studies. The overall implementation was validated by comparing the results of standard fluid-structure interaction test problem to other values reported in the literature.

We demonstrated the applicability of this method to several interesting problems, including membrane and flapping wing aerodynamics and acoustics. In doing so we showed that this method is versatile and robust, and easily scales to large three-dimensional simulations.

Yet much more work remains. We would like to better understand the stability of the method. While the overall stability of the method never caused an issue in practice, the scheme did exhibit instabilities for large time steps. To some extent stability was improved using subiterations, and effect which merits further investigation.

We would like to develop more sophisticated mesh deformation procedures to allow for larger more complicated motions. For example, our three-dimensional wing simulation in section 5.4 involved only a $\pm 15^\circ$ flapping motion, which we would like to extend to a more realistic $\pm 45^\circ$ or higher. In the same vein we would like to improve our flapping flight model to include internal structure within the wing, much like how the bones run through the wing of the bat. This could be accomplished by creating a mesh whose elements could be partitioned into regions of membrane and bone and separate physical parameters could be assigned to those regions. Alternatively several patches of membrane could be coupled to one rigid or semi-rigid bone structure, although this would greatly complicate the structure solver.

Lastly we are interested in investigating musical instruments with more complicated sound generation mechanisms, particularly air-reed instruments like clarinets, oboes, and saxophones.

Bibliography

[1] H.T. Ahn and Y. Kallinderis. "Strongly coupled flow/structure interactions with a geometrically conservative ALE scheme on general hybrid meshes". In: *J. Comput. Phys.* 219.2 (2006), pp. 671–696. DOI: 10.1016/j.jcp.2006.04.011.

[2] American National Standards Institute. *Acoustical terminology*. ANSI S1.1-1994. New York, 1994.

[3] P.R. Amestoy, I.S. Duff, J. Koster, and J.-Y. L'Excellent. "A fully asynchronous multifrontal solver using distributed dynamic scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41. DOI: 10.1137/S0895479899358194.

[4] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. "Hybrid scheduling for the parallel solution of linear systems". In: *Parallel Computing* 32.2 (2006), pp. 136–156. DOI: 10.1016/j.parco.2005.07.004.

[5] E. Anderson et al. *LAPACK users' guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 978-0-89871-447-0; 0-89871-447-8.

[6] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. "Unified analysis of discontinuous Galerkin methods for elliptic problems". In: *SIAM J. Numer. Anal.* 39.5 (2001/02), pp. 1749–1779. DOI: 10.1137/S0036142901384162.

[7] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. "Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations". In: *Appl. Numer. Math.* 25.2–3 (1997). Special issue on time integration (Amsterdam, 1996), pp. 151–167. DOI: 10.1016/S0168-9274(97)00056-1.

[8] J.W. Bahlman, S.M. Swartz, and K.S. Breuer. "Design and characterization of a multi-articulated robotic bat wing". In: *Bioinspir. Biomim.* 8 (1 2013), p. 016009. DOI: 10.1088/1748-3182/8/1/016009.

[9] A. Beckert and H. Wendland. "Multivariate interpolation for fluid-structure-interaction problems using radial basis functions". In: *Aerosp. Sci. Technol.* 5 (2 2001), pp. 125–134. DOI: 10.1016/S1270-9638(00)01087-7.

[10] W. Benson, J.W. Harris, H. Stocker, and H. Lutz, eds. *Handbook of physics*. Springer-Verlag New York, 2002, pp. xxv+1181. ISBN: 978-0-387-95269-7; 0-387-95269-1.

[11] K.Y. Billah and R.H. Scanlan. "Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks". In: *Am. J. Phys.* 59 (2 1991), pp. 118–124. DOI: 10.1119/1.16590.

[12] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK users' guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997. ISBN: 978-0-89871-397-8; 0-89871-397-8.

[13] A. de Boer, M.S. van der Schoot, and H. Bijl. "Mesh deformation based on radial basis function interpolation". In: *Comput. Struct.* 85 (11–14 2007), pp. 784–795. DOI: 10.1016/j.compstruc.2007.01.013.

[14] J. Bonet and R.D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge: Cambridge University Press, 1997, pp. xviii+248. ISBN: 978-0-521-57272-9; 0-521-57272-X.

[15] A. Boocher and B. Froehle. "On generators of bounded ratios of minors for totally positive matrices". In: *Linear Algebra Appl.* 428.7 (2008), pp. 1664–1684. DOI: 10.1016/j.laa.2007.10.011.

[16] D. Botteldooren. "Acoustical finite-difference time-domain simulation in a quasi-Cartesian grid". In: *J. Acoust. Soc. Am.* 95 (5 1994), pp. 2313–2319. DOI: 10.1121/1.409866.

[17] B. Cockburn and C.-W. Shu. "The local discontinuous Galerkin method for time-dependent convection-diffusion systems". In: *SIAM J. Numer. Anal.* 35.6 (1998), pp. 2440–2463. DOI: 10.1137/S0036142997316712.

[18] J.F. Cori, S. Etienne, D. Pelletier, and A. Garon. "Implicit Runge-Kutta time integrators for fluid-structure interactions". In: *48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida*. AIAA-2010-1445. Jan. 2010. DOI: 10.2514/6.2010-1445.

[19] P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, and A. Quateroni. "Fluid-structure interaction simulation of aortic blood flow". In: *Comput. & Fluids* 43 (1 2011), pp. 46–57. DOI: 10.1016/j.compfluid.2010.11.032.

[20] T.A. Davis. "Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method". In: *ACM Trans. Math. Software* 30.2 (2004), pp. 196–199. DOI: 10.1145/992200.992206.

[21] B. Desjardins, E. Grenier, P.-L. Lions, and N. Masmoudi. "Incompressible limit for solutions of the isentropic Navier-Stokes equations with Dirichlet boundary conditions". In: *J. Math. Pures Appl. (9)* 78.5 (1999), pp. 461–471. DOI: 10.1016/S0021-7824(99)00032-X.

[22] W. Dettmer and D. Perić. "A computational framework for fluid-structure interaction: finite element formulation and applications". In: *Comput. Methods Appl. Mech. Engrg.* 195.41-43 (2006), pp. 5754–5779. DOI: 10.1016/j.cma.2005.10.019.

[23] W.G. Dettmer and D. Perić. "A new staggered scheme for fluid-structure interaction". In: *Internat. J. Numer. Methods Engrg.* 93.1 (2013), pp. 1–22. DOI: 10.1002/nme.4370.

[24] C. Farhat and P. Geuzaine. "Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids". In: *Comput. Methods Appl. Mech. Engrg.* 193.39-41 (2004), pp. 4073–4095. DOI: 10.1016/j.cma.2003.09.027.

[25] C. Farhat and M. Lesoinne. "Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems". In: *Comput. Methods Appl. Mech. Engrg.* 182.3-4 (2000), pp. 499–515. DOI: 10.1016/S0045-7825(99)00206-6.

[26] A.C. Faul and M.J.D. Powell. "Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions". In: *Adv. Comput. Math.* 11.2-3 (1999). Radial basis functions and their applications, pp. 183–192. DOI: 10.1023/A:1018923925800.

[27] C.L. Fefferman. *Existence and smoothness of the Navier-Stokes equation*. Clay Mathematics Institute Millenium Prize Problems. URL: http://www.claymath.org/millennium/Navier-Stokes_Equations/navierstokes.pdf.

[28] C.A. Felippa, K.C. Park, and C. Farhat. "Partitioned analysis of coupled mechanical systems". In: *Computer Methods in Applied Mechanics and Engineering* 190.24-25 (2001), pp. 3247–3270. DOI: 10.1016/S0045-7825(00)00391-1.

[29] L. Formaggia, A. Quarteroni, and A. Veneziani, eds. *Cardiovascular mathematics*. Vol. 1. MS&A. Modeling, Simulation and Applications. Modeling and simulation of the circulatory system. Springer-Verlag Italia, Milan, 2009, pp. xiv+522. ISBN: 978-88-470-1151-9. DOI: 10.1007/978-88-470-1152-6.

[30] B. Froehle and P.-O. Persson. "A high-order implicit-explicit fluid-structure interaction method for flapping flight". In: *21st AIAA Computational Fluid Dynamics Conference, San Diego, California*. June 2013. DOI: 10.2514/6.2013-2690.

[31] K.-Y. Fung and H. Ju. "Time-domain impedance boundary conditions for computational acoustics and aeroacoustics". In: *Int. J. Comput. Fluid Dyn.* 18 (6 2004), pp. 503–511. DOI: 10.1080/10618560410001673515.

[32] P. Geuzaine, C. Grandmont, and C. Farhat. "Design and analysis of ALE schemes with provable second-order time-accuracy for inviscid and viscous flow simulations". In: *J. Comput. Phys.* 191.1 (2003), pp. 206–227. DOI: 10.1016/S0021-9991(03)00311-5.

[33] S. Govindjee and P.-O. Persson. "A time-domain discontinuous Galerkin method for mechanical resonator quality factor computations". In: *J. Comput. Phys.* 231.19 (2012), pp. 6380–6392. DOI: 10.1016/j.jcp.2012.05.034.

[34] C. Habchi, S. Russeil, D. Bougeard, J.-L. Harion, T. Lemenand, A. Ghanem, D.D. Valle, and H. Peerhossaini. "Partitioned solver for strongly coupled fluid-structure interaction". In: *Comput. & Fluids* 71 (2013), pp. 306–319. DOI: 10.1016/j.compfluid.2012.11.004.

[35] H.L.F. Helmholtz. *On the sensations of tone as a physiological basis for the theory of music*. Third. London: Longmans, Green, and Co., 1895, pp. xix+576.

[36] G.A. Holzapfel. *Nonlinear solid mechanics*. A continuum approach for engineering. Chichester: John Wiley & Sons Ltd., 2000, pp. xiv+455. ISBN: 978-0-471-82304-9; 0-471-82304-X.

[37] S.G. Johnson. *Harminv*. Version 1.3.1. 2006. URL: `http://ab-initio.mit.edu/harminv/`.

[38] G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: *SIAM J. Sci. Comput.* 20.1 (1998), pp. 359–392. DOI: `10.1137/S1064827595287997`.

[39] C. Kassiotis, A. Ibrahimbegovic, R. Niekamp, and H.G. Matthies. "Nonlinear fluid-structure interaction problem. Part I: implicit partitioned algorithm, nonlinear stability proof and validation examples". In: *Comput. Mech.* 47.3 (2011), pp. 305–323. DOI: `10.1007/s00466-010-0545-6`.

[40] C.A. Kennedy and M.H. Carpenter. "Additive Runge-Kutta schemes for convection-diffusion-reaction equations". In: *Appl. Numer. Math.* 44.1-2 (2003), pp. 139–181. DOI: `10.1016/S0168-9274(02)00138-1`.

[41] T. Koto. "IMEX Runge-Kutta schemes for reaction-diffusion equations". In: *J. Comput. Appl. Math.* 215.1 (2008), pp. 182–195. DOI: `10.1016/j.cam.2007.04.003`.

[42] C.K. Lin. "On the incompressible limit of the compressible Navier-Stokes equations". In: *Comm. Partial Differential Equations* 20.3-4 (1995), pp. 677–707. DOI: `10.1080/03605309508821108`.

[43] I. Lomtev, R.M. Kirby, and G.E. Karniadakis. "A discontinuous Galerkin ALE method for compressible viscous flows in moving domains". In: *J. Comput. Phys.* 155.1 (1999), pp. 128–159. DOI: `10.1006/jcph.1999.6331`.

[44] V.A. Mandelshtam and H.S. Taylor. "Harmonic inversion of time signals and its applications". In: *J. Chem. Phys.* 107.17 (1997), pp. 6756–6769. DOI: `10.1063/1.475324`.

[45] H.G. Matthies and J. Steindorf. "Partitioned strong coupling algorithms for fluid-structure interaction". In: *Comput. Struct.* 81.8–11 (2003), pp. 805–812. DOI: `10.1016/S0045-7949(02)00409-1`.

[46] M. Olivier, J.-F. Morissette, and G. Dumas. "A fluid-structure interaction solver for nano-air-vehicle flapping wings". In: *19th AIAA Computational Fluid Dynamics, San Antonio, Texas*. AIAA-2009-3676. June 2009. DOI: `10.2514/6.2009-3676`.

[47] J. Peraire and P.-O. Persson. "Adaptive high-order methods in computational fluid dynamics". In: vol. 2. Advances in CFD. World Scientific Publishing Co., 2011. Chap. 5 – High-Order Discontinuous Galerkin Methods for CFD. ISBN: 978-981-4313-18-6; 981-4313-18-1.

[48] J. Peraire and P.-O. Persson. "The compact discontinuous Galerkin (CDG) method for elliptic problems". In: *SIAM J. Sci. Comput.* 30.4 (2008), pp. 1806–1824. DOI: 10.1137/070685518.

[49] P.-O. Persson. *DistMesh*. Version 1.1. 2012. URL: http://persson.berkeley.edu/distmesh/.

[50] P.-O. Persson and G. Strang. "A simple mesh generator in Matlab". In: *SIAM Rev.* 46.2 (2004), 329–345 (electronic). DOI: 10.1137/S0036144503429121.

[51] P.-O. Persson. "Scalable parallel Newton-Krylov solvers for discontinuous Galerkin discretizations". In: *47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida*. AIAA-2009-606. Jan. 2009. DOI: 10.2514/6.2009-606.

[52] P.-O. Persson, J. Bonet, and J. Peraire. "Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains". In: *Comput. Methods Appl. Mech. Engrg.* 198.17–20 (2009), pp. 1585–1595. DOI: 10.1016/j.cma.2009.01.012.

[53] P.-O. Persson and J. Peraire. "Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations". In: *SIAM J. Sci. Comput.* 30.6 (2008), pp. 2709–2733. DOI: 10.1137/070692108.

[54] P.-O. Persson, J. Peraire, and J. Bonet. "A high order discontinuous Galerkin method for fluid-structure interaction". In: *18th AIAA Computational Fluid Dynamics Conference, Miami, Florida*. AIAA-2007-4327. June 2007. DOI: 10.2514/6.2007-4327.

[55] S. Piperno, C. Farhat, and B. Larrouturou. "Partitioned procedures for the transient solution of coupled aeroelastic problems. I. Model problem, theory and two-dimensional application". In: *Comput. Methods Appl. Mech. Engrg.* 124.1-2 (1995), pp. 79–112. DOI: 10.1016/0045-7825(95)92707-9.

[56] J.J. Reuther, J.J. Alonso, J.R.R.A. Martins, and S.C. Smith. "A coupled aero-structural optimization method for complete aircraft configurations". In: *37th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*. AIAA-99-0187. Jan. 1999. DOI: 10.2514/6.1999-187.

[57] P.L. Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes". In: *J. Comput. Phys.* 43.2 (1981), pp. 357–372. DOI: 10.1016/0021-9991(81)90128-5.

[58] T.D. Rossing, D.A. Russell, and D.E. Brown. "On the acoustics of tuning forks". In: *Am. J. Phys.* 60.7 (1992), pp. 620–626. DOI: 10.1119/1.17116.

[59] D.A. Russell. "On the sound field radiated by a tuning fork". In: *Am. J. Phys.* 68.12 (2000), pp. 1139–1145. DOI: 10.1119/1.1286661.

[60] W. Shyy, M. Berg, and D. Ljungqvist. "Flapping and flexible wings for biological and micro air vehicles". In: *Prog. Aerosp. Sci.* 35.5 (1999), pp. 455–505. DOI: 10.1016/S0376-0421(98)00016-5.

[61] R.M. Sillitto. "Angular distribution of the acoustic radiation from a tuning fork". In: *Am. J. Phys.* 34.8 (1966), pp. 639–644. DOI: 10.1119/1.1973192.

[62] R.W. Smith and J.A. Wright. "A classical elasticity-based mesh update method for moving and deforming meshes". In: *48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida*. AIAA-2010-164. Jan. 2010. DOI: 10.2514/6.2010-164.

[63] A. Song, X. Tian, E. Israeli, R. Galvao, K. Bishop, S. Swartz, and K. Breuer. "Aeromechanics of membrane wings with implications for animal flight". In: *AIAA J.* 46.8 (2008), pp. 2096–2106. DOI: 10.2514/1.36694.

[64] P.D. Thomas and C.K. Lombard. "Geometric conservation law and its application to flow computations on moving grids". In: *AIAA J.* 17 (10 1979), pp. 1030–1037. DOI: 10.2514/3.61273.

[65] C.S. Venkatasubban. "A new finite element formulation for ALE (arbitrary Lagrangian Eulerian) compressible fluid mechanics". In: *Internat. J. Engrg. Sci.* 33.12 (1995), pp. 1743–1762. DOI: 10.1016/0020-7225(95)00021-O.

[66] E. Walhorn, B. Hübner, and D. Dinkler. "Space-time finite elements for fluid-structure interaction". In: *PAMM* 1.1 (2002), pp. 81–82. DOI: 10.1002/1617-7061(200203)1:1<81::AID-PAMM81>3.0.CO;2-1.

[67] M.R. Wall and D. Neuhauser. "Extraction, through filter-diagonalization, of general quantum eigenvalues or classical normal mode frequencies from a small number of residues or a short-time segment of a signal. I. Theory and application to a quantum-dynamics model". In: *J. Chem. Phys.* 102.20 (1995), pp. 8011–8022. DOI: 10.1063/1.468999.

[68] W.A. Wall and E. Ramm. "Fluid-structure interaction based upon a stabilized (ALE) finite element method". In: *4th World Congress on Computational Mechanics: New Trends and Applications*. Ed. by S.R. Idelsohn, E. Oñate, and E.N. Dvorkin. Barcelona, Spain: CIMNE, 1998.

[69] C. Wood, A.J. Gil, O. Hassan, and J. Bonet. "Partitioned block-Gauss-Seidel coupling for dynamic fluid-structure interaction". In: *Comput. Struct.* 88.23-24 (2010), pp. 1367–1382. DOI: 10.1016/j.compstruc.2008.08.005.

[70] Z. Yosibash, R.M. Kirby, K. Myers, B. Szabó, and G. Karniadakis. "High-order finite elements for fluid-structure interaction problems". In: *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference, Norfolk, Virginia*. AIAA-2003-1729. Apr. 2003. DOI: 10.2514/6.2003-1729.

[71] C. Yvin. "Partitioned fluid-structure interaction with open-source tools". In: *12ème Journées de l'Hydrodynamique, Nantes, France*. Oct. 2010.

[72] M.J. Zahr and P.-O. Persson. "Performance tuning of Newton-GMRES methods for discontinuous Galerkin discretizations of the Navier-Stokes equations". In: *21st AIAA Computational Fluid Dynamics Conference, San Diego, California*. June 2013. DOI: 10.2514/6.2013-2685.

[73] A.H. van Zuijlen. "Fluid-structure interaction simulations: efficient higher order time integration of partitioned systems". PhD thesis. TU Delft, Nov. 2006.

[74] A.H. van Zuijlen and H. Bijl. "A higher-order time integration algorithm for the simulation of nonlinear fluid-structure interaction". In: *Nonlinear Anal.* 63.5–7 (2005), e1597–e1605. DOI: 10.1016/j.na.2005.01.054.

[75] A.H. van Zuijlen, A. de Boer, and H. Bijl. "Higher-order time integration through smooth mesh deformation for 3D fluid-structure interaction simulations". In: *J. Comput. Phys.* 224 (2007), pp. 414–430. DOI: 10.1016/j.jcp.2007.03.024.