UNIVERSITY OF CALIFORNIA

Santa Barbara

Seeing and Hearing Fluid Subspaces

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy

in

Media Arts and Technology

by

Aaron Demby Jones

Committee in Charge:

Professor Theodore Kim, Chair

Professor JoAnn Kuchera-Morin

Professor Clarence Barlow

December 2017

The dissertation of Aaron Demby Jones is approved.

Professor JoAnn Kuchera-Morin

Professor Clarence Barlow

Professor Theodore Kim, Committee Chair

July 2017

Seeing and Hearing Fluid Subspaces

Acknowledgements

This work would not be possible without the help and support of many of my colleagues, friends, and family.

First, I would like to express the deepest gratitude to my committee chair, Professor Ted Kim. I was immediately inspired by your pattern formation class. Somehow, I managed to maneuver into the study of computer graphics despite having little background experience. You were always very patient with me and managed to guide me through the various minefields of C++ programming and computer graphics algorithms. There were many moments of difficulty and frustration, but you helped me persevere in the end.

I am very grateful to Professor JoAnn Kuchera-Morin for serving on my committee. Your computer music class on Gamma helped inspire me to pursue the degree in Media Arts and Technology in the first place. We had many interesting and valuable discussions on the compositional process.

My sincere appreciation goes to Professor Clarence Barlow for serving on my committee. Every single one of your music courses was a revelation to me, and I certainly would not be in the Media Arts and Technology department without your influence.

My research in data compression owes a great debt to my collaborator Professor Pradeep Sen, who provided many insightful comments and ideas.

My thanks to my fellow classmates in Media Arts and Technology, who were a source of both technical and creative inspiration: Karl Yerkes, Sahar Sajadieh, Hannah Wolfe, Pablo Colapinto, Charlie Roberts, Michael Hetrick, Sterling Crispin, Yun Teng, Qiaodong Cui, Yuxiang Wang, and many others along the way.

My thanks to my friends nearby and far away, who provided much-needed diversion as well support in difficult times: Sarah Davis, Kyle Stewart, Chris Mendes, Maritza Fuljencio, Matt Guidry, Jason Garfield, Kyle Miller, Mike Morris, Jason Rothschild, Jamie Pommersheim, and Clara Bakker.

Finally, to my mom—thank you for supporting and believing in me, no matter what. To my partner Lauren, for many years of support, love, and encouragement.

And lastly, to my cat, Professor Charles, who has been waiting long enough for the last member of the family to get a PhD.

Vita of Aaron Demby Jones

Contact Information

Media Arts and Technology Program
University of California, Santa Barbara
Santa Barbara, CA 93106-5080

Phone: (610) 334-1064
E-mail: ajones@aops.com

Education

University of California, Santa Barbara **2011–2017**
PhD Candidate, Media Arts and Technology Program Santa Barbara, California

University of Rochester **2009–2011**
BA, Mathematics Rochester, New York

Brown University **2005–2009**
BA, Music, Mathematics Providence, Rhode Island

Honors and Awards

Chancellor's Fellowship, University of California, Santa Barbara **2011–2017**
Best Paper Award, Symposium on Computer Animation **2016**
Muriel Hassenfeld Mann Premium, Brown University **2009**
Buxtehude Premium, Brown University **2008**
Margery MacColl Award for Musical Excellence, Brown University **2007**

Publications

Jones, Aaron Demby, Kuchera-Morin, JoAnn, and Kim, Theodore. Seeing and hearing the eigenvectors of a fluid. In *Proceedings of Bridges 2017: Mathematics, Art, Music, Architecture, Education, Culture.* Tessellations Publishing, 2017.

Jones, Aaron Demby, Sen, Pradeep, and Kim, Theodore. Compressing fluid subspaces. In *Proceedings of the 2016 ACM SIGGRAPH/Eurographics Symposium on ComputerAnimation.* Eurographics Association, 2016.

Research Experience

University of California, Santa Barbara **2011–2017**
Ph.D. Candidate Santa Barbara, California
Sonification of subspace methods in computational fluid dynamics for computer graphics.

Advisor: Theodore Kim
Media Arts and Technology Program

Brown University **2008–2009**
Senior Honors Thesis Providence, Rhode Island
Percussion Quartet

Advisor: Gerald Shapiro
Department of Music

Teaching Experience

University of California, Santa Barbara **Spring 2015**
Teaching Assistant, Pattern Formation Santa Barbara, California

University of California, Santa Barbara **Fall 2014**
Teaching Assistant, Music and Technology Santa Barbara, California

University of California, Santa Barbara **2012–2013**
Teaching Assistant, Music Appreciation Santa Barbara, California

Hampshire College **Summer 2014**
Junior Faculty
Hampshire College Summer Studies in Mathematics Amherst, Massachusetts

Franklin and Marshall College **Summer 2012–2013, 2015–2017**
Instructor, Number Theory
Johns Hopkins Center for Talented Youth Lancaster, Pennsylvania

University of Rochester **2010–2011**
Teaching Assistant, Linear Algebra with Differential Equations Rochester, New York

University of Rochester **2009–2010**
Teaching Assistant, Calculus II Rochester, New York

Community Involvement

Peer Review **Fall 2016**

Reviewer for Eurographics 2017; Computer Graphics Forum

Abstract

Seeing and Hearing Fluid Subspaces

by

Aaron Demby Jones

Fluids have inspired generations of artists and scientists throughout history. Aesthetically, the wide variety of abstract shapes they form is both surprising and pleasing. Besides visual art, which until the digital age mostly captured frozen moments in time, late 19th-century composers such as Debussy and Ravel wrote works of music inspired by the movement of fluids over time. With the framework of several basic conservation laws of physics, earlier 19th-century scientific work discovered a set of differential equations called the *Navier-Stokes equations* that described the time evolution of fluid velocity fields.

In recent years, the advent of higher computing power and the birth of computer graphics as a discipline has given rise to computational methods for approximating and visualizing solutions to the Navier-Stokes equations, which had previously remained intractably complex. Many artists and musicians have also embraced digital technologies, allowing for the development of algorithmically generated music as well as multimodal representations of large, complex data sets.

With this new technology, it is natural to consider the following question: is it possible to *systematically* generate sounds from fluid dynamics while retaining an underlying musicality? In this dissertation, we present a framework for generating correlated correlated fluid motions and musical sounds using the empirical eigenvalues of a subspace fluid simulation. Our method is multimodal in nature, allowing for the generation of musical sound as well as novel visual forms. The specific mapping from fluid velocity to sound chosen allows for control and modulation of both the visuals and the audio in an integrated, unifying fashion.

The method of subspace simulation, which our mapping framework relies on, has a known drawback of high memory consumption. As a means of overcoming this technical obstacle, we also present a data compression framework for fluid subspaces. Our proposed algorithm can achieve an order of magnitude data compression without any noticeable visual artifacts. Using this compression algorithm allows the potential for simulating greater variety of complex scenes on powerful computers as well as the ability to run previously too-complex scenes on a laptop.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The abstract form of fluids in the natural world has inspired artists and scientists for centuries. For instance, besides studying water from an artistic perspective, Leonardo da Vinci (1452–1519) also discovered the law of conservation of mass for incompressible one-dimensional flows [1]. A myriad of impressionistic works such as Hokusai's *The Great Wave off Kanagawa* (1829–1833), van Gogh's *The Starry Night* (1889), Monet's *Water Lilies* (1897–1926), and many others draw inspiration from the variety of moods that water and air express, capturing destruction, abstraction, and serenity, all with their still images. In the sonic arts, composers such as Debussy and Ravel explored the unfolding undulations of fluids over time in a way that visual artists could not, composing impressionistic pieces such as and *Jeux d'eau* (1901), *La Mer* (1903–1905), *Une barque* (1904–1905), *Reflets dans l'eau* (1905), and *Le vent dans la plaine* (1909–1910). Through their harmonic and rhythmic language, these pieces abstractly depict the variety of moods that fluids can express, from turbulent spray to calm stillness. However, a formal interplay between a particular time-evolving process of fluid dynamics and time-evolving sound is not present. Such formal approaches to generating art were not explored until the twentieth century.

The Greek composer and architect, Iannis Xenakis (1922–2001), was an early pioneer of the use of mathematical processes in music composition. Two of his compositions, *Pithoprakta* (1955–56) and *N'Shima* (1975) relied on mathematical processes derived from the statistical mechanics of gases on a molecular level. His magnum opus, *Formalized Music: Thought and Mathematics in Music* [2], expounds his aesthetic philosophy as well as his techniques for composing music stochastically through mathematical processes. Many other composers and researchers followed up in the direction of algorithmic composition, albeit with their own individual aesthetic goals, including Karlheinz Stockhausen [3], Max Mathews [4], John Chowning [5], and John Cage [6].

In the visual arts, more powerful computing and research into computer graphics eventually led to the area of physics-based fluid animation. Early research in the field of computational fluid dynamics and mechanical engineering in the 1960s [7, 8, 9] became translated into the domain of computer graphics starting in the 1990s. Work by O'Brien [10], Fedkiw [11], Bridson [12], Stam [13], and others paved the way for many different types of fluid animation, ranging from real-time, simplistic effects suitable for computer games, to photorealistic images for films. A large body of work has been dedicated to improving the efficiency of these physics-based simulations, including fluid-implicit particle methods (FLIP), spatial coarsening methods, position-based methods, and vortex-sheet methods [14, 15, 16, 17].

The experiment of Ernst Chladni (1756–1827), involving the patterns of sand accumulating on the surface of vibrating metal plates, was an early audiovisual link between visual shape and sonic frequency. Using a violin bow to vibrationally excite a metal plate covered with sand, Chladni observed a variety of intriguing nodal patterns, depending on the shape of the plate as well as the frequency of the vibration. The Swiss scientist Hans Jenny (1904–1972) pioneered the general study of this prin-

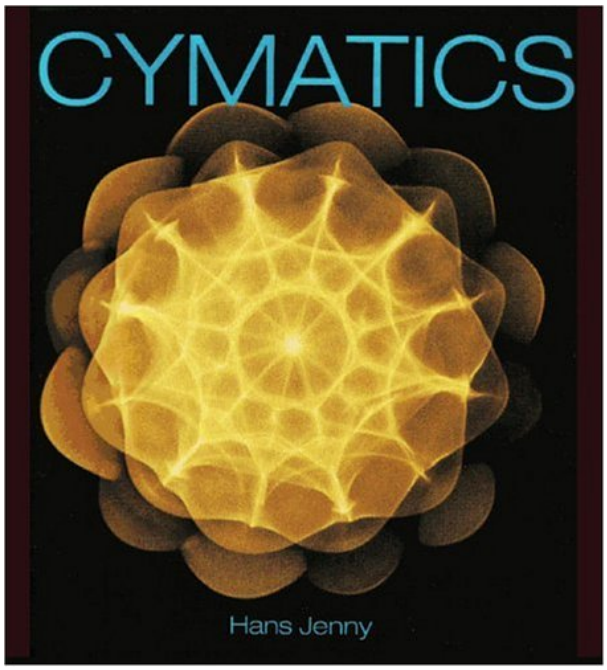


Figure 1.1: *One of the visual patterns from Hans Jenny's seminal text.*

ciple of visual patterns (as in Figure 1.1) associated with sonic vibrations, which he named "Cymatics." Many composers and artists followed in the footsteps of Jenny, including Alvin Lucier [18], György Kepes [19], and Alexander Lauterwasser [20].

This dissertation serves as a bridge between generative art and sound, computer graphics, and cymatics. Using computational fluid dynamics simulations as our background data, we define a bridge between the visual and sonic results in the spirit of cymatics, linking resonant visual shapes of vibration to their corresponding audio frequencies. Along the way, we devise a data compression algorithm as part of the simulation pipeline to ease the computational load.

1.1 Systems of Sonification and Aesthetic Goals

The term "sonification" has many subtly different interpretations and definitions, but we use it in this dissertation to refer broadly to the association of data to sound.

Scientifically-focused sonifications can reveal underlying qualitative patterns in the data, while artistic-focused sonifications can serve as a framework for composing novel musical or even audiovisual works. In this dissertation, we use the phenomenon of fluid dynamics as our data, focusing on a system of sonification that, while revealing certain qualitative features of the fluid motion, mostly serves to generate novel audiovisual pieces. More precisely, we use the the *subspace* method of simulation [21], based on a stable computational fluid dynamics simulation, as our data, and a model-based sonification system to associate the data with musical sound. Such a sonification system is analogous to the resonant modal shapes and frequencies as discovered by Chladni and Jenny in the field of cymatics.

1.2 Data Compression and Memory Obstacles with Subspace Matrices

The general technique of data compression began with Claude Shannon [22] and the field of information theory [23]. It is of practical relevance for many problems in computing, as reducing memory footprints can make certain simulations more feasible to run. The general idea of data compression is that by exploiting patterns or redundancies in data, we can often represent the same data in a more terse form. A wide variety of research in this area has produced many successful compression schemes, from lossless compression of arbitrary files (ZIP) to lossy compression of image (JPEG), audio (MP3), and video (MPEG) [24, 25, 26, 27].

The subspace method of simulation [28], while known to achieve large speedups over regular full-space simulations, also requires a potentially prohibitive memory cost, consuming dozens of gigabytes of RAM in high-resolution, three-dimensional simulations. While other research has been done on compressing blendshape ma-

trices for facial animations [29] and eigenmode matrices for modal sound synthesis [30], to our knowledge, no previous work on compressing fluid subspace matrices has been done.

1.3 Thesis Statement and Main Results

The thesis statement is as follows:

The method of subspaces in computational fluid dynamics for computer graphics can be leveraged to systematically produce artistic audiovisual pieces.

In particular, I demonstrate three main results:

- I propose a versatile system of sonification of fluid subspaces.
- I describe a data compression algorithm for fluid subspaces to help ease high computational costs.
- I demonstrate the sonification system with several audiovisual études.

The first result, the sonification system, achieves a two-way coupling between sound and visual, allowing the visual to drive the sound, the sound to drive the visual, or even a third intermediary process to govern both the sound and visual simultaneously. The flexibility of the system allows the composer to generate a variety of sounds and visual forms that are nonetheless intertwined in a common language.

The second result, the data compression algorithm, achieves an order of magnitude compression of fluid subspace matrices needed in memory during runtime without any perceivable visual artifacts. The main result is the technique behind the compression scheme, a transform-based algorithm, combined with a novel sparse frequency-domain projection and reconstruction.

The third result, the audiovisual études, serve as proof of concept of the possibility of the sonification system as a means toward developing an audiovisual language. Carefully chosen parameters are modulated to demonstrate compositional gestures that can be achieved in both the audio and visual domain.

1.4 Organization

The dissertation is organized as follows. Chapter 2 provides background and related work in the domain of generative art and sonification. Chapter 3 gives additional background in the domain of fluid simulation for computer graphics, focusing in particular on the method of subspaces. Chapter 4 describes the data compression algorithm for fluid subspaces and shows its effectiveness on a variety of different subspace simulations. In Chapter 5 we describe the system of sonification of the fluid subspaces. Chapter 6 demonstrates the versatility of the sonification system through a series of systematic audiovisual études. Finally, Chapter 7 presents conclusions and directions for future work.

Chapter 2

Related Work

2.1 Sonification and Generative Art

The question of mapping computational fluid dynamics data into sound belongs to the intersection of the domains of *sonification* and *generative art*. Sources vary in agreement on the definition of sonification. According to Hermann [31], sonification is "the technique of rendering sound in response to data and interactions." Kramer et al. [32] define it as "the use of nonspeech audio to convey information." We shall make a distinction between *scientific* sonification, which aims primarily toward conveying information clearly, and *musical sonification*, which aims primarily toward aesthetically useful generation of music. There are many possible strategies for sonification, including audification, parameter mapping sonification, and model-based sonification [31]. We shall expand upon the meaning of these terms in §2.1.1.

Generative art is sometimes thought of in very general terms. For instance, Boden specifies eleven different categories of generative art: electronic k art, computer art, computer-assisted art digital art, generative art, computer-generated art, evolutionary art, robot art, interactive art, computer-interactive art, and virtual reality art [33]. In

the text, however, we prefer a more narrow definition of generative art as art which has been created through the design and use of a computational system.

As Curtis Roads observes in [34], one of the main attractions of this compositional style is the possibility of novel discoveries that go beyond what the artist may otherwise have been able to conceive: "The computer can allow a composer to write music that goes beyond that which she is already capable of."

2.1.1 Background

The use of natural processes as a technique in music composition has been well known for centuries, from the Greek's use of harmonic proportions in tuning to Mozart's dice music. The twentieth century saw an increased interest in formalized mathematical systems as part of musical composition. Composers and theorists such as Schillinger, Stockhausen, Xenakis, Cage, Lucier, Lewin, and many others explored different systematic ways of generating sonic art [2, 3, 6, 18, 35, 36]. In the twenty-first century, with the explosion of the idea of "big data," music and sound has been generated from all sorts of data sets such as astronomical measurements, seismographs, web traffic, and more. Gradually, an important question emerged: is the composer focused primarily with the listener's musical experience, or is it the intent that the music inform the listener of features of the underlying data? Such a question touches the tip of the iceberg of many challenging philosophical questions, such as whether music can even have any external meaning, or the distinction between the artist's intention and the listener's response. While we shall only explore a few of these ideas in some detail, the interested reader can find more thorough discussions in [37].

In the earlier forms of sonification, the data itself tended to be the most prominent feature. Indeed, sonification was viewed as an offshoot of data visualization. Thus, for example, the technique of audification, in which data values are mapped directly

to sound pressure levels, by maintaining a literal connection between the data and the sound, serves as a direct form of sonification. Audification has been used in especially for natural phenomenon that generate time series, such as seismology, or stock market prices. Parameter mapping, by contrast, is more of a second-order technique in which one parameter of the data is mapped to a parameter of sound, such as pitch, or volume. For example, a series of data representing temperature values might be mapped to pitch in such a way that higher temperature sounds as higher pitch, and vice versa. Although the data is not literally being interpreted as a waveform, the analogy of human perception still makes the connection between the data and sound tangible. Finally, more abstract forms of sonification, such as model-based techniques, in which a time-varying process is devised that connects the data and sound together, can strain the perceptual limits of the link between the data and the sound, although formally speaking, it still is present. Kramer proposes a "semiotic" spectrum from analogic to symbolic sonifications, with audification at the extreme analogic end as closest to the data, parameter mapping in the middle, and model-based techniques at the symbolic extreme [31].

2.1.2 Model-Based Sonification

As our particular research strategy resembles the category of model-based sonification (MBS) the most closely, we review this technique in some detail. According to Hermann in [31], "Model-Based Sonification is a sonification technique that takes a particular look at how acoustic responses are generated in response to the user's actions, and offers a framework to govern how these insights can be carried over to data sonification. As a result, Model-Based Sonification demands the creation of processes that involve the data in a systematic way, and that are capable of evolving in time to generate an acoustic signal." The latter quality of evolving in time

to generate an acoustic signal interests us the most, as the time evolution in such models is often governed systematically according to physical principles, making it an ideal candidate for a sonification of a physical process such as fluid dynamics. After exploring some fundamental ideas in human perception and modes of listening, Hermann concludes that a successful sonification should satisfy five properties, paraphrased thusly:

- Ubiquity: Each interaction with data should produce a sound.
- Invariance of binding mechanism: The laws producing the sound from the data should not depend on the data itself, much in the same way that the laws of physics do not depend on the specific objects that they govern.
- Immediate response: Each interaction with the data should produce an immediate response in order to mirror as closely as possible interactions with objects in the real world
- Sonic variability: Sonifications should vary according to subtle changes in state and input.
- Information richness: Sonifications should be nontrivial.

Essentially, these properties guarantee that a sonification mimics interactions in the real world with data and sound. This ties in with a more analytical mode of listening—for instance, shaking a wrapped birthday present to try to determine its contents from the corresponding sound, rather than listening to the sound of the jostling present as a musical event. By using the technique of MBS, Hermann concludes that these properties will in fact automatically be satisfied. The basic principle, thus, is to design something analogous to the laws of physics, but as a sonification system of data. The model then becomes the rules by which the data is mapped to

the sound and unfolds over time, much as the laws of physics determine the rules by which objects interact with one another in space and time.

Hermann also identifies six key components that aid in the design of a MBS system: setup, model dynamics, excitation, initial state, link-variables, and listener characteristics. It is also useful to keep as separate abstract concepts the data space in which the data lives, the model space in which the sonification "laws" live, the sound space in which the actual sound itself lives, and the listener space in which the listener's reaction to the sound lives. The model setup, thus, takes data from the data space, and maps it to the space of a dynamical model with time-varying components that create the corresponding sound in the sound space.

As a general example, suppose that the data can be interpreted as a collection of vectors $\{\mathbf{x}_j\}, j = 1, \ldots, N$, living in the d-dimensional vector space $\mathbb{R}^d$. (For instance, more concretely, the data might comprise the temperature, barometric pressure, and humidity in Santa Barbara for each day in the year 2017, meaning $d = 3$ and the collection has $N = 365$ elements.) Such a re-interpretation of the data abstractly in a mathematical space is the primary goal of the model setup. The advantage of such an approach is the ubiquity of mathematical tools in linear algebra at our disposal if we view our data through the lens of a vector space.

Given our model setup, we next turn to the model dynamics. Since our goal is to drive a sound-based model over time, it is natural to consider a time-varying evolution to our model. In other words, we consider the model not as a static configuration but as a dynamical system. Mathematically speaking, if we write $\mathbf{s}(t)$ to denote the state of the model at time t, then we would like a differential equation to govern the time evolution—e.g.,

$$\frac{d\mathbf{s}}{dt} = f(\mathbf{s}(t), t) \tag{2.1}$$

The function f here determines the model dynamics. Already the connection with physics is tempting. Although the function f in principle is arbitrary, the ideas of conservation of energy, gravity and buoyancy, friction, dissipation, and other physical principles provide an attractive palette from which to borrow.

The excitation can be thought of as analogous perturbations to the balance of forces in a dynamical system. In a general dynamical system, an excitation might be obtained through interaction, as a user adds external forces to the physics through a keystroke or mouse click. For example, in a fluid simulation, a user might add a force to the fluid velocity field by clicking and dragging the mouse through a region. Other possible fluid examples include altering the fluid's viscosity or introducing a buoyant external force.

The model setup is analogous to the initial and boundary conditions of a dynamical system. For example, a typical first-order ordinary differential equation $\frac{dx}{dt} = f(x(t), t)$ defines an entire family of possible solutions $x(t)$. Only by specifying an initial condition $x(0) = x_0$ do we uniquely determine which trajectory to take. More general differential equations, in addition to requiring initial conditions, may also require boundary conditions to determine edge behaviors. For instance, in a fluid simulation, we frequently use the "no-slip" boundary condition, meaning that at a solid boundary, the fluid has zero velocity relative to the boundary [38]. These parameters can greatly influence the resulting simulation, but the designer typically through experience and experimentation has knowledge of which conditions to use in order to obtain the desired physical effect.

Link variables form the bridge between the dynamical process of the model and actual sound. In the physical world, these connections often happen literally. For instance, if we model the rigid vibrations of a metal plate using a spring-mass system, the corresponding air pressure variations it induces lead directly to an audible

sound signal. Naturally, this isn't so much of a sonification as an actual physical calculation of sound, but the analogy remains useful for more abstract sonifications. In practice, link variables are often more of a second-order connection, much as in parameter- mapping sonification. For example, we might map the overall energy of a system to a musical loudness. A common challenge with MBS rears its head here, as depending on the complexity of the model, a real-time sound output may not be feasible. In particular, if the link variables are conceived of at an audio sample rate, in order to maintain high-quality sound rendering, we require at least 44100 samples to be computed per second, which may be computationally infeasible. Indeed, the audiovisual system proposed in this dissertation cannot be run in real-time using the available computing power as of writing in 2017. The tradeoffs of real-time vs. non-real-time composition will be discussed further in Chapter 6.

Listener characteristics aim to understand better the desired mode of listening for which the composer aims. For example, shall the listener perceive herself to be "inside" the system, or an outside observer? Should the sound be perceived as a single source, or an entire soundscape? Many of these questions have design answers in the form of the spatialization of the audio output. For instance, by using careful panning techniques, the composer can create many different desired perceptions in the listener.

2.1.3 Parameter Mapping Sonification

Following Grond and Berger in [31], Parameter Mapping Sonification (PMSon) is another important technique for sonification. It is a step removed from the more literal form of audification, in which the raw data is mapped directly into an audio-rate signal. Instead, PMSon is characterized by a mapping of abstract information to auditory parameters. For instance, we might map the value of a stock, as a piece of

abstract information, to a frequency of a sound, as an auditory parameter. While more indirect than audification, working more by analogy than by literal mapping, PMSon can be an especially useful choice for multidimensional data, in which different facets of the data can then be mapped to different facets of the audio.

Many challenges arise in PMSon. For instance, without the direct interpretation offered by audification, there is no one 'correct' way to make mapping associations. Indeed, in most PMSon systems, a wide range of interpretation is possible, even allowing for multiple different sonifications of the same underlying data. Another challenge is that of human perception: many data sets will generate output signals that exceed the limits of human hearing, either through resolution (just-noticeable differences) or through bandwidth (absolute thresholds). Thus, any practically useful mapping must take these limitations into effect, generally by adapting or scaling the data accordingly. The choice of which audio features to use remains an open question as well, as there are many from which to choose: pitch, timbre, rhythm, texture, envelope, duration, etc.

Formalization via Transfer Function

As a mathematical formalism for speaking in general terms about PMSon, consider a d-dimensional data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_j \in \mathbb{R}^d$. Let the sound parameter space have n dimensions, so it is identified with $\mathbb{R}^n$. (It needn't be the case that $d = n$ in general, hence the use of multiple variables.) Next, define $g\colon \mathbb{R}^d \to \mathbb{R}^n$ to be the parameter mapping function that maps an element in the data space to an element in the m-dimensional sound parameter space. Finally, define $f\colon \mathbb{R}^{n+1} \to \mathbb{R}^q$ to be the signal generation function, which takes as an input an n-dimensional vector $\mathbf{a}$ of sound parameters and a time t, and outputs a q-channel sound signal $s(t) = f(\mathbf{a}; t)$. A PMSon is then computed by

$$s(t) = \sum_{j=1}^{N} f(g(\mathbf{x}_j); t) \tag{2.2}$$

The mapping function g is the heart of the system. There are several different types of mappings: one-to-one, one-to-many, and many-to-one. In a one-to-one mapping, a single data point is mapped to a single sound point. For example, a temperature reading could be mapped to a sound frequency. In a many-to-one mapping, a form of dimension reduction occurs in the data. As a simplified example, an average of many temperature readings could be mapped to a single sound frequency. Forms of data reduction such as principle component analysis (PCA) are commonly used, which forms a nice bridge between sonification design and the computer graphics subspace approach we use in our physics-based simulations. Finally, one-to-many mappings can arise when a single data point maps to several different sound parameters. For instance, if a data value of temperature is in a certain region, we might map it to both sound frequency and sound amplitude. The multitude of possibilities again raises many questions and challenges for the appropriate design of PMSon systems.

2.1.4 Synthesis Methods

In designing any sonification system, the craft of audio synthesis itself is an important ingredient. Many different techniques exist for generating digital sound. We explore just a handful of possibilities in this section, following Cook in [31].

The technique of additive synthesis is perhaps the most basic and fundamental. The general strategy is attributed to the concepts of harmonic analysis. According to the theory of Fourier series [39], we know that any periodic signal $f(t)$ with period L can be decomposed into a Fourier series via the equation

$$f(t) = \sum_{n \in \mathbb{Z}} c_n e^{i\frac{2\pi}{L}nt} \tag{2.3}$$

If we write ω_0 as the frequency $\frac{2\pi}{L}$, then we see that f has been decomposed into a superposition of differently-weighted complex exponentials whose frequencies are integer multiples of ω_0. We often refer to ω_0 as the *fundamental frequency*.

The technique of additive synthesis is thus to take only pure sinusoidal tones and mix them together, building up from a fundamental frequency. The higher frequency tones in the signal, called the *partials*, may or may not have an integer ratio to the fundamental, characterizing the perception of the sound as harmonic or inharmonic. Besides the ratio between the partial and the fundamental, the time-varying amplitude envelope of each partial also determines an important synthesis parameter.

Another important technique of synthesis, modal synthesis, is of particular interest. Modal synthesis is based on the concept of a simple harmonic oscillator with damping. The solution to this differential equation is known to be a cosine wave, dampened by a decaying exponential term. More precisely, the system in question, generated from a simple application of Newton's second law $F = ma$, is given by

$$m\ddot{y} = -c\dot{y} - ky \tag{2.4}$$

or, rearranging,

$$\ddot{y} + \frac{c}{m}\dot{y} + \frac{k}{m}y = 0 \tag{2.5}$$

Here, the unknown $y = y(t)$ represents displacement from the equilibrium, m is the mass, c is the damping term, k is the spring constant, and an overdot denotes a time derivative. The solution is given by

$$y(t) = y_0 e^{-\frac{c}{2m}t} \cos\left(t\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2}\right) \tag{2.6}$$

While these equations are a bit abstract, they apply directly to simple idealized musical instruments. For example, the vibrational modes of a plucked string can be determined through such an analysis. The boundary conditions of the string (assuming it is taut and pinned down at both ends) force the string into a particular set of modal vibration shapes that correspond to natural frequencies of the string. In a slightly more sophisticated example, the membrane of a drumhead can also be excited into forming a set of modal vibration shapes that correspond to its natural frequencies. These frequencies again will depend on the boundary condition (how the drum is pinned down) as well as the geometric shape of the drumhead.

Cook presents an interesting interpretation of the dampened harmonic oscillator as a second order, 2-pole feedback filter. If we interpret Equation 2.5 using finite-difference approximations for the derivatives, we obtain

$$\frac{y[N] - 2y[N-1] + y[N-2]}{T^2} + \frac{c}{m}\frac{y[N] - y[N-1]}{T} + \frac{k}{m}y[N] = 0 \tag{2.7}$$

Here, $y[N-k]$ denotes the value of the sample k samples ago. Gathering together like terms, we obtain

$$y[N]\frac{m + cT + kT^2}{m} - y[N-1]\frac{2m + cT}{m} + y[N-2] = 0 \tag{2.8}$$

Solving for $y[N]$ in terms of the previous samples, we get

$$y[N] = c_1 y[N-1] + c_2 y[N-2] \tag{2.9}$$

where $c_1 = \frac{2m+cT}{m+cT+kT^2}$ and $c_2 = \frac{m}{m+cT+kT^2}$. In the Digital Systems Processing (DSP)

literature [40], Equation 2.9 can be implemented using a second order 2-pole feedback filter. Thus, we can interpret modal synthesis as a form of subtractive synthesis, in which a spectrally rich input source, such as noise, or an impulse, excites modal filters into resonance.

2.1.5 Aesthetics

A continual dilemma in generative art is the conflict between the level of rigor of the underlying formal system and the human perception of its output. Some artists (e.g. Milton Babbitt) take the dogmatic position that the logic of the system trumps the general perception of the output [41]. However, others such as Křenek have taken a more careful middle ground, arguing in [42] that the existence of an aesthetically coherent system of rules is no guarantee that an aesthetically coherent artistic result will perceptibly emerge: "We cannot take the bare logical coherence of a musical 'axiomatic' system as the sole criterion of its soundness! ... The outstanding characteristic of music [is] its independence from the linguistic limitations of general logic." In this dissertation, our attitude aligns more closely to Křenek than to Babbitt. While the rigor of the underlying system is an important concern, the sonic output still must be considered carefully and adjusted as necessary. To that end, our freedom of choice in the sonification strategy allows us to make choices that emphasize musical qualities such as texture, timbre, and dynamic control while still adhering to the logic of the system. We view our system as a high-level tool through which novel sounds and visuals can be generated automatically, evolving together over time. Aesthetically, the visual and sonic impression on the viewer/listener is an important criterion which cannot be ignored by appealing to the logic of the system. As a basic example of applying that principle in our work, we selected a visually interesting training set of fluid motions so that the subspace dynamics retained a variety of pleasing forms

and shapes. A more in-depth discussion of our aesthetic principles as applied to our audiovisual system follows in Chapter 6.

Chapter 3

Background

3.1 Fluids

A *fluid* is a substance that continuously deforms under the application of shear stress [43]. More intuitively, it is a substance that *flows*. Although colloquially we may use the term fluid to refer only to liquids, gases can exhibit flow, and are therefore considered to be fluids as well. (Indeed, the results demonstrated in this dissertation focus entirely on the turbulent flow of smoke.) The natural world is full of captivating fluid motions: ocean waves, ripples in a pond, clouds, the steam of a teakettle, etc. These flowing patterns, while beautiful, appear difficult to formalize precisely, and the modeling of fluids continues to be a mathematical and computational challenge to this day.

3.2 Physics-based modeling

The field of *fluid dynamics* in physics seeks to model and describe fluid flow systematically according to equations generated from a set of basic assumptions and

principles. In order for continuous mathematical operators to be used to describe fluids, one of the basic assumptions is that of the *continuum*: for mathematical purposes, we assume that fluid velocity varies continuously across space, even though in reality, fluids are actually composed of discrete molecules. Besides this continuum assumption, the various conservation laws of mass, energy, and momentum, imply the famous Navier-Stokes equations [44], which describe the velocity field $\mathbf{u}$ of fluid flow depending on the intrinsic stress of viscosity and pressure. In the case where we make the additional simplifying assumption that the fluid is incompressible,[1] the equations take the following form:

$$\nabla \cdot \mathbf{u} = 0 \tag{3.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f}_e \tag{3.2}$$

The incompressibility assumption is violated for fluid dynamics such as supersonic flows; however, these types of motions are not the dynamics we seek to model in this dissertation. There are several terms here that must be unpacked:

- The time-varying vector field $\mathbf{u} = u(\mathbf{x}, t)$ describes the fluid velocity field. It is the main quantity we are interested in, and solving these differential equations amounts to describing what $\mathbf{u}$ is for all times t.
- The constant ν represents the viscosity of the fluid, which represents its resistance to deform while flowing. (Intuitively, this can be thought of its resistance to being stirred: molasses, for instance, has higher viscosity than water.)
- The time-varying scalar field $p = p(\mathbf{x}, t)$ represents the pressure field of the fluid.

[1] i.e., the fluid density does not vary within the flow field

- The time-varying vector field $\mathbf{f}_e = \mathbf{f}_e(\mathbf{x}, t)$ represents any external forces affecting the fluid, such as gravity.

Equation 3.1 enforces the assumption of incompressibility, while equation 3.2 is a consequence of the aforementioned conservation laws.

3.3 Simulation

Simulation is the imitation of a natural process, in this case over time. Any simulation must first determine a technique of modeling the process computationally, and there are often many different possible strategies. We consider here a few typical approaches in computational fluid dynamics.

As in all numerical techniques of modeling fluids, we first discretize both the spatial and time domain. In other words, to perform the calculations, we dice up the region of space into a regular grid of small cubical cells. Each cell then contains a vector, or arrow, describing the velocity of the fluid at the position in space. In order to evolve the motion of the flow over time, we also dice up time itself into a sequence of discrete time steps, computing how the velocity of the fluid in each cell changes at each time step. This approach is called the *Eulerian* viewpoint of the flow. Intuitively, this can be thought of as staying in a fixed location and observing the flow through that particular location. More precisely, the fluid's velocity in this viewpoint is represented as $\mathbf{u}(\mathbf{x}, t)$, which gives the velocity of the fluid at each spatial location $\mathbf{x}$ and time t.

In contrast, the *Lagrangian* viewpoint of the flow takes the point of view of each individual fluid particle as it moves along the flow through space and time. Intuitively, this can be thought of drifting along with the flow. If we label each particle based on its position in space using a vector field $\mathbf{r}$, we then describe the fluid flow

with the position field $\mathbf{X}(\mathbf{r}, t)$, which gives the position of each particle labeled by $\mathbf{r}$ at time t.

Following [45], we can connect these two viewpoints with the following equation:

$$\mathbf{u}\left(\mathbf{X}\left(\mathbf{r}, t\right)\right) = \frac{\partial \mathbf{X}}{\partial t}\left(\mathbf{r}, t\right) \tag{3.3}$$

The left-hand side of equation 3.3 describes the velocity of the flow at position $\mathbf{r}$ and time t using the Eulerian-specified velocity field $\mathbf{u}$, while the right-hand side describes the same velocity by taking the partial derivative of the Lagrangian specified position field $\mathbf{X}$.

Given an Eulerian-specified fluid velocity field $\mathbf{u}(\mathbf{x}, t)$ over a domain of space, and given another Eulerian-specified scalar field $\varphi(\mathbf{x}, t)$ (e.g., temperature) over the same domain, we can consider the total derivative of φ. By the chain rule, this expands as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi\left(\mathbf{x}, t\right) = \frac{\partial \varphi}{\partial t} + \dot{\mathbf{x}} \cdot \nabla \varphi \tag{3.4}$$

The partial derivative term $\frac{\partial \varphi}{\partial t}$ indicates the instantaneous rate at which the temperature is changing with time, holding the spatial position constant. Thus, if we imagine a fluid flow being comprised of many infinitesimal particles, this term represents any change in temperature of a particle at a fixed spatial location. The remaining term $\dot{\mathbf{x}} \cdot \nabla \varphi$ describes a more dynamic rate of change, as it incorporates changes in temperature caused by moving *along* the flow through the path $\dot{\mathbf{x}}$. This movement is in the Lagrangian spirit of drifting along with the flow. In the case where $\dot{x}$ is taken to align with the flow velocity $\mathbf{u}$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi\left(\mathbf{x}, t\right) = \frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi \tag{3.5}$$

which is often called the *material derivative*. The operator $\mathbf{u} \cdot \nabla$ is the advection operator, which we will be prominent later on.

3.3.1 Numerical Simulation

Numerical simulation brings the abstract equations of mathematical simulation to a concrete, computational implementation. Typically, this involves discretization of any continuous parameters in the model, such as time and space. As such, idealized concepts such as spatial and time-dependent derivatives must be approximated. Numerical analysis is a broad and rich field in applied mathematics and engineering, and many different approaches are available, each with various pluses and minuses. For the scope of this dissertation, we consider the numerical method of Jos Stam's *Stable Fluids* [46], which is the backbone of our fluid simulation technique.

The first step in Stam's method, following the technique of Chorin and Marsden in [47], is to devise a technique to combine the two equations 3.1 and 3.2 into one equation. We rely on a result from vector calculus called the Helmholtz-Hodge decomposition, which states that any arbitrary vector field $\mathbf{w}$ can be decomposed uniquely into the sum of a divergence-free, or solenoidal, vector field and a curl-free, or irrotational, vector field. More precisely, we can write

$$\mathbf{w} = \mathbf{u}_{\text{solenoidal}} + \mathbf{u}_{\text{irrotational}} \tag{3.6}$$

$$= \mathbf{u}_{\text{solenoidal}} + \nabla q \tag{3.7}$$

since the curl $\nabla \times \nabla q$ must be zero for some scalar field q. Hence, we can define a projection operator P which maps a vector field to its (unique) solenoidal component. In other words, $P\mathbf{w} = \mathbf{u}_{\text{solenoidal}}$. Indeed, this operator can be discovered explicitly

by taking the divergence of both sides:

$$\nabla \cdot \mathbf{w} = \nabla^2 q \tag{3.8}$$

which is a Poisson equation for the scalar field q. Given $\mathbf{w}$, we can solve this equation to obtain q, and then compute $\mathbf{u}_{\text{solenoidal}}$ via $\mathbf{u}_{\text{solenoidal}} = \mathbf{w} - \nabla q$. Thus, we can take the second equation in the Navier-Stokes equations, 3.2, and apply the projection operator P to both sides, obtaining

$$\frac{\partial \mathbf{u}}{\partial t} = P\left(-(\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}_e\right) \tag{3.9}$$

This conveniently eliminates the pressure term ∇p and unites the two equations 3.1 and 3.2 into one.

The next important technique that Stam's method uses involves *splitting* the operators in the Navier-Stokes equations 3.9 into a sequence of simpler operators. In other words, because the right- hand-side of 3.9 can be regarded as a combination of several terms (namely, an advection term, a diffusion term, an external forces term, and a projection) 3.10, we focus our attention on solving each of these simpler differential equations in isolation. Once we have a solution to each individual term, we can construct an approximation for the solution to the more complex full equation 3.9. Establishing some notation, suppose we begin from an initial state $\mathbf{u}_0 = \mathbf{u}(\mathbf{x}, 0)$ and would like to step forward in time by a timestep Δt. Beginning from the previous timestep's solution $\mathbf{w}_0 = \mathbf{u}(\mathbf{x}, t)$, we solve a sequence of equations $\mathbf{w}_0, \ldots, \mathbf{w}_4$, ending with the last velocity field which is equal to the value at the next time step: $\mathbf{w}_4(\mathbf{x}) = \mathbf{u}(x, t + \Delta t)$. Iterating this procedure generates a complete numerical solution, as desired.

$$\mathbf{w}_0(\mathbf{x}) \xrightarrow{\text{add forces}} \mathbf{w}_1(\mathbf{x}) \xrightarrow{\text{advect}} \mathbf{w}_2(\mathbf{x}) \xrightarrow{\text{diffuse}} \mathbf{w}_3(\mathbf{x}) \xrightarrow{\text{project}} \mathbf{w}_4(\mathbf{x}) \tag{3.10}$$

The external forces can be approximated simply using forward Euler or other explicit schemes. In other words, we can write $\mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}_{\text{ext}}(\mathbf{x}, t)$. The advection term, however, is nonlinear and adds considerable complexity to the solution. Stam's insight was to consider the advection from a Lagrangian viewpoint. Intuitively speaking, each fluid particle is advected by its own velocity. Hence, to calculate the velocity at a particular spatial point $\mathbf{x}$ at the next time step $t + \Delta t$, we backtrace from the point $\mathbf{x}$ through the previous velocity field over a time step of Δt. Letting $\mathbf{x}$ vary, following Stam's notation, this defines a path $\mathbf{p}(\mathbf{x}, s)$ along a partial streamline of the velocity field. We assume then that the new velocity at the point $\mathbf{x}$ at time $t + \Delta t$ must therefore be the same as the old velocity at the backtraced point at time Δt previous. (Although the backtraced point may not lie exactly on a grid cell, we can use linear interpolation to resolve this problem.) More concretely, we have $\mathbf{w}_2(\mathbf{x}) = \text{Interpolate}\left(\mathbf{w}_1\left(\mathbf{p}(\mathbf{x}, -\Delta t)\right)\right)$. Because this approach exploit both the Eulerian and Lagrangian viewpoints, it is known as a *semi-Lagrangian* strategy.

One of the key advantages to semi-Lagrangian advection technique is that it is *unconditionally stable*: in other words, no matter how large a time step Δt we select, the velocity will remain bounded. Intuitively, this is straightforward to observe: each velocity update is the result of an interpolation of previous velocity values, and therefore cannot be any greater than any of the previous values. From a practical standpoint, this numerical stability is a big plus, and the heart of Stam's argument as to why this method is useful in computer graphics. However, with very large time steps, the problem of numerical dissipation becomes more of an issue. A more detailed discussion of these issues can be found in [48].

The remaining terms in the splitting are the diffusion and pressure projection. The diffusion term is simply the heat equation, relying on the Laplace operator:

$$\frac{\partial \mathbf{w}_2}{\partial t} = \nu \nabla^2 \mathbf{w}_2 \tag{3.11}$$

Many well-known techniques exist for numerical solutions to this equation, including discretization of the Laplace operator as well as implicit methods [49].

Finally, as observed previously, the pressure projection reduces to a Poisson equation to solve for q from $\mathbf{w}_3$.

$$\nabla^2 q = \nabla \cdot \mathbf{w}_3 \tag{3.12}$$

From here, we can recover $\mathbf{w}_4$ via the equation $\mathbf{w}_4 = \mathbf{w}_3 - \nabla q$. Hence, the numerical splitting is complete.

Within Stam's paper is another interesting topic of carrying out the equations of motion within the Fourier domain, explored further in [50]. Although we do not implement our solver using this technique, it is a topic of interest based on our compression technique as well as, our frequency-domain interpretation of the fluid modes and our corresponding sonification technique later on, so we delve into a few details here. The key observation is that the Fourier transform diagonalizes spatial derivatives. In other words, if we denote the Fourier transform of $\mathbf{u}(\mathbf{x})$ by $\mathcal{F}[\mathbf{u}(\mathbf{x})](\mathbf{k}) = \hat{\mathbf{u}}(\mathbf{k})$, then we have $\mathcal{F}[\nabla \mathbf{u}(\mathbf{x})](\mathbf{k}) = i\mathbf{k}\hat{\mathbf{u}}(\mathbf{k})$. The divergence operator similarly satisfies $\mathcal{F}[\nabla \cdot \mathbf{u}(\mathbf{x})](\mathbf{k}) = i\mathbf{k} \cdot \hat{\mathbf{u}}(\mathbf{k})$, and the Laplacian, being the divergence of the gradient, satisfies $\mathcal{F}[\nabla^2 \mathbf{u}(\mathbf{x})](\mathbf{k}) = -k^2 \hat{\mathbf{u}}(\mathbf{k})$, where $k = |\mathbf{k}|$. The upshot of all this is that we can view some of the fluid operators in the spatial domain in simple ways in the frequency domain. For example, the diffusion, based on the Laplacian operator, can be thought of as a low-pass filter with decay governed by the

viscosity. In general, this mode of thinking bears fruit later for our discussions of the compression algorithm in Chapter 4 and our sonification system in Chapter 5.

3.4 Subspace Methods

Subspace methods, also known as model reduction, or reduced order methods, has a long history in engineering and applied mathematics, including applications to fluid simulation [51], and was introduced to the computer graphics literature in 1989 by Pentland and Williams [28, 52]. The basic principle of these methods, as the name suggests, is to reduce the computational complexity of a large numerical simulation. A typical simulation will have many degrees of freedom in principle, generating a vast state space of possibilities. However, in practice, not all of these degrees of freedom are of equal importance. By systematically discovering a reduced subspace of the state space, and carrying out calculations within this subspace, a subspace simulation can lead to large computational accelerations with minimal degradation of accuracy.

3.4.1 Modal Analysis

Following Barbič and James [53], to see an example of the subspace method in action, consider the modeling of small deformations of rigid bodies. Although rigid bodies in principle have many degrees of freedom over which they can deform, they have certain characteristic shapes into which they are more likely to deform. These shapes, also known as *modes*, are the ones which minimize the strain, and in an intuitive sense are the "natural" deformations of the rigid body. To compute these, we assume the rigid body is discretized into a mesh with N vertices. We then compute the system mass and stiffness matrices $\mathbf{M} \in \mathbb{R}^{3N \times 3N}$ and $\mathbf{K} \in \mathbb{R}^{3N \times 3N}$, respectively. The modes

then satisfy the generalized eigenvalue equation

$$\mathbf{K}\mathbf{x} = \lambda\mathbf{M}\mathbf{x} \tag{3.13}$$

Due to the special properties of the matrices $\mathbf{M}$ and $\mathbf{K}$, which are both symmetric positive-definite (SPD), the eigenvalues λ_i are positive real numbers. Although there are $3N$ eigenvalues in total, in practice, we can take a subset of them, $\lambda_1, \lambda_2, \ldots, \lambda_r$, from least to greatest, and their associated eigenvectors $\psi_1, \psi_2, \ldots, \psi_r$. (Here, $r \ll 3N$ represents the number of modes we wish to retain.) The eigenvectors ψ_i we choose corresponding to the eigenvalues λ_i are not unique; for convenience, we select those that satisfy the mass-orthogonality condition $\langle \mathbf{M}\psi_i, \psi_j \rangle = \delta_{ij}$, where δ_{ij} is the Kronecker delta, equaling 0 for $i \neq j$ and 1 for $i = j$. The r-dimensional subspace $S \subset \mathbb{R}^{3N}$ formed by the linear span of these eigenvectors can be encapsulated in matrix form by assembling the modal basis matrix $\mathbf{U} \in \mathbb{R}^{3N \times r}$ as follows:

$$\mathbf{U}^T = \begin{pmatrix} | & | & & | \\ \psi_1 & \psi_2 & \ldots & \psi_r \\ | & | & & | \end{pmatrix} \tag{3.14}$$

Thus, the mass-orthogonality condition we specified for the modes can be captured in the more succinct matrix form $\mathbf{U}^T\mathbf{M}\mathbf{U} = \mathbf{I}_r$, where $\mathbf{I}_r \in \mathbb{R}^{r \times r}$ denotes the $r \times r$ identity matrix.

Now consider the typical equation of motion for undamped small deformations

$\mathbf{x} \in \mathbb{R}^{3N}$ corresponding to external forces $\mathbf{f} \in \mathbb{R}^{3N}$:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f} \tag{3.15}$$

Equation 3.15, is a very high-dimensional linear differential equation obtained by applying the standard equations of elastic deformation to the mesh with $3N$ nodes using the finite element method (FEM). For very fine meshes with large values of N, this equation may present a computational bottleneck. Hence, the approach of model reduction is welcome. We proceed by approximating $\mathbf{x}$ as a linear combination of the modes $\psi_1, \ldots, \psi_r$:

$$\mathbf{x} \approx \mathbf{U}\mathbf{q} \tag{3.16}$$

where the vector $\mathbf{q} \in \mathbb{R}^r$ has components q_i that are the corresponding weights of the associated modes ψ_i, $i = 1, \ldots, r$. This vector $\mathbf{q}$ is called the *reduced coordinates* of $\mathbf{x}$, and can also be thought of as the projection of $\mathbf{x}$ into the subspace S. Rewriting equation 3.15, we obtain

$$\mathbf{M}\mathbf{U}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{U}\mathbf{q} = \mathbf{f} \tag{3.17}$$

By the generalized eigenvector condition, we have $\mathbf{K}\mathbf{U} = \Lambda\mathbf{M}\mathbf{U}$, where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_r)$. Hence, we simplify as follows:

$$\mathbf{M}\mathbf{U}\left(\ddot{\mathbf{q}} + \Lambda\mathbf{q}\right) = \mathbf{f} \tag{3.18}$$

Next, to clear away the factor of $\mathbf{M}\mathbf{U}$, we pre-multiply both sides by $\mathbf{U}^T$:

$$\mathbf{U}^T\mathbf{M}\mathbf{U}\left(\ddot{\mathbf{q}} + \Lambda\mathbf{q}\right) = \mathbf{U}^T\mathbf{f} \tag{3.19}$$

By the mass-orthogonality condition, the term $\mathbf{U}^T\mathbf{M}\mathbf{U}$ reduces to $\mathbf{I}_r$. On the right-hand-side, $\widetilde{\mathbf{f}} = \mathbf{U}^T\mathbf{f} \in \mathbb{R}^r$ is the reduced form of $\mathbf{f}$. Hence, we have the following lower-dimensional linear differential equation with unknown $\mathbf{q} \in \mathbb{R}^r$:

$$\ddot{q} + \Lambda\mathbf{q} = \widetilde{\mathbf{f}} \tag{3.20}$$

In this particular case, the system is uncoupled since Λ is diagonal, so we have a collection of r independent one-dimensional ordinary differential equations. These can be integrated forward in time very efficiently, obtaining $\mathbf{q}_{t+1}$, the value of $\mathbf{q}$ at the next time step. To finish the computation, we reconstruct into the full-dimensional space by computing $\mathbf{x}_{t+1} = \mathbf{U}\mathbf{q}_{t+1}$.

3.4.2 Sound generation

Small rigid deformations of a solid body are precisely those that generate sound waves. Hence, following O'Brien [54], we briefly consider the application to audio of the previous discussion. Recall that our subspace reduction produces a collection of r independent one-dimensional equations, which, expanded out, take the form

$$\ddot{q}_i + \lambda_i q_i = \psi_i^T f_i, \quad i = 1, \ldots, r \tag{3.21}$$

Recalling that each λ_i is a positive real number, let ω_i be the positive square root $\omega_i = \sqrt{\lambda_i}$. Then the general analytical solution to the homogenous portion of each one-dimensional differential equation in 3.21 is given by $q_i = c_1 e^{i\omega_i t} + c_2 e^{i\omega_i t}$. In other words, we can think of the i-th mode as resonating at a natural frequency of ω_i. This allows us to perform simple additive synthesis by mixing together a collection of sinusoids at the corresponding frequencies, generating the audio signal that corresponds to the physical deformation. The spirit of this basic approach motivates our

sonification process in Chapter 5.

3.4.3 A more general subspace formulation

We have seen how subspace methods apply to rigid deformations, which can be characterized by linear systems. However, more general and complex phenomena, including fluid flow, require non-linear characterizations. Hence, we consider a more general framework for applying subspace methods, following Treuille [21].

In the most abstract setting, the subspace technique takes a vector $\mathbf{u} \in \mathbb{R}^N$ living in a high-dimensional space and maps it to a corresponding reduced-space vector $\mathbf{q} \in \mathbb{R}^r$, with $r \ll N$. (We refer to this high-dimensional space as the "full space" and the reduced space as the "subspace.") This mapping can be seen as a projection operator $P : \mathbb{R}^N \to \mathbb{R}^r$, which carries $\mathbf{u}$ to $\mathbf{q}$, and its corresponding reconstruction operator $P^{-1} : \mathbb{R}^r \to \mathbb{R}^N$, which lifts $\mathbf{q}$ back up to $\mathbf{u}$. (Note that since $r < N$, the projection map P is not one-to-one, and hence necessarily must lose some information.)

Within physical simulations, we are not concerned just with the projection mapping of one vector in the high space down to its corresponding reduced vector, but also a framework for time evolution. In other words, given a differential equation in the full space

$$\dot{\mathbf{u}} = F(\mathbf{u}, t) \tag{3.22}$$

we would like to formulate a corresponding reduced-space differential equation

$$\dot{\mathbf{q}} = \hat{F}(\mathbf{q}, t) \tag{3.23}$$

so that the equations of motion can be integrated through time within the reduced space. In particular, we desire a reduced-space differential equation that can be

solved in asymptotic time depending only on the reduced dimension r and not the full dimension N.

The obvious and standard approach is to compute $\hat{F}$ via what is known as *Galerkin projection* as follows:

$$\hat{F} = P \circ F \circ P^{-1} \tag{3.24}$$

In the situation of linear equations, this projection as described before can be described with matrix-vector multiplies. In particular, the matrix corresponding to $\hat{F}$ can be precomputed, leading to large speedups. However, nonlinear dynamics must be handled with more sophisticated methods. Furthermore, in the absence of analytical eigenmodes, as utilized during section §3.4.1, it is unclear how to select a set of basis vectors to form a reasonable reduced space $\mathbb{R}^r$ in the first place.

Treuille's technique is to form a representative basis using "snapshots" from a corresponding full-space fluid simulation. Ideally, these snapshots capture the user's desired range of motion and dynamics. Upon performing a form of Principal Component Analysis (PCA), a set of basis vectors $\mathbf{q}_1, \ldots, \mathbf{q}_r$ are derived. Following the basic strategy of splitting the Navier-Stokes equations, it remains then to demonstrate how to perform reduced-space external forces, advection, diffusion, and projection. While most of these operations can be viewed as a full-space matrix, and thus projected and precomputed, the advection term, being nonlinear, requires a more careful approach. Although in principle, by discretizing the advection operator, a matrix corresponding to the advection term could be formulated, this matrix will depend on the current timestep, and thus cannot be precomputed like the others. To combat this issue, the original approach of Treuille is to construct a static third order tensor, which can be precomputed. However, as Kim and Delaney point out, this technique

will not in general be a *consistent* integration method [55, 56]. The intuition for this problem is that the finite difference methods in the reduced space are not equivalent to the semi-Lagrangian methods in the full-space, but further details are beyond the scope of this dissertation. However, the need for a different strategy for advection and other nonlinear phenomena motivates the notion of cubature schemes, described next.

3.4.4 Cubature schemes

The work of An et al. [57] brought the idea of cubature methods to nonlinear solid mechanics in computer graphics. The term cubature itself is a multidimensional analogue of the classical one-dimensional problem of *quadrature*, which seeks to approximate the definite integral of a function by estimating it as a weighted sum of point-sampled versions of the function. More concretely, we have an approximation of the form

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{3.25}$$

where the w_i are weights and the x_i are the point samples. Careful choice of these weights and point samples leads to efficient and accurate estimation of the integral.

In the more general setting, following Kim and Delaney [55], we introduce some standard notation. Suppose our full space is $\mathbb{R}^{3N}$ and our subspace is $\mathbb{R}^r$, so that our subspace projection matrix is $\mathbf{U} \in \mathbb{R}^{3N} \times r$. Furthermore, we denote reduced-space quantities with an over-tilde, e.g., $\tilde{\mathbf{u}} = \mathbf{U}^T u \in \mathbb{R}^r$ denotes the projection by $\mathbf{U}$ of $\mathbf{u}$ in the full space down to $\tilde{\mathbf{u}}$ in the reduced space. Next, suppose we have a (possibly nonlinear) function $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}^3$ and a vector of N points in $\mathbb{R}^3$, $\mathbf{x} \in \mathbb{R}^{3N}$ that represents a velocity field on a grid. Since $\mathcal{F}$ can operate on any point $p \in \mathbb{R}^3$, we can

transform the entire vector $\mathbf{x}$ pointwise by having $\mathcal{F}$ operate on each of its points, at each stage computing $\mathbf{f}_p = \mathcal{F}_p(\mathbf{x}_p) \in \mathbb{R}^3$. This yields a corresponding vector $\mathbf{f} \in \mathbb{R}^{3N}$ that represents $\mathbf{x}$ pointwise transformed by $\mathcal{F}$. (Although the discussion here is generic to what $\mathcal{F}$ represents, to give a concrete example, $\mathbf{x}$ could be a velocity field prior to advection, $\mathcal{F}$ could be an advection scheme, and $\mathbf{f}$ could be the resulting velocity field after advection.)

In the full space, we have a mapping from $\mathbf{x}$ to $\mathbf{f}$ via $\mathcal{F}$, so in order for the subspace technique to work, we need to discover a mapping from $\tilde{\mathbf{x}}$ to $\tilde{\mathbf{f}}$. The simplest cases are when $\mathcal{F}$ itself is a linear transformation, so that its projection can be precomputed, but in the nonlinear case, as discussed previously, we need another strategy. Observe first that if we are willing to calculate with full-space coordinates, we can write

$$\tilde{\mathbf{f}} = \mathbf{U}^T\mathbf{f} = \mathbf{U}^T\mathcal{F}\mathbf{x} = \mathbf{U}^T\mathcal{F}(\mathbf{U}\tilde{\mathbf{x}}) \tag{3.26}$$

which implies a method from computing $\tilde{\mathbf{f}}$ starting from $\tilde{\mathbf{x}}$ by reconstructing first into the full space via $\mathbf{U}$, applying $\mathcal{F}$, and then projecting back down into the subspace. This method, however, is counter to the spirit of the subspace approach, as it depends on the dimension of the full space, N. The insight of Kim and Delaney is to regard Equation 3.26 as a multidimensional integral over the entire simulation domain Ω:

$$\tilde{\mathbf{f}} = \mathbf{U}^T\mathcal{F}(\mathbf{U}\tilde{\mathbf{x}}) = \int_\Omega \mathbf{U}_p^T\mathcal{F}_p(\mathbf{U}_p\tilde{\mathbf{x}})d\Omega \tag{3.27}$$

Now in integral, we can therefore apply the method of cubature [58] to obtain an efficient approximation of $\tilde{\mathbf{f}}$. In particular, we would like to select a set $\mathcal{S}$ of points $p \in \mathcal{S}$ and corresponding weights $w_p \in \mathbb{R}$ so that we have

$$\tilde{\mathbf{f}} = \int_{\Omega} \mathbf{U}_p^T \mathcal{F}(\mathbf{U}_p \tilde{\mathbf{x}}) d\Omega \approx \sum_{p \in \mathcal{S}} w_p \mathbf{U}_p^T \mathcal{F}(\mathbf{U}_p \tilde{\mathbf{x}}) \tag{3.28}$$

Provided that the cardinality $|\mathcal{S}|$ is on the order of the reduced space-dimension r rather than the full-space dimension N, this approach will pay dividends. However, it remains to see how to select both the cubature set $\mathcal{S}$ and the weights w_p.

The original approach by An is to select a set of cubature points and corresponding weights that minimize the error of a training set. First, we describe the method of choosing weights, given a particular set of cubature points. To that end, suppose we have a collection of T snapshots, $\mathbf{f}^1, \ldots, \mathbf{f}^T$, and their associated reduced-space projections $\tilde{\mathbf{f}}^1, \ldots, \tilde{\mathbf{f}}^T$, where $\tilde{\mathbf{f}}^i = \mathbf{U}^T \mathbf{f}^i$. In addition, suppose we have selected a collection of n cubature points. Then we look to solve the following nonnegative least squares (NNLS) problem:

$$\begin{bmatrix} \tilde{\mathbf{f}}_1^1 & \cdots & \tilde{\mathbf{f}}_i^1 & \cdots & \tilde{\mathbf{f}}_n^1 \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{f}}_1^t & \cdots & \tilde{\mathbf{f}}_i^t & \cdots & \tilde{\mathbf{f}}_n^t \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{f}}_1^T & \cdots & \tilde{\mathbf{f}}_n^T & \cdots & \tilde{\mathbf{f}}_n^T \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}^1 \\ \vdots \\ \tilde{\mathbf{f}}^t \\ \vdots \\ \tilde{\mathbf{f}}^T \end{bmatrix} \tag{3.29}$$

with the unknown weight vector $\mathbf{w}$ satisfying $\mathbf{w} \geq \mathbf{0}$. For convenience, we abbreviate equation 3.29 as $\mathbf{A}\mathbf{w} = \mathbf{b}$, with $\mathbf{A} \in \mathbb{R}^{rT \times n}$, $\mathbf{w} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^{rT}$. Each column of the matrix $\mathbf{A}$ represents subspace point-sampled versions of the advection operator of the snapshot $t \in [1 \ldots T]$ at the corresponding cubature point $i \in [1 \ldots n]$. The vector $\mathbf{w}$ represents the unknown nonnegative weights we seek. Finally, the vector $\mathbf{b}$ is a projected full-rank advection computed via equation 3.26. Schemes for solving the NNLS problem efficiently are well-known, including the standard Lawson-

Hanson method [59], which has $O(rTn^3)$ complexity, and the "fast" nonnegative least squares method (FNNLS) introduced by Bro and de Jong [60], which achieves a constant speedup over Lawson-Hanson. Provided that the number of cubature points n remains much smaller than the full-space resolution N, these timings are acceptable. We review two different techniques for selecting the cubature points: *greedy* selection and *importance sampled* selection.

Greedy Cubature Selection

We see from equation 3.29 that given a set $\mathcal{S}$ of cubature points, we can calculate the corresponding weights from the weight vector $\mathbf{w}$. However, the question remains of how to select the set $\mathcal{S}$ in the first place. The technique of greedy selection was introduced by An et al. in [57]. Suppose for the sake of argument we have already selected a set of cubature points, giving us a preliminary set $\mathcal{S}$, yet the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{w}$ has a significant magnitude $\|\mathbf{r}\|_2 > \epsilon$ greater than a desired tolerance. To decrease the error, we must add a new cubature point to $\mathcal{S}$, which extends the matrix $\mathbf{A}$ by appending a new column. This will then update the new residual and decrease the error. The "greedy" solution is to select a cubature point p whose corresponding column $\mathbf{a}_p$ maximally projects onto the residual $\mathbf{r}$; i.e., we wish to maximize the dot product $\mathbf{a}_p \cdot \mathbf{r}$. Once such a point p is found, we add it to our set $\mathcal{S}$ and update the residual. We repeat this process until the residual has a magnitude smaller than the desired tolerance ϵ. The overall complexity of the greedy approach is $O(rTn^4)$.

Importance Sampled Cubature Selection

The technique of importance sampled cubature selection was proposed by Kim and Delaney in [55]. In the greedy algorithm, because the matrices A_S between one iteration and the next differ only by one column, much of the work becomes redundant.

Algorithm 1 Greedy Cubature Selection

1: **function** GREEDYCUBATURE($\mathbf{A}$, $\mathbf{b}$, ϵ)
2: $\quad\mathcal{S} \leftarrow \emptyset$
3: $\quad\mathbf{r} \leftarrow \mathbf{b}$
4: $\quad$**while** $\|\mathbf{r}\|_2 > \epsilon$ **do**
5: $\quad\quad\mathcal{C} \leftarrow$ SELECTCANDIDATEPOINTS($\mathcal{S}$)
6: $\quad\quad p \leftarrow \arg\max_{p \in \mathcal{C}} \mathbf{a}_p \cdot \mathbf{r}$
7: $\quad\quad\mathcal{S} \leftarrow \mathcal{S} \cup \{p\}$
8: $\quad\quad\mathbf{w} \leftarrow \text{NNLS}(\mathbf{A}_\mathcal{S}, \mathbf{b})$
9: $\quad\quad\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}_\mathcal{S}\mathbf{w}$
$\quad$**return** $(\mathcal{S}, \mathbf{w})$

Hence, instead of adding only one cubature point at a time, in the importance sampled selection strategy, we add C points at each iteration. The strategy for selecting the C points is done through importance sampling according to the following probability mass function:

$$\text{PMF}(p) = R\frac{|\mathbf{a}_p \cdot \mathbf{r}|}{\mathbf{r} \cdot \mathbf{r}} \tag{3.30}$$

Here, R denotes the number of points not yet in the cubature set.

To implement the importance sampling algorithm, we replace lines 5–7 of the Algorithm 1 with the following:

Algorithm 2 Importance Sampled Cubature Selection

1: **function** IMPORTANCESAMPLEDCUBATURE(C)
2: $\quad$**while** C points have not been added to $\mathcal{S}$ **do**
3: $\quad\quad$Randomly select a candidate $p \notin \mathcal{S}$
4: $\quad\quad$Add p to $\mathcal{S}$ with probability PMF(p)

The importance sampling algorithm has the advantage of a superior run-time complexity: asymptotically, it runs in $O(rTn^3)$ time. Other approaches such as non-negative hard thresholding pursuit (NN-HTP) [61] and alternating direction method of multipliers (ADMM) [62] have been explored as well, but they are outside the

scope of this dissertation.

Chapter 4

Transform-based compression of fluid subspaces

Note: A large portion of this chapter has previously appeared as [63].

In the previous chapter, we discussed the potential for subspace methods to accelerate the computational cost of physics-based simulations. However, a significant drawback of the subspace approach is the time/memory tradeoff: the speed increase comes at a cost of much larger memory requirements. Specifically, subspace simulations can easily consume dozens of gigabytes of memory when dealing with high-resolution scenes. In this chapter, we discuss a compression method to reduce the memory footprint of subspace methods by an order of magnitude. Using such a compression scheme will allow us to compute longer and more complex scenes on powerful computers, while at the same time giving us the ability to compute scenes of reasonable complexity on laptop computers.

4.1 Previous Work

Since memory consumption is a known challenge with subspace techniques, other research has focused on reducing the memory footprint of these simulations. In the applications of sound [30] and blendshape matrices [29], compression techniques have been developed; however, we are unaware of analogous research in subspace fluid simulation. In the work of Wicke et al. [64], a modular fluid basis is used that can be tiled throughout the domain. However, our approach is complementary, as the modular tiles themselves could be further compressed by applying our algorithm.

4.2 Background

4.2.1 Data Compression Preliminaries

The general problem of data compression at first blush seems daunting: how do we reduce the memory footprint of a set of data without losing information? Fortunately, the sets of data that we typically work with are not random, but rather contain many patterns and redundancies. Exploiting these redundancies is the key to any successful data compression algorithm.

Data compression algorithms can be divided into two distinct families: lossless and lossy. A lossless compression algorithm is able to take an input signal, compress it, and reconstruct the exact same input signal. A simple example is run-length coding. For instance, to represent 100 white pixels, followed by 200 black pixels, followed by an another 50 white pixels, we can code the sequence ('w', 100), ('b', 200), ('w', 50), rather than coding the 350 pixels individually. Decoding is straightforward and retains all information. The ZIP file format is a familiar example of widely-used

lossless data compression, based on methods pioneered by Lempel and Ziv [24].

Lossless compression can be very powerful when applied to original data with many redundancies; however, it has strict limitations based on the mathematics of information theory [22]. Every file, no matter how redundant, contains a certain amount of information. Thus, for instance, after using run-length coding once on a file that previously contained many long sequential redundancies, trying to use it a second time on the newly compressed file will actually *increase* the file size, since there will no longer be any runs.

Lossy compression, by contrast, may not be able to reconstruct the exact same input signal. Its applications tend to focus on perceptual data, such as images, video, and sound. Although its reconstructions are not perfectly faithful, if the technique is soundly based on the limits of human perception, the results are virtually indistinguishable. The JPEG file format for images is a widely-used lossy data compression algorithm [25].

Data streams of images oftentimes contain more information than a human can reasonably resolve. For example, a picture might contain fine details that are indistinguishable from their smoothed counterpoints, or subtle color gradients that are indistinguishable from coarser ones. The key to most lossy data compression algorithms is finding a way of transforming the original information into a domain that captures these characteristics more naturally. These techniques are called *transform coding*.

4.2.2 Mathematical Preliminaries

In order to describe transform coding formally, we will require several notions from linear algebra and analysis.

Vector Spaces

The basic framework of most of our data manipulation can be expressed through the notion of vector spaces. We are familiar with the physical notion of vectors in two- or three-dimensional space as quantities with both a magnitude and a direction, often depicted graphically with an arrow, as in Figure 4.1. For convenience, we often represent vectors using coordinates. We can view the coordinate representation as a decomposition into fundamental unit vectors pointing in the x and y direction. These are called the *basis* vectors. Any arbitrary vector can be decomposed uniquely into a combination of these basis vectors.

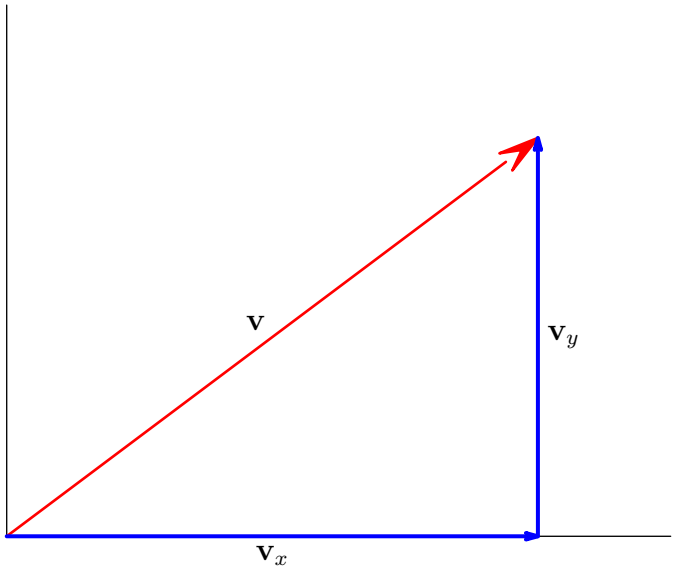


Figure 4.1: *A 2D vector* $\mathbf{v}$ *decomposed into its orthogonal components* $\mathbf{v}_x$ *and* $\mathbf{v}_y$.

Vectors can be added or subtracted, producing another vector. They can also be multiplied by a scalar (i.e., a number in $\mathbb{R}$ or $\mathbb{C}$), yielding another vector. The formalization of this idea is the concept of a *vector space*, which allows for a more abstract treatment of vectors. For example, an image represented on a computer as a sequence of N pixel values can be regarded as a vector living in the vector space $\mathbb{R}^N$.

This abstraction allows us to bring the tools of linear algebra to bear on a variety of seemingly different applications.

Inner Products

An *inner product*, or dot product, is a binary operation $\langle \cdot, \cdot \rangle$ between two vectors that yields a scalar as an output. The most familiar example is in $\mathbb{R}^2$, where, given two vectors $\mathbf{v} = (v_x, v_y)$ and $\mathbf{w} = (w_x, w_y)$, we have $\langle \mathbf{v}, \mathbf{w} \rangle = v_x w_x + v_y w_y$. Inner products add a notion of geometry to a vector space through angles and length. When an inner product between two vectors is 0, the vectors are orthogonal, and the inner product of a vector with itself gives the square of its length. For example, the basis vectors $\mathbf{e}_x = (1, 0)$ and $\mathbf{e}_y = (0, 1)$ are orthogonal, and the vector $\mathbf{v} = (3, 4)$ has squared length $\langle \mathbf{v}, \mathbf{v} \rangle = 3^2 + 4^2 = 5^2$, which matches with the usual Euclidean notion of squared length. Inner products also are closely related to the projection of one vector onto another. For example, if we take an arbitrary vector $\mathbf{v} = (v_x, v_y)$ and take its inner product with the unit basis vector $\mathbf{e}_y = (0, 1)$, the result is the component v_y in the y direction. In general, given a set of unit basis vectors which are mutually orthogonal (known as an *orthonormal* basis), we can compute the various components of arbitrary vectors by calculating their inner products with the corresponding basis vector.

Besides the familiar inner product in $\mathbb{R}^2$, more general inner products exist in other vector spaces. The canonical inner product in $\mathbb{C}^n$ is given by $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{k=1}^{n} v_k \overline{w_k}$, where the overbar denotes the complex conjugate. The conjugate may seem strange, but is necessary to preserve the notion that taking an inner product of a vector with itself should give the squared length of the vector.

The Discrete Fourier Transform

The *Discrete Fourier Transform*, or DFT, is a transformation that maps one vector in $\mathbb{C}^N$ to another vector $\mathbb{C}^N$. Given a vector $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$, the DFT maps $\mathbf{x}$ to the output $\mathbf{X} = (X_0, X_1, \ldots, X_{N-1})$, where

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N}, \quad k = 0, 1, \ldots, N-1 \tag{4.1}$$

This definition, as written, appears somewhat unmotivated, so we give a brief geometrical interpretation of the DFT in terms of inner products and roots of unity. To begin, we write $\omega = e^{i2\pi/N}$ as the first N-th root of unity[1]. Then the N powers of ω comprise the entire set of N-th roots of unity: $\omega^0, \omega^1, \omega^2, \ldots, \omega^{N-1}$. If we collect these into a single vector $\boldsymbol{\omega}$, we have

$$\boldsymbol{\omega} = \left(1, \omega^1, \omega^2, \ldots, \omega^{N-1}\right) \in \mathbb{C}^N. \tag{4.2}$$

We can also consider the N powers of $\boldsymbol{\omega}$:

$$\boldsymbol{\omega}^k = \left(1, \omega^k, \omega^{2k}, \ldots, \omega^{(N-1)k}\right), \quad k = 0, 1, \ldots, N-1 \tag{4.3}$$

The collection of N vectors $\{\boldsymbol{\omega}^0, \boldsymbol{\omega}^1, \ldots, \boldsymbol{\omega}^{N-1}\}$ forms an orthogonal[2] basis for $\mathbb{C}^N$, which we shall call the Fourier basis. As such, given an arbitrary vector $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$, we can compute its representation in the Fourier basis by projecting $\mathbf{x}$ against each of the new basis functions using the complex inner product. More concretely, the new representation $\mathbf{X} = (X_0, X_1, \ldots, X_{N-1})$ in the Fourier basis is given by

[1] That is, ω satisfies $\omega^N = 1$.

[2] Orthogonal, but not orthonormal, since each basis vector has magnitude $\sqrt{N} \neq 1$.

$$X_k = \frac{1}{\sqrt{N}} \langle \mathbf{x}, \boldsymbol{\omega}^k \rangle, \quad k = 0, 1, \ldots, N-1 \tag{4.4}$$

Recalling the definition of the complex inner product given in §4.2.2, this can be expand to agree with the original definition 4.1 up to a constant factor:

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \overline{\omega^{nk}} \tag{4.5}$$

$$= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \overline{\left(e^{i2\pi/N}\right)^{nk}} \tag{4.6}$$

$$= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N}, \quad k = 0, 1, \ldots, N-1 \tag{4.7}$$

This interpretation of projecting against the Fourier basis will prove more useful than the explicit formula for having an intuition about what the DFT is doing under the hood. In fact, using this interpretation, we can even write the DFT as the following $N \times N$ change-of-basis matrix $\mathcal{F}$, with each of the N columns given by the corresponding $\boldsymbol{\omega}^k$ vector:

$$\mathcal{F} = \begin{pmatrix} | & | & | & & | \\ \boldsymbol{\omega}^0 & \boldsymbol{\omega}^1 & \boldsymbol{\omega}^2 & \dots & \boldsymbol{\omega}^{N-1} \\ | & | & | & & | \end{pmatrix} \tag{4.8}$$

$$= \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(N-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \dots & \omega^{-(N-1)^2} \end{pmatrix} \tag{4.9}$$

Thus, the DFT of a vector $\mathbf{x} \in \mathbb{C}^N$ can now simply be defined by the matrix-vector multiplication $\mathcal{F}\mathbf{x} = \mathbf{X}$. In particular, since we already observed that the N column vectors of this matrix form on orthogonal basis for $\mathbb{C}^N$, by normalizing the DFT matrix to make each column have unit magnitude, we obtain the *unitary* form of the DFT:

$$\mathcal{F}_{\text{unitary}} = \frac{1}{\sqrt{N}}\mathcal{F} \tag{4.10}$$

Being unitary means that $\mathcal{F}_{\text{unitary}}$ preserves inner products. In other words, for any vectors $\mathbf{v}, \mathbf{w} \in \mathbb{C}^N$, we have

$$\langle \mathcal{F}_{\text{unitary}}\mathbf{v}, \mathcal{F}_{\text{unitary}}\mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle \tag{4.11}$$

We shall exploit this property later on in §4.3.3.

The asymptotic run-time of the DFT of a vector $\mathbf{x} \in \mathbb{C}^N$, if implemented as a matrix-vector multiply, would be $O(N^2)$, as we must compute N multiply-adds for each of the N columns. However, the well-known fast Fourier transform (FFT) algorithm, by recursively expressing the N-point DFT in terms of smaller $N/2$-point DFTs, is able to compute the same result in $O(N \log N)$ time, making it much more practical for large inputs [65].

The multidimensional DFT applies takes multidimensional arrays of inputs rather than vectors. For example, the two-dimensional DFT of the array $\mathbf{x} \in \mathbb{C}^{M \times N}$ is given by the array $\mathbf{X} \in \mathbb{C}^{M \times N}$ whose entries are

$$\mathbf{X}_{u,v} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathbf{x}_{m,n} e^{-i2\pi(um+vn)} \tag{4.12}$$

In the most general case, given a d-dimensional array $\mathbf{x} \in C^{N_1 \times N_2 \times \cdots \times N_d}$, its d-dimensional DFT is given by the array $\mathbf{X} \in \mathbb{C}^{N_1 \times N_2 \times \cdots \times N_d}$ whose entries are

$$\mathbf{X}_{u_1,u_2,\ldots,u_d} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_d=0}^{N_d-1} \mathbf{x}_{n_1,n_2,\ldots,n_d} e^{-i2\pi(u_1 n_1 + u_2 n_2 + \cdots + u_d n_d)} \tag{4.13}$$

In practice, however, we will stick to using $d = 2$ or $d = 3$.

4.2.3 The JPEG Coding Algorithm

The JPEG coding scheme is a form of lossy, transform compression for digital images. Because our subspace fluid compression scheme is based on many of the techniques used in JPEG, we shall explore the implementation of JPEG in some detail to better motivate our own coding algorithm.

The point of departure in JPEG is a digital image file, which comprises a two-

dimensional array of pixel values. (For simplicity, we will assume a grayscale image.) The basic redundancy that JPEG exploits is that for most images, the majority of the "energy" is packed into lower spatial frequencies. In other words, generally speaking, there are not many extremely sharp changes between hues in a typical image. (Notable exceptions include medical imaging, for which JPEG is typically not applied.)

So, to begin, suppose we have a two-dimensional grayscale image of resolution 800×800, as in Figure 4.2. While over the space of the entire image, there may be sharp changes between hues, if we subdivide the image into smaller regions, within each individual region, there are sharp changes much more rarely. Hence, the first step of JPEG is to subdivide the image into many smaller regions. The typical "sweet spot" chosen is 8×8 blocks, so in this case, there will be $100 \cdot 100 = 10000$ such blocks. Figure 4.2 shows one such block in the center of the whole image. The grayscale values (as unsigned 8-bit integers between 0 and 255) for this block $\mathbf{B}$ are as follows:

$$\mathbf{B} = \begin{pmatrix} 56 & 42 & 46 & 43 & 46 & 48 & 43 & 48 \\ 45 & 47 & 59 & 47 & 57 & 50 & 50 & 65 \\ 48 & 59 & 37 & 41 & 45 & 52 & 67 & 74 \\ 51 & 39 & 47 & 50 & 55 & 70 & 54 & 36 \\ 39 & 66 & 59 & 72 & 88 & 49 & 26 & 20 \\ 66 & 69 & 76 & 75 & 50 & 24 & 25 & 25 \\ 70 & 78 & 79 & 38 & 20 & 23 & 22 & 20 \\ 75 & 56 & 27 & 23 & 23 & 23 & 26 & 56 \end{pmatrix} \tag{4.14}$$

To exploit the redundancy of the image in the frequency domain, we now trans-

Figure 4.2: Left: *The original* 800×800 *image.* **Right:** *An* 8×8 *sub-block zoomed in.*

form each block according to a two-dimensional discrete cosine transform (2D-DCT). The discrete cosine transform, with some massaging, can actually be regarded as a particular case of the discrete Fourier transform, so conceptually, we can regard it in the same spirit. The resulting data transforms $\mathbf{B}$ into $\mathrm{DCT}(\mathbf{B}) = \widehat{\mathbf{B}}$ as follows:

$$\widehat{\mathbf{B}} = \begin{pmatrix} 388.13 & 48.51 & 3.43 & -5.62 & 3.63 & -10.51 & 1.42 & 2.12 \\ 21.90 & -65.26 & -10.63 & 10.88 & -0.39 & 1.66 & 2.68 & 2.75 \\ -27.13 & 5.92 & 53.83 & -8.29 & 5.35 & 1.02 & 11.21 & 0.80 \\ 6.26 & 45.24 & -45.46 & -13.37 & 1.97 & 3.12 & 0.07 & 6.32 \\ -12.63 & -15.30 & -11.94 & 28.36 & 11.38 & -4.05 & 1.56 & -3.43 \\ -8.00 & 9.89 & 14.93 & 5.72 & -20.23 & 16.34 & 9.22 & -2.35 \\ 0.82 & -8.88 & 17.96 & -0.44 & 13.89 & 4.13 & -10.33 & -7.66 \\ 0.16 & 3.34 & -1.24 & -2.58 & -5.01 & -15.33 & -11.65 & -0.71 \end{pmatrix} \tag{4.15}$$

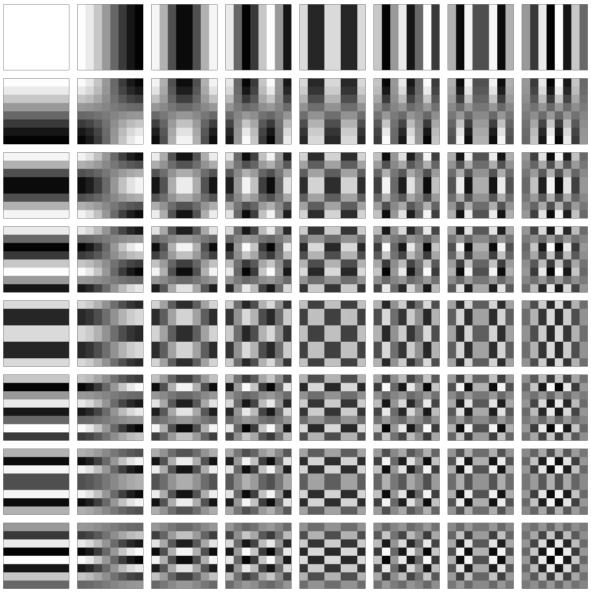


Figure 4.3: *The* 64 *2D-DCT basis vectors for an* 8×8 *domain. Any* 8×8 *image can be expressed as a linear combination of these vectors. For example, the* 8×8 *array in 4.15 gives the corresponding weights that would generate the original sub-block* **B** *in 4.14.*

Next, we *dampen* the frequencies by point-wise dividing the resulting set of values by a pre-determined damping array, $\mathbf{Q}_{2D}$, given by Equation 4.16. This array was determined empirically through perceptual limits and is given in [25].

$$\mathbf{Q}_{2D} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{4.16}$$

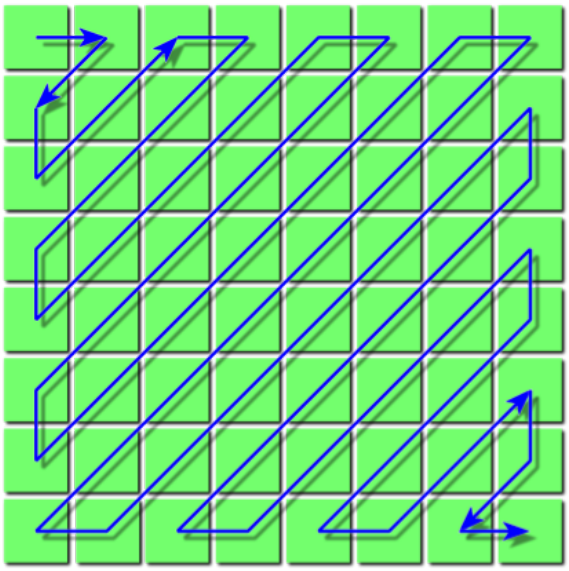


Figure 4.4: *The zigzag scan from northwest to southeast corner. The goal is to catch long consecutive strings of zeroes.*

The goal of this procedure is to dampen enough of the frequencies to values small enough that, upon integer rounding, they become zero. This allows for compression gains, albeit at the cost of information loss. Equation 4.17 shows the result.

$$\text{Round}\left(\frac{\widehat{\mathbf{B}}}{\mathbf{Q}_{2D}}\right) = \begin{pmatrix} 24 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -5 & -1 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & -2 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.17}$$

Note how many of the entires have been dampened to zero, as desired. Next, we scan through this array in a zigzag fashion, as demonstrated by Figure 4.4.

This allows us to catch long strings of zeroes. In this case, the final 31 entries are all zeroes, so run-length coding will prove to be highly effective. Although the full JPEG standard includes several additional wrinkles, these are beyond the scope of the current discussion.

The decoding procedure is a straightforward reversal of each of the coding steps: for each block, we first decode the run-length code, we unzigzag the data into the correct shape, we undampen it by multiplying by $\mathbf{Q}_{2D}$, and finally, we compute the inverse two-dimensional discrete cosine transform (inverse 2D DCT). Figure 4.5 shows the JPEG-decoded block side-by-side with its original counterpart. The overall average error in this block comes out to approximately 7 values per pixel (on a 256-value grayscale). Figure 4.6 shows the whole image compressed. The average error across the whole image comes out to about 2 bits per pixel, with an overall compression ratio of around $8:1$. Despite artifacts being noticeable at the block level when zoomed in, as in Figure 4.5, the overall image has no visible artifacts.

Lossy compression typically has a "sweet spot"—that is, a level of compression that achieves a significant data reduction without introducing noticeable artifacts. We also see in Figure 4.6 the effect of going beyond this sweet spot and destroying the image quality: here, despite the attractive $100:1$ compression ratio, the image is unacceptably distorted.

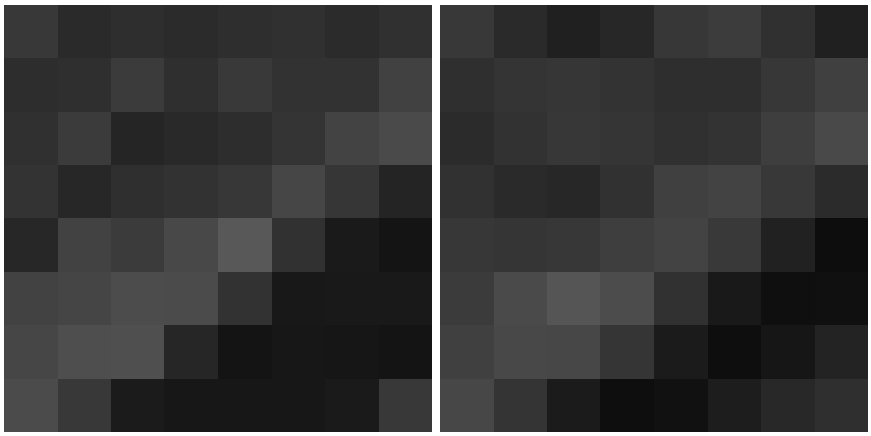


Figure 4.5: Left: *The original, uncompressed* 8×8 *data block.* **Right:** *The same* 8×8 *block after undergoing JPEG compression at quality* 50 *percent. Observe the general smoothing, which is an artifact of the information lost during the dampening and rounding stage.*

Figure 4.6: Left: *A compression ratio of* $8:1$ *is achieved using the* 50 *percent quality damping array with no visible artifacts.* **Right:** *A compression ratio of* $100:1$ *is achieved using the* 5 *percent quality damping array; however, there are many visible artifacts.*

4.3 A Subspace Compression Scheme

Recall from Chapter 3 the subspace projection matrix **U** that creates a heavy memory footprint. A compression method to reduce its size would alleviate this potential

computational bottleneck. However, in order for the subspace simulations to run smoothly, we also must ensure that the compression method efficiently supports the following three operations:

- **Projection:** The time needed to compute the full $\mathbf{U}^T\mathbf{u} = \mathbf{q}$ matrix-vector product should not prohibitively increase due to the presence of a decompressor.
- **Dense reconstruction:** Conversely, $\mathbf{Uq} = \mathbf{u}$ must also be fast.
- **Batched random access:** In order to support sparse reconstruction, it should be possible to query the velocity field at a set of random points on the simulation grid. In addition to efficient sparse reconstruction, this feature is also needed to support certain types of time integration, e.g. cubature-based semi-Lagrangian schemes [55].

With these requirements in mind, we can design our codec.

4.3.1 Compression Basis Selection

In order to design a compression scheme for the velocity fields that comprise the columns of the subspace projection matrix $\mathbf{U}$, we must first select a transform basis that would ideally result in extremely sparse fields. Both discrete cosine transform (DCT) [66] and wavelet [67, 68] bases have been successfully used in the past to compress scalar volumes, so they are promising candidates for velocity fields. We choose to use DCT because we observe that in the special case of Laplacian Eigenfunctions [69], they actually yield ideal compression. The eigenfunctions inside a closed 3D

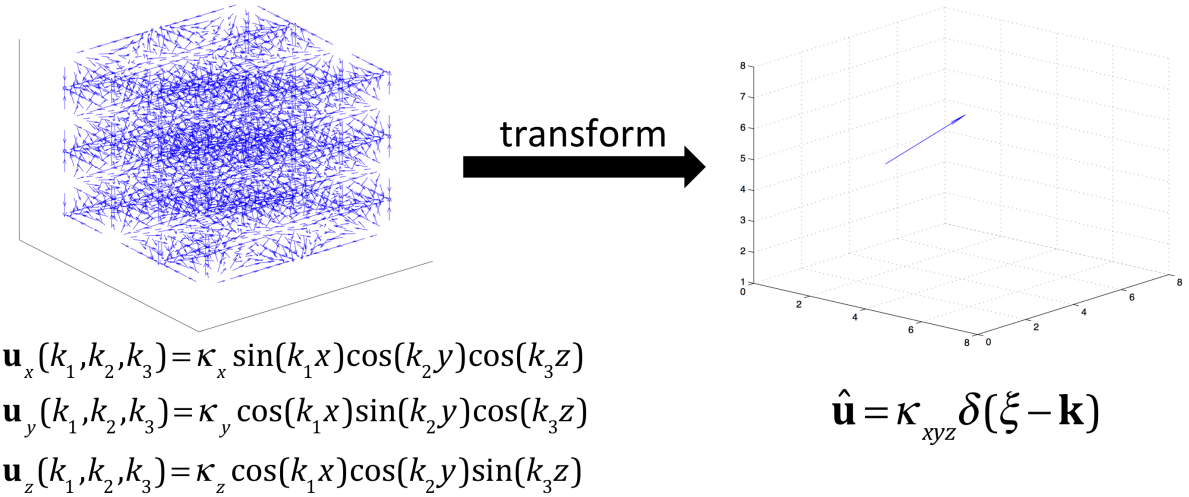


Figure 4.7: *An eigenfunction of the Laplacian operator transforming into a delta function in frequency space.*

box take the general form:

$$\begin{aligned}
\mathbf{u}_x(k_1,k_2,k_3) &= \kappa_x \sin(k_1x)\cos(k_2y)\cos(k_3z) \\
\mathbf{u}_y(k_1,k_2,k_3) &= \kappa_y \cos(k_1x)\sin(k_2y)\cos(k_3z) \\
\mathbf{u}_z(k_1,k_2,k_3) &= \kappa_z \cos(k_1x)\cos(k_2y)\sin(k_3z),
\end{aligned} \tag{4.18}$$

where $\mathbf{u}_x, \mathbf{u}_y$, and $\mathbf{u}_z$ respectively represent the x, y, and z components of a velocity field. The k_i coefficients determine the frequency content of the field, and the three κ terms are scaling coefficients that are derived from k_i.

We make the straightforward observation that the spatially varying components of each of these functions are purely trigonometric functions, so by applying appropriately interleaved DCTs and discrete sine transforms (DSTs) to these fields, they can be reduced to delta functions, regardless of their spatial frequency. In the notation of Long and Reinhard [70], if we use $\mathcal{F}_{\text{SCC}}$ to denote a DST in the x direction and DCTs

in the y and z directions, we obtain,

$$\mathcal{F}_{\text{SCC}}\left[\kappa_x \sin(k_1 x)\cos(k_2 y)\cos(k_3 z)\right] = \kappa_x \delta(k_1, k_2, k_3), \tag{4.19}$$

where $\delta(k_1, k_2, k_3)$ is a delta function in SCC frequency space located at the (k_1, k_2, k_3) grid cell. Similar transforms, e.g. $\mathcal{F}_{\text{CSC}}$ and $\mathcal{F}_{\text{CCS}}$ can be used to generate delta functions for $\mathbf{u}_y$ and $\mathbf{u}_z$. Thus, any eigenfunction can be compressed down to three integers (the k_is) and three floats (the κ terms).

Asymptotically, this mixed DCT/DST losslessly transforms an $O(N)$ eigenfunction down to $O(1)$; no sparser representation is possible. This result only applies to the ideal case of analytic divergence-free velocity fields defined on the interior of a box. However, the high compression it achieves is encouraging, and motivates our further use of DCT to compress more general divergence-free fields.

4.3.2 DCT-Based Compression

Following from the previous discussion, we design a DCT-based, JPEG-like compression scheme. Each column of $\mathbf{U}$ represents a vector field, and the columns are usually constructed using an SVD. Since this SVD has already minimized the amount of redundant information between columns, we compress them separately. Each column contain x, y and z velocity components, and we extract each of these components and compress them separately. Thus, if we describe the encoding procedure for a single scalar field, it can be applied to each velocity component of each column of $\mathbf{U}$.

Analogously to JPEG, given a 3D scalar field, we decompose it into small blocks of size $b \times b \times b$, adding continuous extra padding in the case that one or more resolutions are not evenly divisible by b. We then perform a 3D-DCT on each block. In anticipation of quantization, the result is then normalized so that the largest frequency-

domain value maps to the largest signed value for a 32-bit integer.

Adaptive Quantization: After transforming the signal to the frequency domain and normalizing the coefficients, as discussed in §4.2.3, the JPEG coding scheme then performs an element-wise division of the coefficients using a 2D quantization matrix in order to increase the likelihood that they will quantize to zero. In the JPEG standard, this matrix is adjusted depending on the quality setting; higher quality settings will have lower values to preserve more detail after dividing by the matrix, while lower quality settings will have higher values in the matrix to suppress more coefficients. For example, we previously saw in Equation 4.16 the JPEG matrix corresponding to 50% quality.

We index the matrix entries $\mathbf{Q}_{2\mathrm{D}}(u, v)$ such that the upper-left corner, $\mathbf{Q}_{2\mathrm{D}}(0, 0) = 16$, corresponds to the DC (i.e., direct current, or zero frequency) component. The entries of $\mathbf{Q}_{2\mathrm{D}}$ were obtained from perceptual data [71], and in general, higher frequency components have larger entries in order to suppress these coefficients which tend to carry less information than those in the lower frequencies.

Since we are not working with 2D color data but rather with 3D velocity fields, we need to construct a 3D version of this matrix, $\mathbf{Q}^{\gamma}_{3\mathrm{D}} \in \mathbb{R}^{b\times b\times b}$. A close inspection of all of the complete matrix corresponding to Eqn. 4.16 suggests that $\mathbf{Q}_{2\mathrm{D}}(u, v) \propto u + v$, so a straightforward first attempt is:

$$\mathbf{Q}^{\gamma}_{3\mathrm{D}}(u, v, w) = 1 + u + v + w. \tag{4.20}$$

Other applications [66] have used similar reasoning to arrive at similar matrices. However, while the 2D case has a suite of $\mathbf{Q}_{2\mathrm{D}}$ matrices at its disposal that correspond to different levels of perceived visual quality, this data does not generalize to non-chromatic, 3D velocity data.

We instead propose to automatically generate a variety of different $\mathbf{Q}_{3D}^{\gamma}$ matrices during the compression stage. For different $b \times b \times b$ blocks, the energy is likely to reside in different frequencies, so we generate a one-parameter family of matrices,

$$\mathbf{Q}_{3D}^{\gamma}(u, v, w) = (1 + u + v + w)^{\gamma}, \tag{4.21}$$

where γ is a parameter that is adjusted per block. Analogous to the 2D case, the user specifies a quality parameter p. In 3D, we interpret p as the percentage of the original energy[3] that should be preserved. Each block then performs a bisection search over the range $\gamma \in [0, n]$, where $n = 32$ is the number of bits that were used for normalization prior to quantization. Higher values of n are essentially meaningless, as they damp everything except the DC component to zero. In practice, we found that this bisection search terminates within a very small tolerance of the desired energy preservation after at most 8 iterations.

This approach provides a custom quantization matrix for each block while maintaining an approximately constant energy loss per block. Important high-frequency components are preserved when they are present, while smoother, low-frequency blocks are still aggressively compressed. This block-varying value γ must then be computed by the encoder and provided to the decoder in the encoded bytestream. For an $8 \times 8 \times 8$ block, the memory footprint of a single additional scalar γ per block is negligible. We compare this strategy to a uniform non-adaptive quantization approach in §4.4.

Flattening and Encoding: After quantization, we convert the 3D array into a 1D array and perform run-length encoding [66, 71]. No novel strategy needs to be devised for this component. The 3D to 1D conversion is performed in a zigzag pattern that is

[3] That is, L_2 norm.

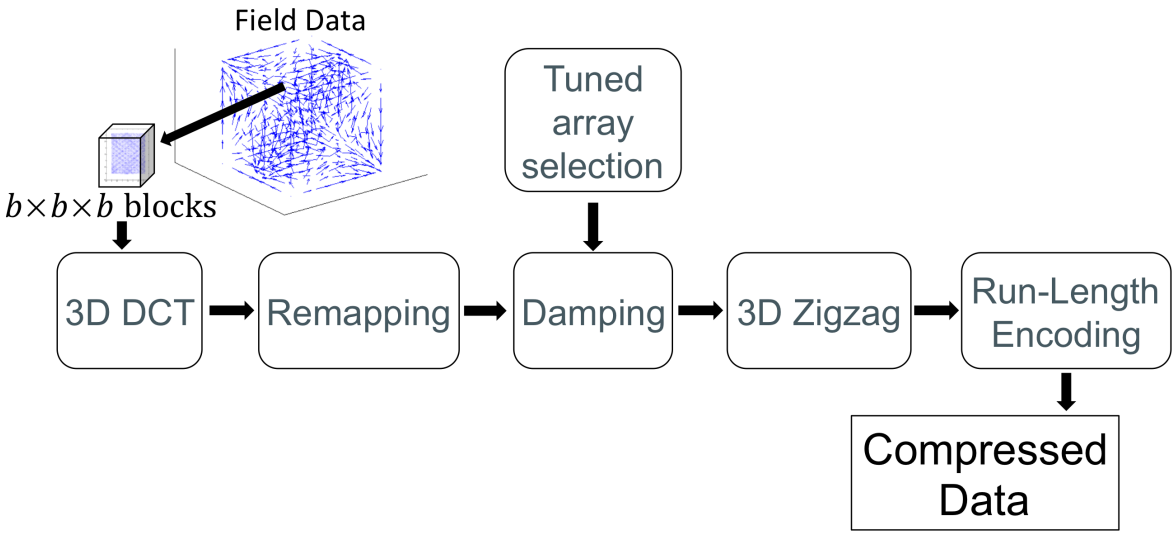


Figure 4.8: *An overview of the encoding pipeline.*

a straightforward 3D extension of the usual 2D JPEG ordering, which tries to group coefficients with similar sizes together in the bytestream. In our case, the entries of $\mathbf{Q}_{3D}^{\gamma}(u,v,w)$ are arranged in increasing order of their sum $u+v+w$, which effectively clusters components with approximately the same frequency. The results are then run-length encoded in order to discover long runs of zeros.

4.3.3 Subspace Decompression

Batched Random Access: The block-wise compression scheme we have described supports the batched random access requirements described in the beginning of §4.3 at runtime. For a single random access, the block containing the cell of interest is decompressed, which means the other $b \times b \times b - 1$ entries are potentially decoded needlessly. However, the coherency of the underlying incompressible flow tends to cluster cells of interest in the same blocks, so we did not find that this extra overhead created a major bottleneck.

The decompression proceeds in two stages, where an initial pass assembles the

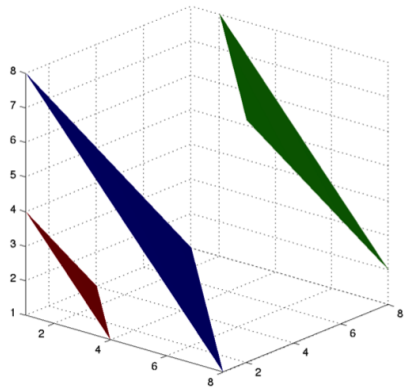


Figure 4.9: *The 3D zigzag scan order. We move along slice planes of increasing index sum* $c = 1 + u + v + w$.

batch of requested cells and determines which blocks need to be decompressed. A second pass then decompresses the actual blocks. By consolidating the cell requests, it is guaranteed that no block is ever redundantly decompressed twice.

Projection and Reconstruction: The fast projection and reconstruction requirements from §4.3 are not as straightforward. A naïve strategy is to decompress the entire matrix $\mathbf{U}$ for each projection and reconstruction. For a given block size $B = b \times b \times b$, this results in $\frac{3N \times r}{B}$ DCTs and IDCTs at every timestep. These transforms dominate the running time (Table 4.1), and largely negate the performance gains of the subspace approach. While memory savings are achieved, the speed-memory tradeoff is unacceptable.

However, we observe that both of the matrix-vector products $\mathbf{Uq} = \mathbf{u}$ and $\mathbf{U}^T\mathbf{u} = \mathbf{q}$ can be performed *sparsely in the frequency domain*. The projection operator then only performs a DCT on $\mathbf{u}$, not all r columns of $\mathbf{U}$. This operation is permissible because the DCT is a unitary transform, and therefore preserves inner products, as seen in §4.2.2. Specifically, if $\mathbf{x}$ and $\mathbf{y}$ are vectors in the spatial domain and $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are their counterparts in the frequency domain, $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \hat{\mathbf{x}}, \hat{\mathbf{y}} \rangle$.

We define the following notation to describe the advantages of this approach. The frequency domain version of a quantity is denoted with a hat, e.g., $\mathbf{U}$ with DCT applied to each column is $\widehat{\mathbf{U}}$. The *lossy, compressed* version of $\widehat{\mathbf{U}}$, where near-zero values have been quantized to zero, is denoted $\widehat{\mathbf{C}}$. The spatial domain version of $\widehat{\mathbf{C}}$ is correspondingly $\mathbf{C}$. In essence, our compression scheme has introduced the approximations $\mathbf{U} \approx \mathbf{C}$ and $\widehat{\mathbf{U}} \approx \widehat{\mathbf{C}}$.

Using the unitary property discussed in §4.2.2, we can see that if we use DCT to transform $\mathbf{u}$ to $\hat{\mathbf{u}}$, then $\mathbf{U}^T\mathbf{u} = \mathbf{q}$ is equivalent to $\widehat{\mathbf{U}}^T\hat{\mathbf{u}} = \mathbf{q}$. Using the compressed versions will yield a result slightly different from $\mathbf{q}$, but the same relation holds: $\mathbf{C}^T\mathbf{u} = \widehat{\mathbf{C}}^T\hat{\mathbf{u}} \approx \mathbf{q}$. The naïve approach spends most of its time transforming $\widehat{\mathbf{C}}$ to $\mathbf{C}$, but by constructing $\hat{\mathbf{u}}$, we can avoid this stage altogether. Replacing the IDCT over all r columns of $\widehat{\mathbf{C}}$ with a single DCT of $\mathbf{u}$ is significant, because even for a modest $r = 10$, the number of transforms is reduced by an order of magnitude. Additionally, the $\widehat{\mathbf{C}}^T\hat{\mathbf{u}}$ product can now exploit the sparsity of $\widehat{\mathbf{C}}$ that was discovered by the compression stage. Since $\widehat{\mathbf{C}}$ is static over the lifetime of a simulation, the location of all the zero entries can be cached, and these multiplies can be skipped.

A fast reconstruction strategy then follows due to the linearity of the DCT: $\widehat{\mathbf{C}}\mathbf{q} \approx \hat{\mathbf{u}}$. The sparsity of $\widehat{\mathbf{C}}$ can again be exploited here, as each column $\widehat{\mathbf{C}}$ is scaled by an entry of $\mathbf{q}$, and all the multiplies with respect to zeroes can again be skipped. Once $\hat{\mathbf{u}}$ is known, an IDCT can be performed on it once, and IDCTs over all r columns of $\widehat{\mathbf{C}}$ is again avoided.

For a $\mathbf{C}$ matrix containing $3N \times r$ non-zero entries, after taking the complexity of the DCTs and IDCTs from §4.2.2 into account, naïve projection and reconstruction each take $O\left(3N \times r + \frac{3N \times r}{B} B \log B\right) = O\left((3N \times r) \log B\right)$ time. Using our approach, a $\widehat{\mathbf{C}}$ containing S non-zero entries instead takes $O\left(S + \frac{3N}{B} B \log B\right) = O\left(S + 3N \log B\right)$ time. The r factor has been removed as a multiplier of $\log B$, and replaced with the

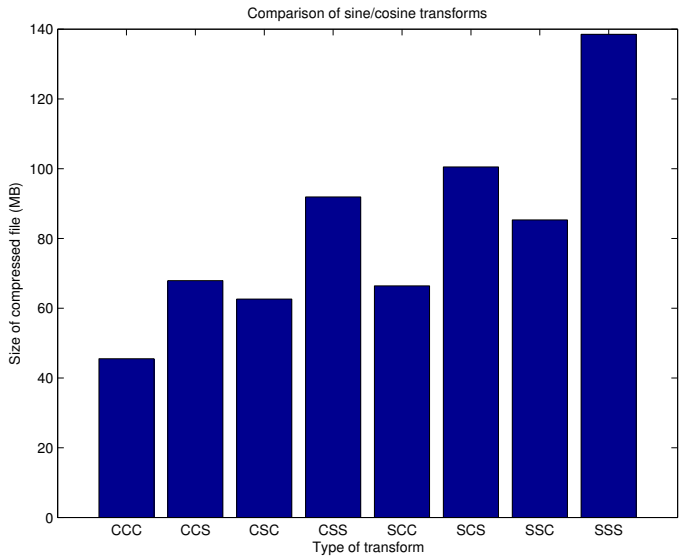


Figure 4.10: *A comparison of compression results using the* 8 *possible cosine/sine interleavings through each of the three spatial dimensions. 'C' and 'S' are abbreviations for a cosine and sine transform, respectively. We see that the 'CCC' transform yields the most compressed result with the smallest output file size.*

additive S term. As seen in Table 4.1, this results in speedups that approach an order of magnitude.

4.3.4 Discussion

We have described one possible scheme for compressing subspace fluid basis matrices. Several initially promising possibilities were also investigated, but ultimately discarded.

The motivating example from §4.3.1 uses mixed DST and DCT to achieve ideal compression, whereas we only use DCT. The use of DST was investigated as well, but was not found to give superior results. Unless the velocity values along a block border are all exactly zero, i.e. the block has Dirichlet boundaries along all its walls, the DCT consistently yields superior results, as shown in Figure 4.10.

The **U** matrix is usually constructed using an SVD [21, 55] or an eigenanalysis [69, 72]. Therefore, corresponding singular values or eigenvalues are usually available for each column of **U**. These values could be used to guide the compressor, e.g., by allowing columns with unimportant singular values σ_i to be compressed more aggressively. However, we found that the relationship between σ_i and visual quality is not straightforward. Especially during re-simulation, columns that had unimportant σ_i during the initial analysis can obtain large coefficients in $\mathbf{q}$. In such cases, the aggressive compression can become visible.

Prior to compression, an additional SVD could be run on each $b \times b \times b$ block in **U** to determine if there is a superior coordinate system for compression other than the canonical x, y, and z axes. However, the per-block rotation that this introduces breaks the fast matrix-vector multiply described in §4.3.3. Thus, we put this aside in favor of the fast multiplies, but a method that supports both operations is an interesting direction for future work.

4.4 Results

We tested our compression scheme on several subspace fluid re-simulation scenarios generated by the open-source package Zephyr [55]. The fluid simulation data were generated using a Preconditioned Conjugate Gradient (PCG) solver with a Modified Incomplete Cholesky preconditioner [73]. The codebase was implemented in C++ and tests were run on a 12-core, 2.66 GHz Mac Pro with 96 GB RAM. For the DCT and IDCT, we used FFTW [74], and Eigen [75] was used for other linear algebra operations.

In all of our simulations, we set $b = 8$, so $8 \times 8 \times 8$ blocks were used. Block sizes of $b = 4$ and $b = 16$ were also tested, but we found that smaller blocks redundantly

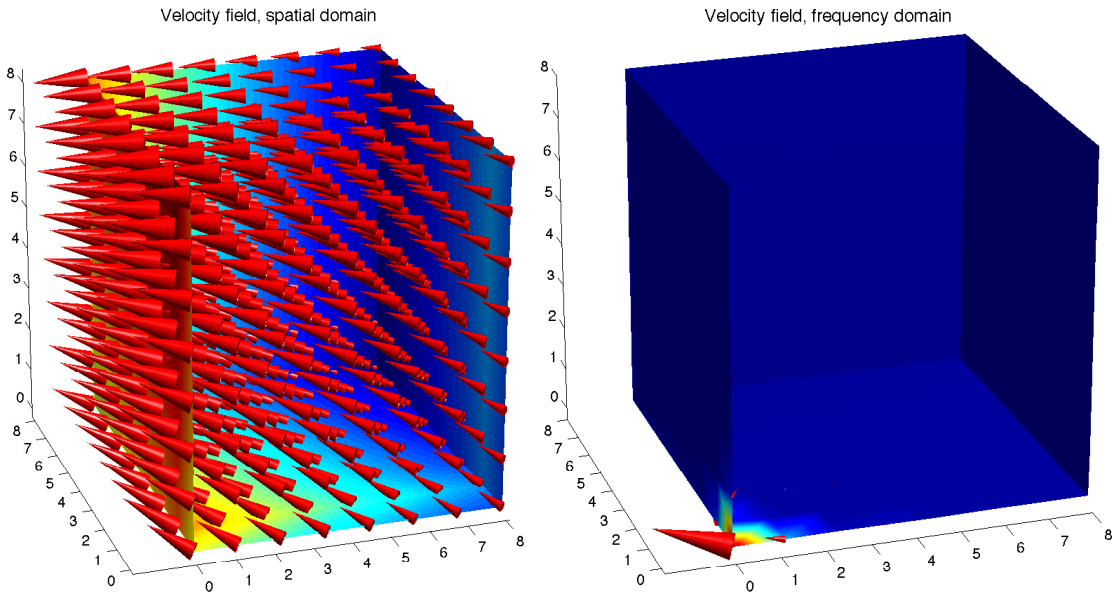


Figure 4.11: Left: *An* $8 \times 8 \times 8$ *block of one of the velocity fields obtained from the Singular Value Decomposition.* **Right:** *The same* $8 \times 8 \times 8$ *velocity field block in the frequency domain after taking a 3D Discrete Cosine Transform. The sparsity not only allows for transform compression but also frequency-domain subspace projection speedups.*

captured the same low frequency information, while larger blocks lessened the likelihood of finding smooth regions that could be compressed aggressively. Figure 4.12 summarizes these results.

In all of our subspace simulations, we used the matrix-vector product strategy from §4.3.3, which accelerated the computation significantly (Table 4.1). Without this acceleration, the subspace simulations ran almost at parity with the original full-space simulations, invalidating any speed advantages of the subspace approach. On average, our sparse product ran roughly 3–4 times as slow as the uncompressed matrix vector product. Asymptotically, our sparse product can have a superior running time, as it does not need to touch all $3N \times r$ entries in the matrix. Our scenes did not achieve sufficient sparsity to demonstrate this superiority, but we expect that as compression methods improve, multiplying against $\widehat{\mathbf{C}}$ will eventually become faster

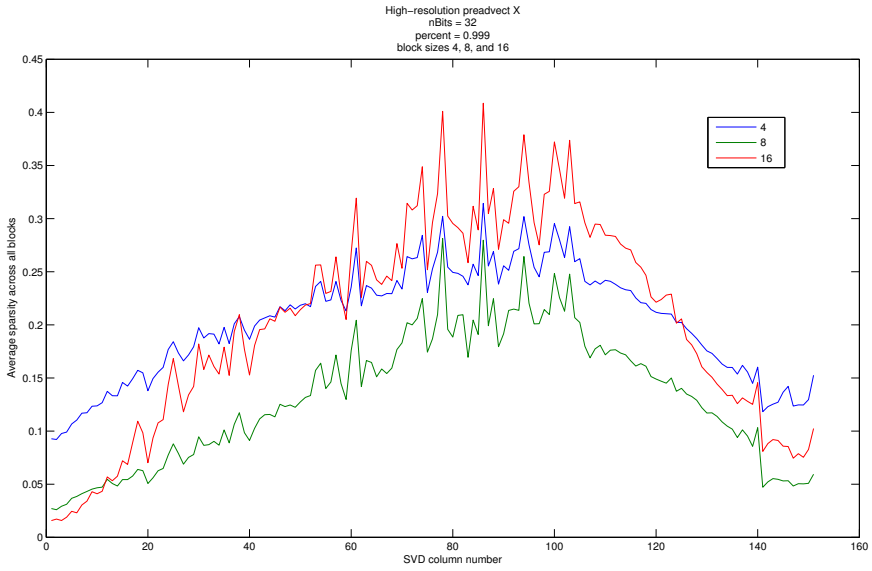


Figure 4.12: *Sparsity comparison of using different block sizes. Using $b = 8$ leads to sparser, and hence more compressible results.*

	Naïve, e.g., $\mathbf{Cq}$	Sparse, e.g., $\widehat{\mathbf{C}}\hat{\mathbf{u}}$	Speedup
Plume	72.0s	8.7s	8.3X
Sphere	74.9s	7.6s	9.9X
Fan	72.6s	12.9s	5.6X

Table 4.1: *Timings of naïve projections vs. sparse projections. The sparse projection is significantly faster, and dramatically reduces the overhead of using the compressed representation of* $\mathbf{U}$*. These timings represent the average time to perform* both *the projection and reconstruction stages in a single timestep.*

Plume	Uncompressed	6 : 1	8 : 1	**11 : 1**	13 : 1	22 : 1
Time per frame	4.5s	19.9s	17.9s	16 1s	15.2s	12.8s
Compression preprocess	N/A	02h 07m 55s	01h 53m 48s	02h07m55s	02h 12m 17s	02h 15m 59s
Sphere	Uncompressed	10 : 1	**14 : 1**	16 : 1	23 : 1	30 : 1
Time per frame	5.4s	17.1	15 1s	14.4s	13.0s	12.2s
Compression preprocess	N/A	02h 01m 38s	02h20m14s	02h 07m 59s	02h 34m 30s	02h 27m 59s
Fan	Uncompressed	5 : 1	**6 : 1**	8 : 1	11 : 1	29 : 1
Time per frame	5.7s	24.4s	23 2s	19.8s	18.3s	14.8s
Compression preprocess	N/A	01h 09m 37s	01h08m25s	01h 26m 57s	02h 18m 45s	02h 11m 51s

Table 4.2: *Compression performance for each of the three scenes. The "sweet spot" for each scene that achieves a good balance between compression and visual quality is shown in gray.*

than multiplying with **C**.

Plume scene: We simulated the buoyant flow of a plume in a scene containing no obstacles, as shown in Figure 4.13. The original simulation resolution is $200 \times 266 \times 200$, was run for 150 frames, and took 09h 48m 49s (3.92 minutes per frame).

The SVD to construct the subspace from this data took 07h 05m 40s, and the compressing the subspace took at most 02h 15m 59s (Table 4.2). Constructing and compressing the subspace is therefore roughly at parity with running the entire simulation a second time. However, once the subspace has been constructed once, we can run new simulations very quickly.

We found that the subspace could be compressed by roughly an order of magnitude (11 : 1) before the visual quality began to degrade. However, we also noted that the compression scheme appears to degrade relatively gracefully. For higher compression rates, the motion gradually deviates from the uncompressed motion, and JPEG-like block artifacts begin to appear (cf. the actual JPEG artifacts exhibited in §4.2.3).

Sphere scene: Next, we simulated the same plume in the presence of a sphere obstacle. The original simulation resolution is $200 \times 266 \times 200$, was run for 150 frames, and took 10h 37m 50s (4.25 minutes per frame). The time to construct and compress

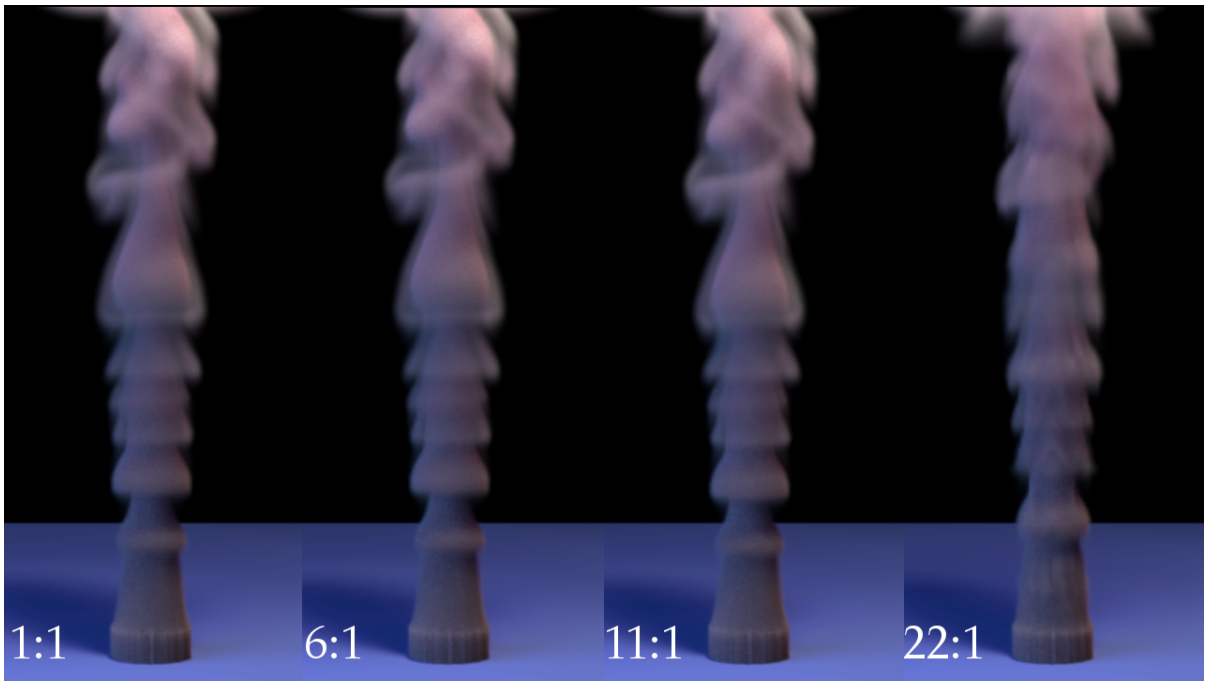


Figure 4.13: Plume scene: *The overall motion and visual quality of the plume is preserved until the compression ratio is increased to approximately* $22:1$. *The* $1:1$ *corresponds to using the original* $\mathbf{U}$ *matrix.*

the subspace, respectively 09h 17m 19s and a maximum of 02h 34m 30s, was again found to be roughly at parity with running the simulation a second time.

Surprisingly, we found that this subspace compressed slightly better than the plume scene ($14:1$) before the visual quality began to degrade. The expectation was that the sphere boundary would create a discontinuity in the velocity field that the compressor would have trouble capturing. Instead, the interior of the static obstacle created a region of constant (zero) velocity, and also induced the formation of smooth, near-zero regions in its vicinity. Rather than create a discontinuity containing many high frequencies, these constant and smooth regions mostly contained low-frequencies that the compressor could easily leverage. No-slip boundaries were used along the surface of the obstacle; if free-slip were used instead, the anticipated discontinuities may still appear.

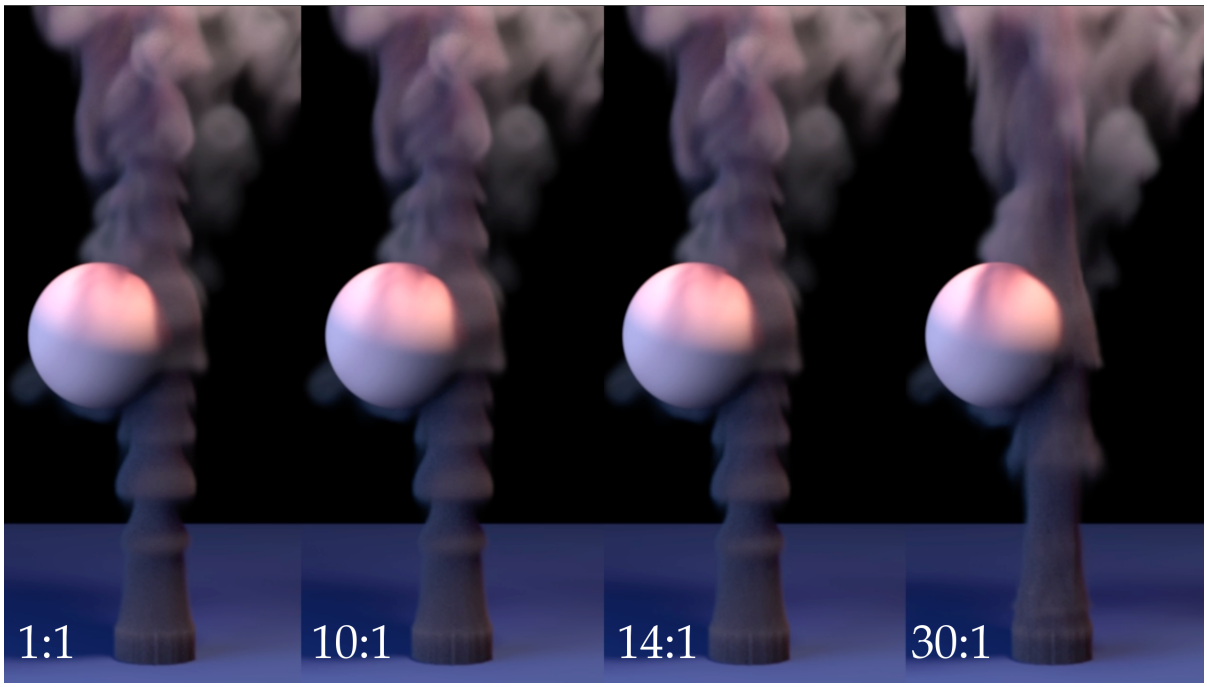


Figure 4.14: Sphere scene: *The motion and visual quality remains high until approximately* 14 : 1 *compression. At* 30 : 1*, the differences are very significant.*

For this scene, we also compared our adaptive quantization strategy to a uniform, non-adaptive approach. There is no canonical 3D version of Eqn. 4.16, so we instead selected a uniform γ that produced an equivalent energy loss in the highest frequency component, i.e., we set γ such that it matched the 32-bit equivalent of the lower-right hand entry of Eqn. 4.16. This strategy still produced a 7 : 1 compression, but our adaptive strategy was able to achieve a higher compression of 14 : 1.

Out-of-core Comparison: We compared our performance to an uncompressed simulation that does not fit in core by running the 14 : 1 version of the sphere scene on a 2-core, 1.8 GHz Macbook Air with 8 GB RAM. On this system, the full space simulation ran at 286.5s per frame, while our compressed subspace simulation ran at 71.7s, yielding a speedup of roughly 4.0×.

The uncompressed subspace simulation requires over 70 GB of memory, and immediately started to swap on the 8 GB system. It ran for over 17 hours without

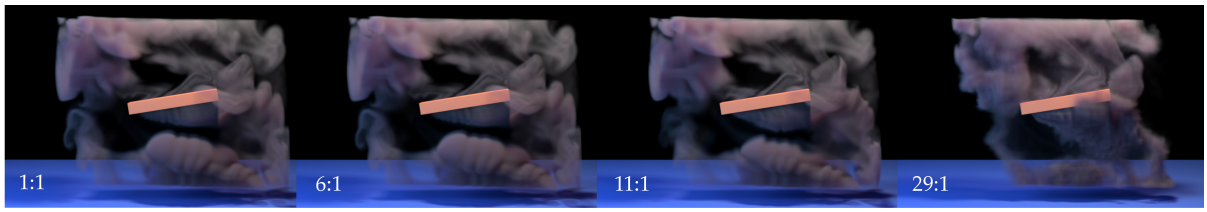


Figure 4.15: Fan scene: *The quality is preserved at* 6 : 1*, but at* 11 : 1*, the motion begins to change, and at* 29 : 1*, artifacts begin to appear.*

completing a single frame, at which time it had more than doubled the running time of the original full space simulation, and was terminated.

Fan scene: Finally, we simulated smoke being stirred by a fan. The original simulation resolution is $266 \times 200 \times 200$, was run for 150 frames, and took 12h 18m 05s (4.92 minutes per frame). The subspace construction and compression times were 08h 13m 22s and at most 02h 18m 45s.

The moving obstacle scene achieved a compression ratio of 6 : 1 before the quality began to degrade. Here, the reduced compression we expected to see in the sphere scene appeared. Instead of having a smoothing effect on the velocity field, the obstacle creates new, high-frequency velocities that are more difficult to compress. However, we did not choose to leverage any knowledge of the obstacle motion during compression, which could be a promising avenue for finding sparser representations.

4.5 Discussion and Conclusions

In order to determine whether the compression compromised the generality of the subspace, we also ran re-simulations [55] on each of our scenes using a variety of different simulation settings. In the plume scene, we both reduced the vorticity confinement constant to zero and doubled the number of total timesteps (see video), for

the sphere scene we halved the buoyancy constant (see video) and in the fan scene, we increased the vorticity confinement constant by a factor of ten (Fig. 4.16). The re-simulations were all run on the "sweet-spot" compression ratios that are highlighted in Table 4.2. In all cases, the overall motion was preserved, and we did not observe any significant visual artifacts compared to the uncompressed subspace re-simulation.

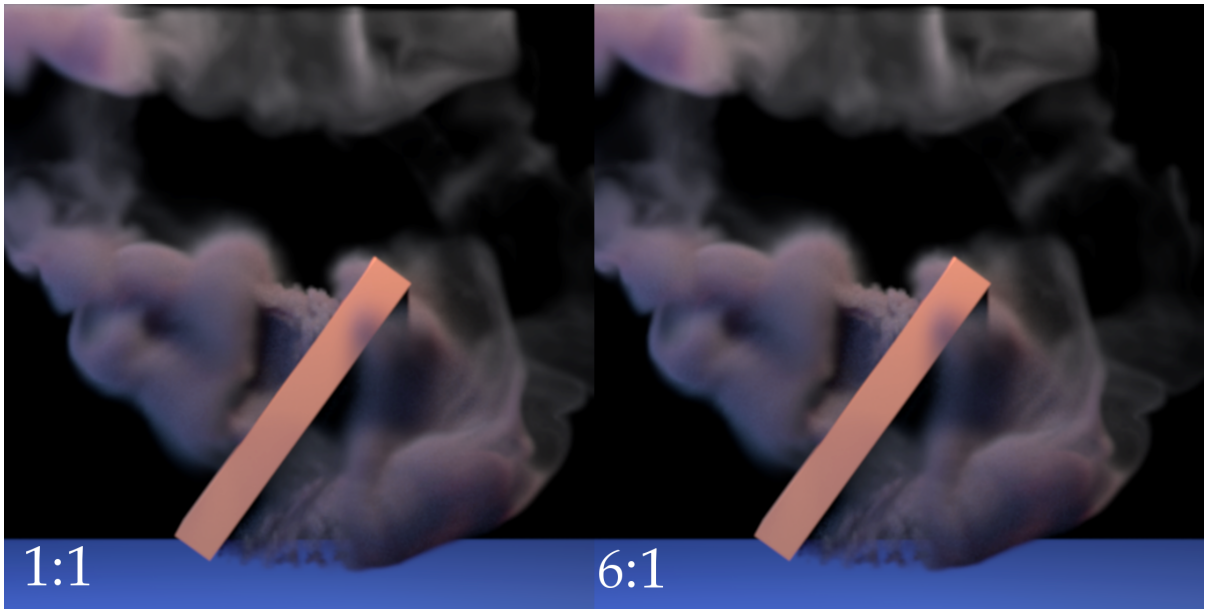


Figure 4.16: Fan Re-Simulation: *With vorticity confinement increased by a factor of ten, the novel turbulent detail that is introduced remains intact, even after basis compression.*

In order to better understand the error introduced by the compression, we plotted the relative L_2 error between the $\mathbf{q}$ vectors obtained by the uncompressed and compressed simulations. While it is difficult to tell whether a given compressed $\mathbf{U}$ is too aggressive *a priori*, an overaggressively-compressed $\mathbf{U}$ quantitatively corresponds to a 10^{-1} relative error appearing early in the simulation. In this case, the compression error exceeds that of the cubature integration scheme [55], and begins to visually dominate.

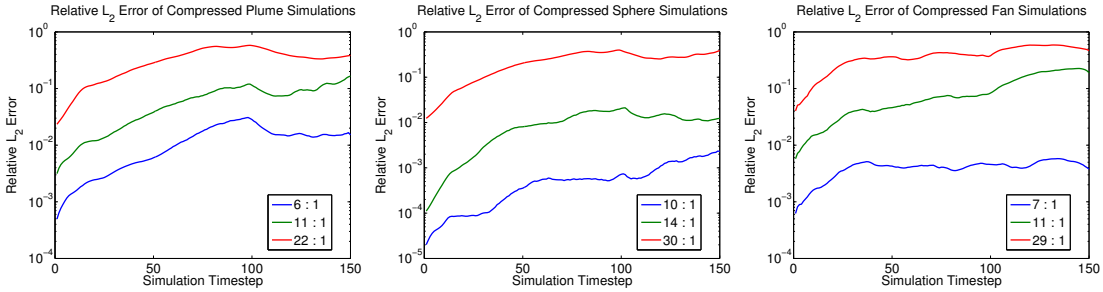


Figure 4.17: *Relative L_2 error introduced by the compressed matrix* $\mathbf{C}$ *compared to an uncompressed* $\mathbf{U}$ *over the course of a simulation. The transition between visually undetectable and visible error corresponds to roughly an order of magnitude jump in the relative error.*

Conclusions and Future Work: We have presented a compression method for subspace fluid simulations that is able to reduce the size of the $\mathbf{U}$ matrix by up to an order of magnitude. The most immediate direction for future work is the development of algorithms that are able to reduce this matrix size even further. As seen in Table 4.2, as the sparsity of the compressed representation increases, the speed of the matrix-vector product improves. According to our asymptotic analysis, it should eventually surpass the performance of the direct $\mathbf{Uq}$ product. Therefore, finding new bases that possess the same unitary property as the DCT while increasing sparsity should yield algorithms that are efficient in both memory and time.

Lossless compression schemes such as Huffman coding could also be investigated, but these would not directly increase the sparsity of the representation, so the trade-off between memory savings and decompression speed would need to be balanced. Finally, the method we have presented assumes that the velocity field is defined on a regular grid. An approach that can be applied to unstructured, tetrahedral meshes [72, 76] would also be welcome.

Chapter 5

Visualizing and sonifying fluid subspaces

Note: A large portion of this chapter has previously appeared as [77].

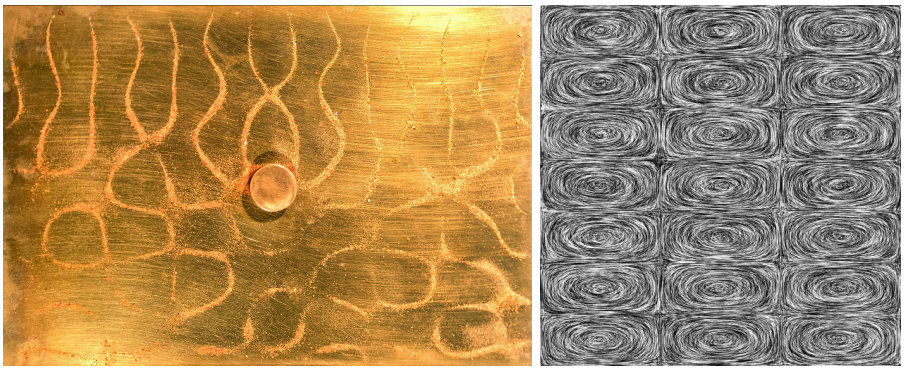


Figure 5.1: Left: *Chladni patterns realized through a physical experiment. Source: Wikimedia Commons.* **Right:** *Analytic* 2D *eigenvector of the Navier-Stokes equations using the method of de Witt et al.* [69].

5.1 Introduction

As discussed briefly in Chapter 1 during the discussion of cymatics, Chladni plates reveal beautiful patterns when vibrated at specific frequencies (Figure 5.1). Both the spatial patterns and sonic frequencies that arise have long been known to have intimate connections to the eigenvectors and eigenvalues of an idealized rigid plate. For this idealized case, closed form expressions can be obtained, and the eigenvalues are known to correspond to the audio spectrum emitted by the plate. In this chapter, we seek to explore a generalization of this phenomenon by applying a similar procedure to a more complex scenario: a turbulent computational fluid dynamics (CFD) simulation.

Unlike the Chladni plate case, closed form expressions are not available for the eigenvectors of an arbitrary CFD simulation, so we instead discover a set of "empirical" eigenvectors, as described in Chapter 3 during the discussion of subspace methods. The natural connection between eigenvalues and audio frequencies in the Chladni plate case is also no longer present, so we instead construct a sonification, as discussed more thoroughly in 2, to produce a mapping between fluid trajectories and sound. Using this approach, we obtain a variety of organic forms that have a unique visual character and generate associated sounds that unfold over rich spectral envelopes. Our ability to directly control the spatial and audio frequency spectrum allows the potential for a mathematically-principled artistic exploration of the audiovisual space through the medium of short film. We demonstrate the potential of this system for artistic expression with several brief preliminary results. Further exploration of the audiovisual system follows in Chapter 6.

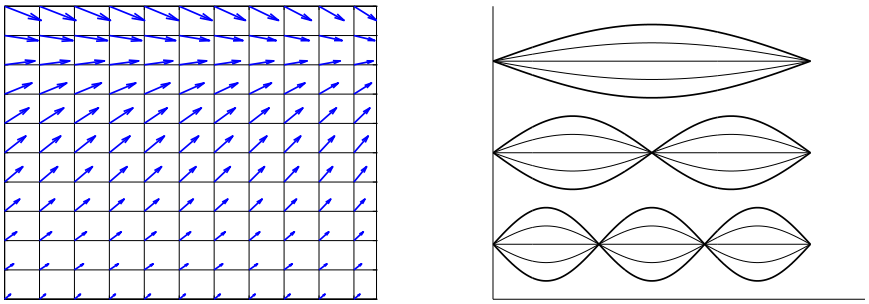


Figure 5.2: Left: *A* 2D *velocity field over a regular grid.* **Right:** *From top to bottom, the modes of vibration of a guitar string for* $N = 1, 2, 3$.

5.2 Eigenvector Preliminaries

In order to better understand the role they play in this work, we start with a review of eigenvectors and values. The eigendecomposition of a square matrix $\mathbf{A}$ is usually written as $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$. The matrix Λ is zero except along its diagonal, and each individual entry along the diagonal is an "eigenvalue," where the German *eigen* roughly translates to "characteristic." The matrix $\mathbf{Q}$ is generally not diagonal, and contains columns that are each considered an "eigenvector" of $\mathbf{A}$. The essential character of the matrix is captured by these vectors and values because they satisfy a specific relationship. If we place the ith column of $\mathbf{Q}$ into a vector $\mathbf{q}$, and the ith diagonal entry of Λ into a scalar λ, the relation $\mathbf{Aq} = \lambda\mathbf{q}$ will always hold. The vector $\mathbf{q}$ will be scaled by the value λ, but will otherwise remain unchanged.

There are many different ways to understand this relationship, but the scenario of a vibrating string offers a clean physical interpretation. One way to describe the phenomenon of vibration is as one of a fixed shape that is repeatedly scaled. When a guitar string is plucked, it visibly forms the shape of a sine wave over $(0, \pi)$. Over time, the amplitude of this shape scales up to some positive value, attenuates back

to zero, and then scales down to some negative value that is symmetric with the positive one. This cyclical sequence of amplitudes in time encapsulates the visual phenomenon of vibration.

The $\mathbf{Aq} = \lambda\mathbf{q}$ relation can be understood to model exactly this behavior. If the entries of $\mathbf{q}$ are set to be a sine wave over $(0, \pi)$, and the entries of the matrix $\mathbf{A}$ are set according to the correct physical principles[1], then multiplying by $\mathbf{A}$ will produced a scaled version of $\mathbf{q}$. Repeated multiplications will produce a sequence of scaled vectors, mimicking vibration. The eigenvectors thus describe a set of characteristic vibrational shapes that a string is capable of producing. The other columns of the matrix $\mathbf{Q}$ describe other shapes ("vibration modes") that a string is capable of producing. For example, sine waves defined over $(0, 2\pi)$, $(0, 3\pi)$ and up to $(0, N\pi)$ will all make an appearance (Figure 5.2). (Note the boundary conditions force nodal points into the shape of vibration, thus precluding the possibility of sine waves at frequencies that would otherwise vibrate at those points.) They are less dominant from a physical perspective, which is why the $(0, \pi)$ version is the one we most visually associate with a vibrating string. This dominance is reflected in the eigenvalue analysis—the $(0, \pi)$ sine wave appears as the first vector in $\mathbf{Q}$, and each successive column yields progressively less visually and sonically prominent shapes.

This understanding can be extended to both 2D and 3D, but some effort is needed to rearrange these higher-dimensional phenomena so that they can be packed into a 1D vector $\mathbf{q}$. For example, we can cut a 2D square plate into a regular grid, and rearrange the 2D values defined over this grid into a 1D vector $\mathbf{q}$. A matrix $\mathbf{A}$ can again be assembled according to physical principles so that its eigenvectors correspond to the characteristic vibrations of a square plate. When these eigenvectors are

[1] In this case, $\mathbf{A}$ would correspond to the wave equation: a Laplacian with Dirichlet boundary conditions.

rearranged into a 2D grid, their visual character is in close agreement with those found by laboratory experiments (Figure 5.1).

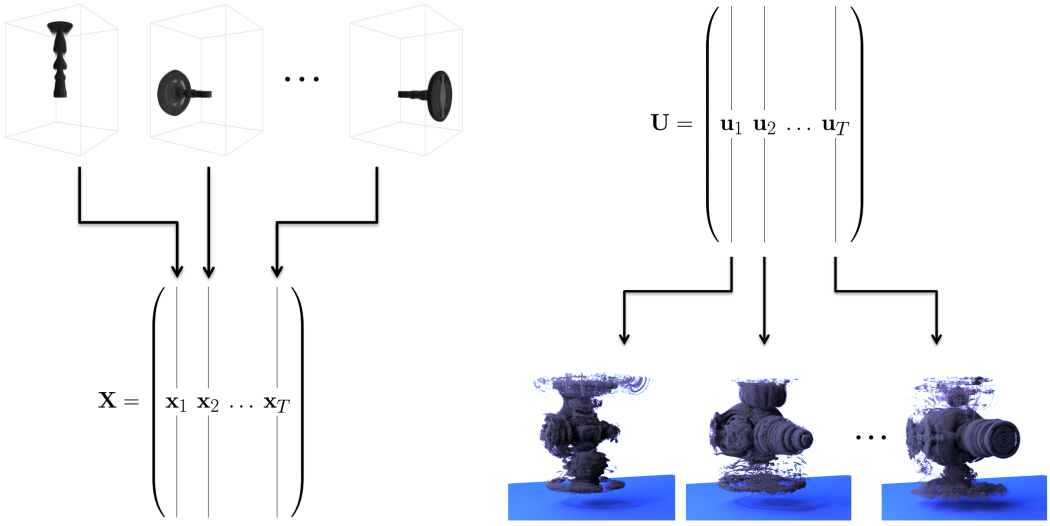


Figure 5.3: Left: *Assembling the velocity fields column-wise into a matrix* **X**. *We use simulations of a plume moving toward each face of its bounding box.* **Right:** *Obtaining the empirical eigenvectors after the singular value decomposition is performed on* **X**, *yielding* **U**.

5.3 Empirical Eigenvectors in Computational Fluid Dynamics

The eigenvector analysis that automatically discovers Chladni patterns does not extend directly to more complex phenomena. Nevertheless, we are interested in discovering analogous patterns that arise in turbulent fluid flows, as they may have artistic value. The equations for these flows are inherently non-linear, while the eigenvector approach is linear, so we will instead use the method of "empirical" eigenvectors. In order to motivate these techniques, we will review some concepts from computational fluid dynamics discussed in Chapter 3.

Recall that a fluid is usually defined using a velocity field, where a velocity vector

is associated with each point in space. While there are many different ways to represent these fields, we take the perspective from the previous section where a bounded region of space has been diced into a set of regular squares (or, in 3D, cubes). Each cube is then assigned a corresponding velocity vector (Figure 5.2), and in order to generate an animation, the vectors must then be evolved over time. There are many different equations that can be used to specify this evolution in time, but, following §3.2, we use the well-known incompressible Navier-Stokes equations for fluid flow. Hence, we assume that we have divided time into a discrete number of steps, and that at each step, the vector inside each cube in our computational grid has been assigned an appropriate value.

Unlike the classic Chladni pattern case, CFD simulations do not yield a single matrix $\mathbf{A}$ on which an eigenvalue analysis can be performed. If the pressure, advection, diffusion, and external forces are composed together into a single matrix $\mathbf{A}$, then this matrix $\mathbf{A}$ is time-dependent and must be updated at every timestep. Hence, instead of one canonical $\mathbf{A}$ that can be said to characterize the behavior of the entire system, there are instead infinitely many $\mathbf{A}$s, none of which inherently take precedence over the others, as discussed in §3.4.3. Fortunately, if some sort of precedence is imposed, a process akin to an eigenvalue analysis can still be applied. The method of "empirical" eigenvectors [78], also known as the Proper Orthogonal Decomposition, Karhunen-Love expansion, or Hotelling transform, establishes one such criterion. Informally, given an existing simulation, we can analyze the series of matrices that arose during that simulation and give those matrices precedence over all others. The space of infinite $\mathbf{A}$s is thus reduced to a tractable, finite size.

Returning to Figure 5.2, reviewing the method of selecting a subspace from training data in §3.4 we first perform a simulation over a regular $N \times N$ grid for T timesteps. While the simulations in our actual experiments were run in 3D (Figure

5.4), we will limit our discussion to 2D for simplicity. Each grid cell then contains a 2D velocity vector which possesses an x and y component, so the entire grid contains $N \times N = N^2$ such vectors. We can rearrange these values so that they are packed into a 1D vector $\mathbf{x}$, which then has $2N^2$ entries. We can perform this repacking for each of the T velocity fields from the simulation and concatenate all of the vectors into a matrix $\mathbf{X}$ with $2N^2$ rows and T columns (Figure 5.3, left).

Generally, $2N^2 \neq T$, so the matrix $\mathbf{X}$ will be rectangular, and an eigenvalue analysis can only be performed on square matrices. However, the singular value decomposition (SVD) can always be performed regardless of dimension, and the results possess many eigenvalue-like qualities. Instead of the eigendecomposition $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$, the SVD instead yields $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$. Similar to the eigendecomposition, the middle matrix Σ is diagonal, and its entries correspond to the "singular" values of the matrix $\mathbf{X}$. Again, similar to the eigenvalue case, the columns of the left matrix $\mathbf{U}$ form an ordered set of the most important shapes, or quasi-vibration modes, that appeared during the simulation (Figure 5.3, right). The matrix $\mathbf{V}$ is a T-dimensional rotation matrix that was applied to $\mathbf{X}$ in order to arrive at $\mathbf{U}$ and Σ; for our purposes, it can be discarded.

We are interested in the representation formed by $\mathbf{U}$ and Σ for two reasons. First, these two quantities comprise a quasi-frequency spectrum. The shapes that are encoded in each column of $\mathbf{U}$ are roughly analogous to the sine waves from the string case. If we take a single step from the original fluid simulation, $\mathbf{x}_t$, and apply the matrix-vector multiply $\mathbf{U}^T\mathbf{x}_t = \mathbf{q}_t$, then we have performed a quasi-Fourier transform that translates $\mathbf{x}_t$ into a quasi-frequency domain. It then becomes straightforward to start interpreting the entries of $\mathbf{q}_t$ as the amplitudes in some auditory representation. Second, an inverse-quasi-Fourier transform has also been defined. Given some arbitrary audio signal $\mathbf{q}_*$, we can convert back to a spatial shape by performing the

operation $\mathbf{U}\mathbf{q}_* = \mathbf{x}_*$. Given some sound unfolding over time, we can then generate a sequence of velocity fields to drive a fluid's motion.

Finally, empirical eigenvectors are a topic of interest in engineering because running simulations in this quasi-frequency-domain can have certain computational advantages. These "subspace" simulations were explored in greater detail in Chapters 3 and 4.

5.4 Visualization

One of the challenges when using empirical eigenvectors is the construction of an interesting set of eigenvectors. For example, if a set of smooth, featureless laminar flows are input into the SVD, there is no reason to believe that the shapes (a.k.a modes) corresponding to the resulting eigenvectors will be visually interesting. In order to ensure that a rich set of eigenvectors are produced, we ran six separate CFD simulations using a standard fluid simulator [46], where a turbulent plume of smoke was aimed at each face of a rectangular simulation domain. All of the simulation data was then concatenated into a matrix $\mathbf{X}$. To reveal the shape of the modes, we then ran a series of simulations with each of the modes sequentially isolated. A "delta" vector $\mathbf{q}_d$, similar to an impulse response, was generated at each timestep where the dth entry is set to 1 and the rest of the vector was set to zero. A selection of these results can be seen in Figure 5.4. More detailed discussion of this choice of training data follows in Chapter 6.

Other excitation strategies are also possible, including ones that are more closely guided by the physics that generated the original input data. We can gradually activate each eigenvector in sequence and then push the resulting vector $\mathbf{q}$ through a subspace simulation that approximates the Navier-Stokes equations [55]. Less phys-

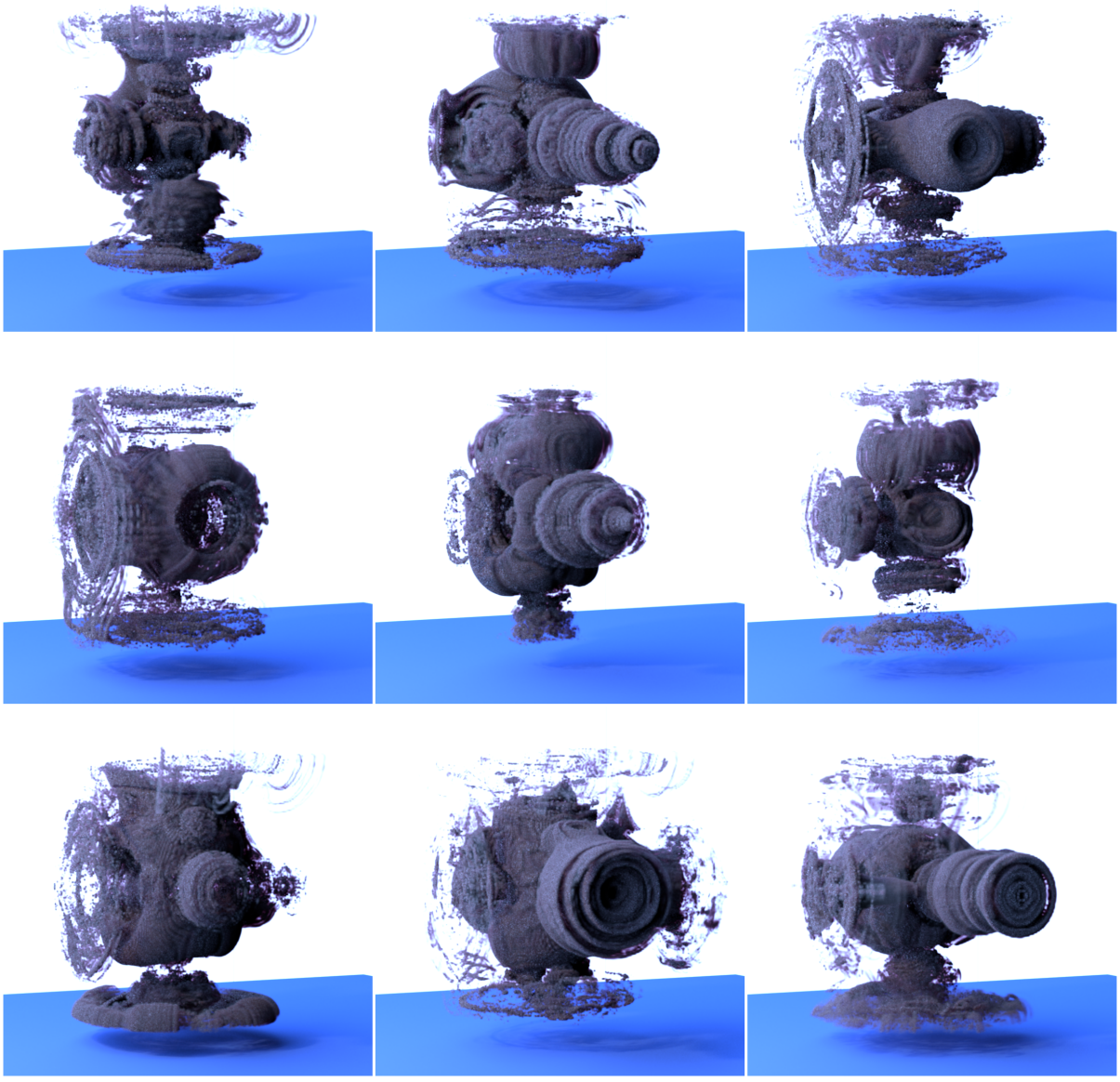


Figure 5.4: *In reading order: An assortment of nine of the* 150 *empirical eigenvectors, from lowest singular value to highest, discovered by taking the combined SVD of six separate Navier-Stokes simulations.*

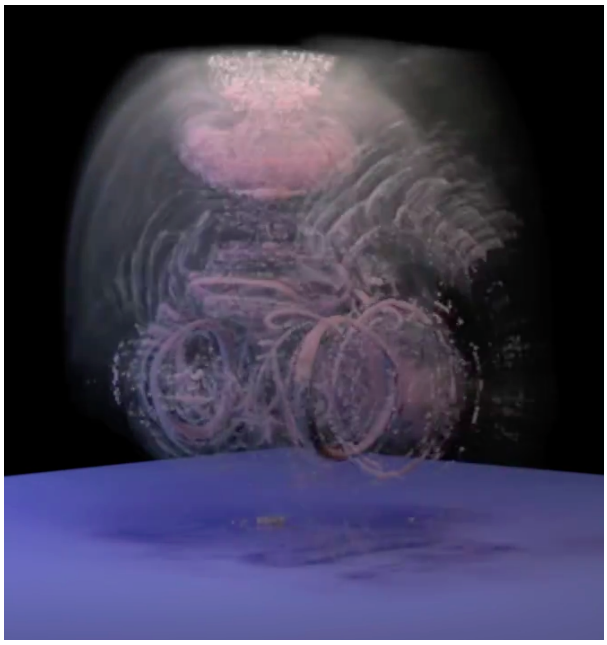


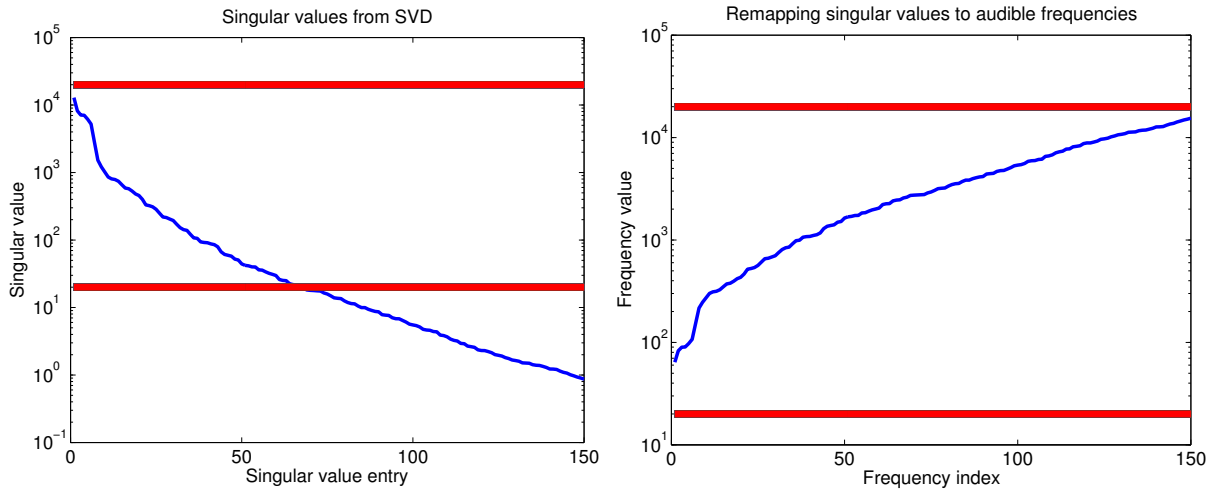
Figure 5.5: *A still frame from a random walk over an r-dimensional sphere of constant radius.*

ical approaches are also possible by treating $\mathbf{q}$ as a set of mixing weights. We can then construct arbitrary paths φ, sampled at T time steps $\varphi_1, \ldots, \varphi_T$, that determine the corresponding T time steps of a full-coordinate velocity field. In this vein, we show the fluid simulation that results from a random walk over a higher-dimensional sphere of constant radius in the following video.[2] (Figure 5.5 shows a still frame from the video for reference.)

5.5 Sonification

We will now devise a strategy for converting the motion of a fluid into a sound. As mentioned previously, one of the main advantages of the empirical eigenvectors approach is that it already yields a quasi-frequency spectrum. However, some additional choices need to be made before these vectors can be made audible. As

[2] *Sphere*: `https://youtu.be/y6zraBpH5PA`



(a) Singular values: $T = 150$ **(b)** Remapped frequencies: $f = 64$ Hz, $s = 1.75$

Figure 5.6: *Singular values and their remapped frequencies. Note the logarithmic scale on both y-axes. The horizontal red lines on both plots indicate the bounds of the human audible frequency range.*

reviewed in Chapter 2, this sequence of subjective but judicious choices is known as sonification. Our system resembles a hybrid strategy between Parameter Sonification (PMSon), discussed in §2.1.3, and Model-Based Sonification (MBS), discussed in §2.1.2.

We begin by interpreting the T singular values $\sigma_1, \ldots, \sigma_T$ from the singular value decomposition as characteristic frequencies. There are two immediate concerns about this raw data. First, we can see in Figure 5.6a that the values range over approximately 4 orders of magnitude, which is approximately 13 octaves, and decrease to numbers less than 1. However, the human audible frequency range begins at 20 Hz and spans approximately 10 octaves, up to 20000 Hz [79]. Thus, we must somehow normalize the data to this audible range. Secondly, the singular value spectrum begins with a maximum value and then decreases, whereas an audio spectrum starts at fundamental frequency and then increases. Hence, we invert the singular values. One way to construct a practical mapping is to specify a fundamental frequency f

and an octave scaling s. Writing our audible frequencies as $f_1, \ldots, f_T$, we can define our mapping from singular values to audible frequencies as follows:

$$\begin{aligned} f_i &= f \cdot \left(\frac{\sigma_i^{-1}}{\sigma_{\max}^{-1}} \right)^{\frac{1}{s}} \\ &= f \cdot \left(\frac{\sigma_{\max}}{\sigma_i} \right)^{\frac{1}{s}}, \; i = 1, \ldots, T. \end{aligned} \tag{5.1}$$

The effect of this remapping can be seen in Figure 5.6b, where we have used a fundamental frequency of $f = 64$ Hz and an octave scaling of $s = 1.75$. The spectrum now begins at the fundamental, $f = 64$ Hz, and ranges up to a maximum of approximately 15000 Hz, which is an acoustically acceptable spread.

With an audible spectrum established, we now choose amplitudes for each individual frequency. These can be mapped from a corresponding subspace vector $\mathbf{q} \in \mathbb{R}^T$, as each of its T components, $q_1, \ldots, q_T$ can be thought of as an amplitude for the T corresponding frequencies $f_1, \ldots, f_T$. Care must be taken here to ensure that sum of all the amplitudes does not exceed unity gain, as this would lead to clipping. More concretely, we constrain the L_1 norm of $\mathbf{q}$, $|\mathbf{q}|_1 = \sum_{i=1}^{T} |q_i|$, to be at most 1. A typical subspace vector $\mathbf{q}$ may also contain negative components. However, these can simply be thought of as encoding a positive amplitude and a reversal of phase. The phase reversal can be discarded, as its perceivable effect is typically undesirable clicking artifacts.

With these considerations in hand, we design a mapping from a subspace vector $\mathbf{q}$ to an amplitude vector $\mathbf{a}$ of unit L_1 norm as follows:

$$a_i = \frac{|q_i|}{|\mathbf{q}|_1}, \; i = 1, \ldots, T. \tag{5.2}$$

Given a sequence $(\mathbf{q}_t)$ of vectors that describe a sampled trajectory $\mathbf{q}_1, \ldots, \mathbf{q}_T$ through

the subspace $\mathbb{R}^T$, we can normalize each $\mathbf{a}_t$ vector individually. Alternatively, we can determine which $\mathbf{q}_t$ has the maximum L_1 norm, which we denote $\mathbf{q}_{\max}$, and normalize based on $|\mathbf{q}_{\max}|_1$. The effect of the former is a relatively uniform volume level, while the latter yields a more variable volume envelope. Each approach has its own musical merit depending on the compositional situation. Further analysis and justification of our sonification choices is given in Chapter 6.

5.6 Synthesis

With our mappings carried out, we can now produce an audible sound. To illustrate the basic mapping between the visual forms of the eigenvectors and their corresponding remapped frequencies, we move sequentially through the eigenvectors and their corresponding frequencies in the following video.[3] (Figure 5.7 shows a single frame from this video for reference.) This mapping generates a musical scale, and compositional operations can be carried out at the note level. For example, by scrambling the order of the frequencies and adding some rhythmic variety, we can produce a short melody, as demonstrated in the next video.[4] (Figure 5.8 shows a single frame from this video for reference.)

The previous example essentially captures freeze frames of each individual mode. However, a smoother, more sophisticated auditory mapping is needed if we want to capture the flow of the smoke through mixtures of the modes. As a point of departure, following the synthesis model in §2.1.4, we use subtractive synthesis, which passes a spectrally rich input signal through various filters in order to alter its timbre. In our case, we construct a filter bank that attenuates the input signal everywhere except at the resonant frequencies that correspond to the active modes. A particular

[3] *Sequential*: `https://youtu.be/cB79S4NwCHc`
[4] *Melody*: `https://youtu.be/N6fzJXbn2ts`

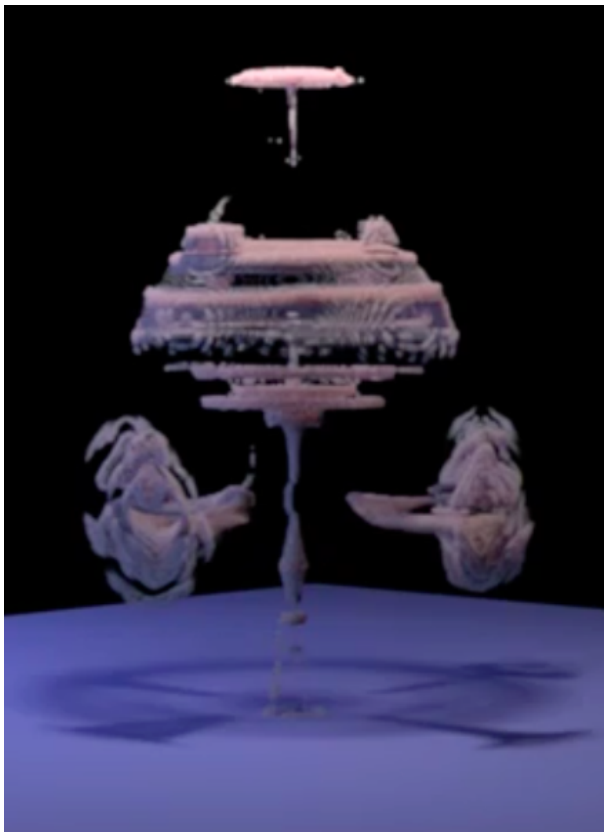


Figure 5.7: *A still frame from moving through the modes in sequential order from principal singular value to least significant singular value.*

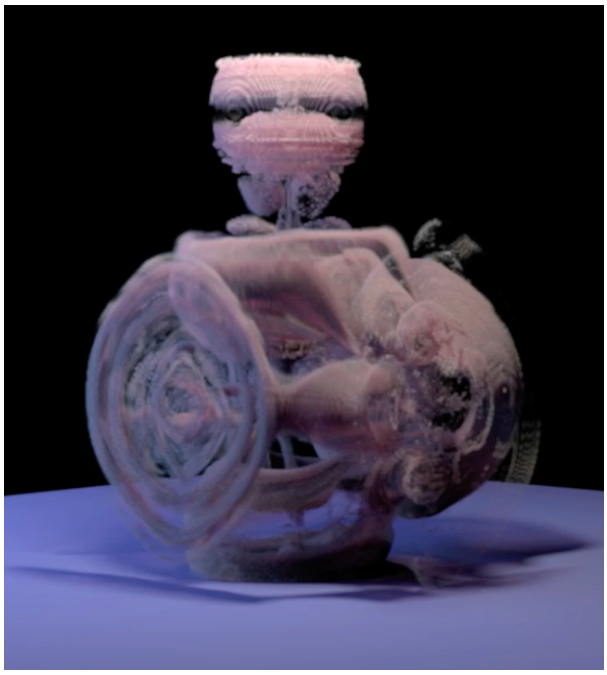


Figure 5.8: *A still frame from a permutation of the modes, forming a melody.*

amplitude vector **a** with component a_i then tells us the strength with which each corresponding frequency f_i will resonate. The nature of the input sound also strongly influences the resulting timbre: noise creates a more atmospheric effect, while impulses can create driving rhythmic textures. To match the fluid feel of the smoke propagation, we use broadband noise for the primary sound examples in this paper.

5.7 Time Evolution

Static sound, while interesting as a new timbre for a few seconds, eventually grows stale. Thus, we would like to capture the time evolution of the subspace trajectory as a dynamic sonic event. This can be achieved by cycling through the the sequence of amplitude vectors $\mathbf{a}_1, \ldots, \mathbf{a}_T$ corresponding to the subspace trajectory $\mathbf{q}_1, \ldots, \mathbf{q}_T$. Using subtractive synthesis, as described above, we can generate a corresponding sound signal that changes subtly at each time step t. As previously discussed, dif-

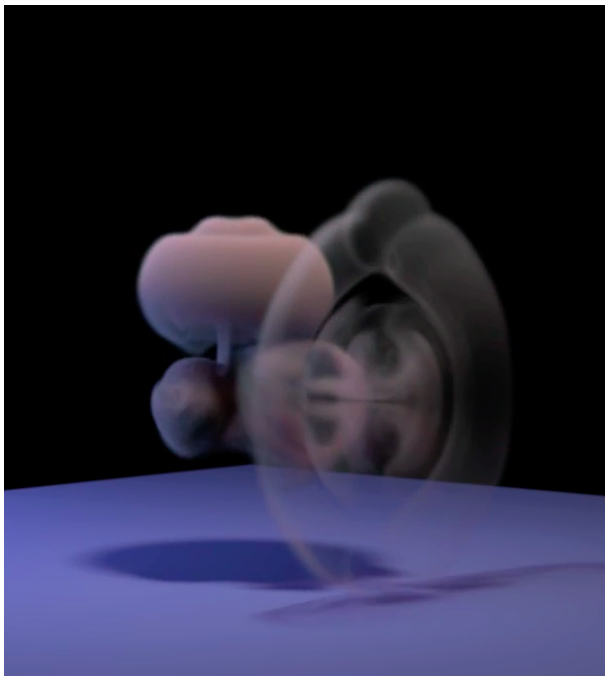


Figure 5.9: *A still frame from the reduced equations of motion.*

ferent trajectories through the subspace generate different sequences of amplitude vectors. Musically, the unfolding of these trajectories over time occurs on a micro-scale, but the overall effect is perceived smoothly, much as the individual frames of a video fuse together into a continuous motion.

Experimentally, we have explored several different categories of subspace trajectories. In the third supplemental video[5], we gradually activate each eigenvector in sequence and push the resulting subspace vector through the reduced-space Navier-Stokes equations. (Figure 5.9 shows a single frame from this video for reference.) In the final supplemental video[6], referenced earlier in §5.4, we put aside the equations of fluid flow and walk the subspace vector $\mathbf{q}$ across the surface of a sphere in $\mathbb{R}^T$. (See Figure 5.5.)

[5] *Reduced*: `https://youtu.be/_DXRQZYLcdE`
[6] *Sphere*: `https://youtu.be/y6zraBpH5PA`

5.8 Conclusion

In this chapter, have performed a preliminary aesthetic exploration of the visual and sonic patterns that arise from a generalized form of the Chladni plate experiment, in the spirit of computer-generated cymatics. Instead of studying the eigenvectors of a rigid vibration, we computed the empirical eigenvectors associated with fluid flow. The resulting forms are visually striking, and the corresponding spectrum can be sonified to produce a promising compositional palette. Suspending the underlying physics and performing an abstract subspace traversal produces intriguing results, which suggest many possible avenues for future artistic work. The interplay between the audio and the visual shapes seems to form a compositional palette that is worthy of further exploration, as pursued in Chapter 6.

Chapter 6

Compositional explorations of fluid subspaces

With our sonification system from Chapter 5 in hand, we now delve into its implications for composition. The process of musical composition, at its core, focuses on different organizations of time. Commonly, we organize our thoughts and experiments on different timescales: the sound object level, which governs very short durations such as individual notes, the mesolevel, which is an intermediate scale which governs larger groups such as phrases or themes, and the macrolevel, which governs the overall form and structure of a piece of music. The idea of conceptualizing the organization of music into three hierarchical time levels is especially emphasized in the works of Heinrich Schenker, whose theory of Schenkerian analysis works to unite the structure of musical pieces on the scales of foreground, middleground, and background [80]. Further subdivision of time scales have been noted by Curtis Roads, who articulates a total of nine time scales, ranging from the infinite down to the infinitesimal [81]. While this more broad taxonomy is conceptually useful, in practice, composing a piece whose duration is only several minutes does not typically require

working at time scales broader than our stated macro level. Additionally, unless the compositional process includes the use of "microsound," or sounds that "extend down to the threshold of auditory perception," we will also not need to work at time scales shorter than the sound object level. Hence, for our purposes, conceptualizing music in terms of three different time scales shall suffice.

6.1 Mode Isolation

The simplest compositional parameter of interest is the activation or deactivation of individual modes. While a physically accurate subspace re-simulation will, in general, at each time step, require a linear combination of each of the $r = 150$ modes, it is also possible to evolve the velocity fields over time according to algorithmic rules rather than physics-based rules. We imagine the r modes as a sort of configuration space, so that the corresponding r weights map to a particular spatial and aural phenomenon. As such, the most elementary experiment is the sequential activation of one mode at a time, creating a corresponding fluid shape of "vibration" which is mapped to its related "frequency" according to the system explained in Chapter 5. A low-[1] medium-[2] and high-frequency mode[3] are each shown isolated in the referenced videos.

Because each mode in isolation can be regarded as a musical note, this strategy allows us to carry out musical composition on the note level, producing simple melodies. Without a principled way to choose a rhythm, however, we are left only to general aesthetic considerations. An example of such a mode-based melody[4] is demonstrated in the referenced video. (Figure 5.8 shows a single frame from the

[1]Low: `https://www.youtube.com/watch?v=CAoQLYr8doE`
[2]Medium: `https://www.youtube.com/watch?v=Vwpi6U7AD5A`
[3]High: `https://www.youtube.com/watch?v=o0UtONgtpFo`
[4]Melody: `https://www.youtube.com/watch?v=N6fzJXbn2ts`

video for reference.)

6.2 Mode Superposition

The next experiment to try is the superposition of modes, creating mixtures of the modal vibration shapes in the spatial domain and harmonies in the audio domain. Again, the physics-based time evolution governs a complex coupling of the modal weights that resists simple exploration, so we turn to simple algorithmic rules to better understand the system. We can see the result of mixing a low-, medium-, and high-frequency mode with equal proportions.

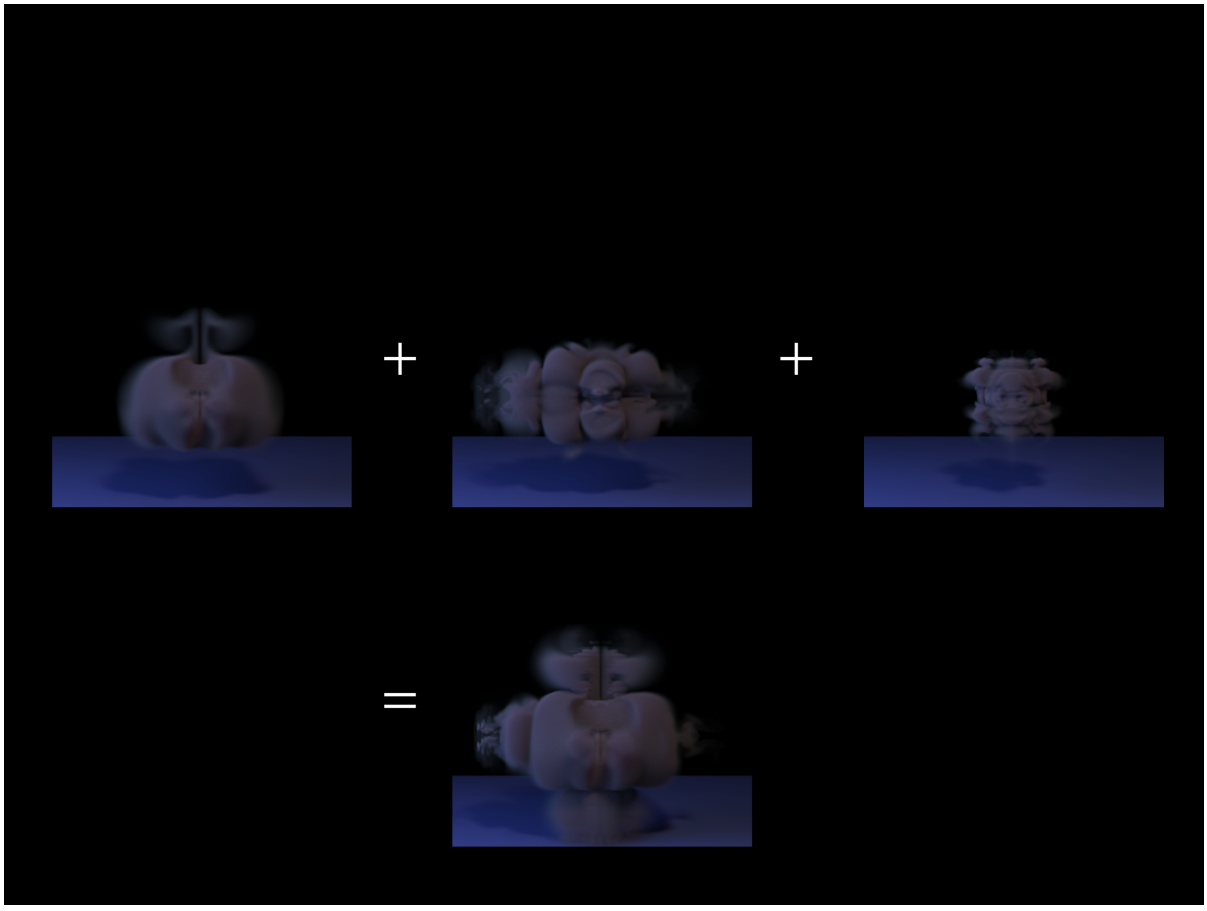


Figure 6.1: *Three individual modes in superposition produce a mixed modal shape.*

6.3 Dynamic Control

If we define the *energy* of a fluid snapshot in time as the L_2 norm of the vector of modal weights, then we see that as time unfolds, the energy waxes and wanes according to the strength of the various modal weights. Accordingly, the sound increases or decreases in overall spectral density and loudness based on the same principle. However, by forgoing the physics-based time-evolution, we can harness direct spectral control over the visuals and sound, producing simple and pleasing audiovisual gestures such as crescendo, diminuendi, and swells. An accent followed by diminuendo[5], and a crescendo-diminuendo swell[6] are each shown in the referenced videos, accompanied by still frames in Figures 6.2 and 6.3, respectively.

[5]Accent: `https://www.youtube.com/watch?v=0An95mF3Yk0`
[6]Swell: `https://www.youtube.com/watch?v=zUkpNKpkwP4`

Figure 6.2: *A still frame from the accent followed by diminuendo video.*

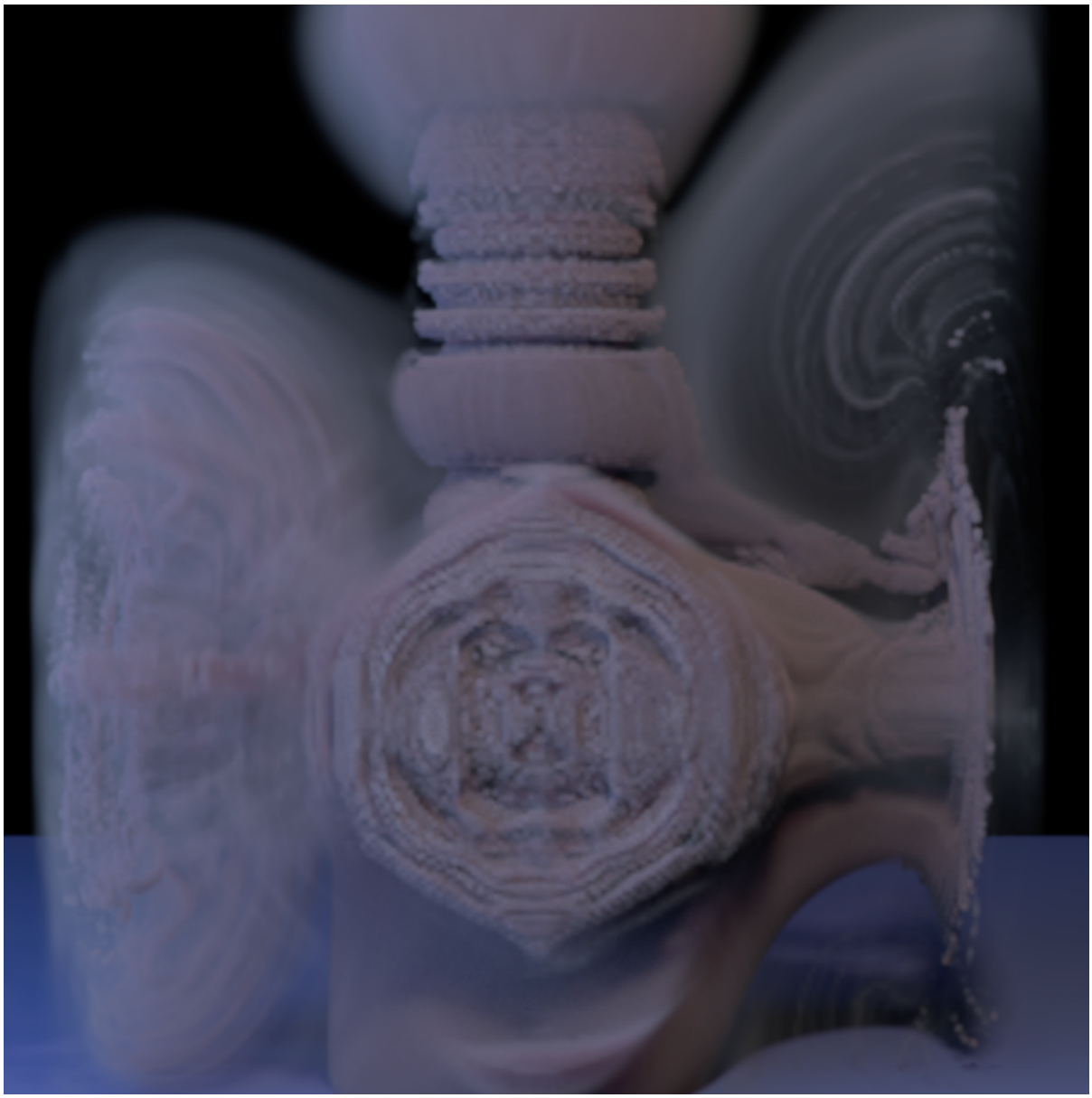


Figure 6.3: *A still frame from the crescendo-diminuendo swell video.*

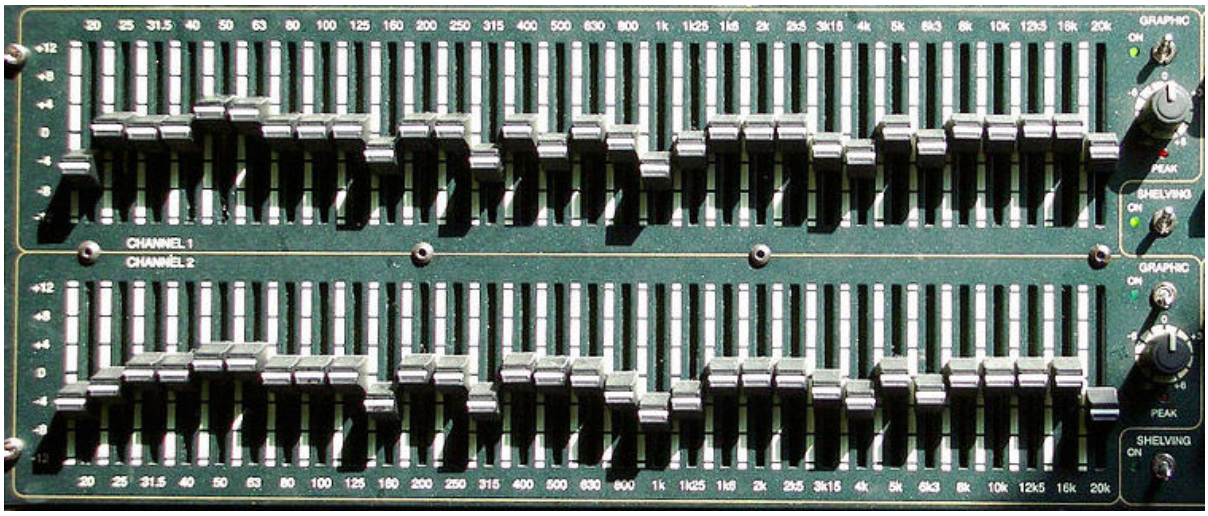


Figure 6.4: *Each of the $r = 150$ modes can be controlled individually, analogous to a set of equalization faders. Source: Wikimedia Commons.*

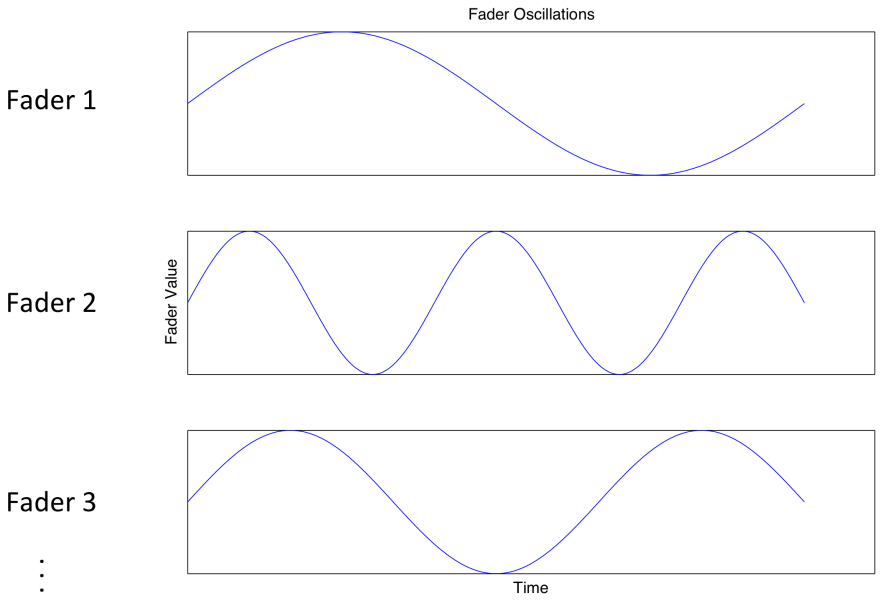
6.4 Mode Coupling and Envelopes

The simultaneous activation of all modes produces a spectrally rich sound quality, or timbre[7], not dissimilar from noise. However, more refined filtering leads to a sparser spectrum, yielding a sound quality similar to musical chords. This filtering can be achieved simply by activating only a small subset of the $r = 150$ modes at once, much as in the mode superposition study. However, such a chord is static over time, creating a dull musical effect for any prolonged duration. We can inject extra life into these chords by modulating their *envelopes*—i.e., creating time-varying amplitude envelopes around each mode. This strategy is akin to slowly adjusting the different fader knobs from high to low at different speeds. A simple mathematical collection of envelopes are sinusoids at different frequencies,[8] as the referenced video example illustrates. (Figure 6.6 shows a single frame from this video for additional

[7] According to ANSI 1960, "Timbre is that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar. [. . .] Timbre depends primarily upon the spectrum of the stimulus."

[8] Oscillation: `https://www.youtube.com/watch?v=6nEVXFkWSZ4`

reference.) Visually, the shapes morph in and out of a slow cycle of superpositions, while the sound fades in and out of subtle frequencies of an overall chord.



106

Figure 6.5: *Each modes's fader knob can be modulated smoothly over time, creating spectrally-varying envelopes.*

Figure 6.6: *A single frame from the oscillation video generated by using time-varying sinusoidal envelopes of a chord.*

We can also *couple* different modes together, or transfer from one collection of modes smoothly into another, in analogy to a musical chord progression. (Coupling the modes is also inspired by the complex nonlinear coupling induced by the real physical phenomenon of advection that we are simulating.) As time evolves, both

the amplitude envelopes and the spectral content itself shifts, producing a complex visual and audio effect[9] as seen in the referenced video example. (Figure 6.7 shows a single frame from this video for additional reference.)

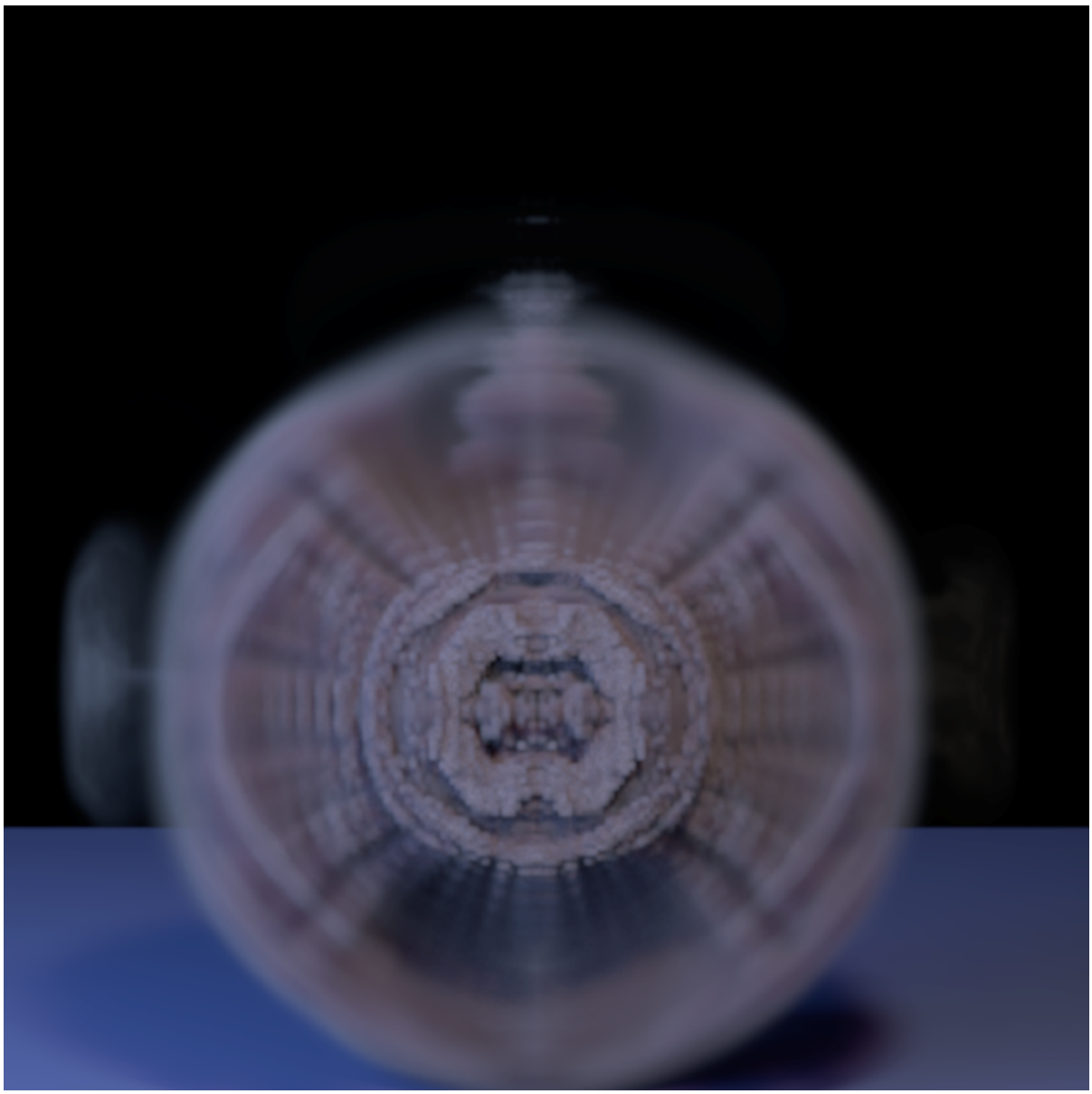


Figure 6.7: *A single frame from the crossfade video generated by modulating both the amplitude envelopes and spectral content.*

[9]Crossfade: `https://www.youtube.com/watch?v=RCIuZaWqvds`

6.5 Sonification Choices

Now that we have seen the potential of our sonification system in action, we return to the justification of the many aesthetic choices made in designing it. We have an overwhelming freedom of choice here. In particular, there is the choice of the physical training set, the choice of mapping between the dynamical system and the audio, and the choice of aesthetic stance. In the following sections, we will consider these issues in detail.

6.5.1 Training Data

Because our system relies on the empirical eigenvectors of a set of training data, the first choice we make in designing our system is the selection of the training data. An insufficiently rich training set will generate a series of results that are limited in dynamic variety due to the nature of the subspace algorithm. Early experiments in this direction were unsuccessful, as the collection of eigenvectors formed results that almost exactly mimicked the full-space simulation from which they were generated. Thus, the idea of using a sequence of full-space simulations was a natural attempt to mitigate this issue. However, since the eigenvectors are agnostic to time, it is useful to construct a sequence of simulations that are very distinct from one another. Hence, our strategy of modifying the direction of buoyancy was appealing. This technique results in six different outward-facing plumes, each of which having its own unique dynamics. The inherent symmetry to this strategy is also appealing from an aesthetic standpoint, as by privileging no one direction over another, the eigenvectors acquire a more abstract quality, detaching them from the more scientifically-grounded full-space simulation. As a result, the empirical eigenvectors take on a series of aes-

thetically pleasing turbulent forms, ideally suited for generating novel audiovisual compositions.

6.5.2 Sound Synthesis

As discussed in Chapter 2, the details in implementing a system of sonification allow for a wide freedom of choice. Hence, it is important to justify our choices by perceptual or aesthetic concerns. We identify and discuss several of our choices in this section.

Perhaps the choice that influences most the character of our auditory results is our choice for sound synthesis. No matter what mapping we choose from physics to sound, the engine which produces actual audible sound influences the output tremendously. Thus, our choice of using subtractive synthesis of spectrally rich noise deserves further investigation. Indeed, our initial concept utilized a simpler system of additive synthesis. However, we found the subtractive synthesis more suitable to our aesthetic needs for several reasons.

Firstly, our system produces not just sound, but sound and visuals. Aesthetically speaking, we desired a reasonable congruence between the character of both the audio and the visual domain. In particular, although the dynamics of the visuals varied substantially, the overall look and feel did not—the simulations were all fluid dynamics of smoke billowing about smoothly. Thus, we immediately discarded many audio results that were choppy and robotic, as the resulting incongruence was displeasing aesthetically. We also discarded a quantized scale such as 12-tone equal temperament, feeling that its rigidity would belie the abstract amorphous quality of our visuals. Therefore, what remained to explore were ambient, textural sounds. Both additive and subtractive synthesis seemed promising, as we had in hand a collection of frequencies from our mapping scheme. While additive synthesis is simple and

effective, its results tend to be too sterile for our taste. The individual sinusoids are too digitally clean, and their superpositions result in a computer-like purity that also belies the messy, turbulent swirls of our visuals. Hence, we turned our attention to subtractive synthesis. As discussed in Chapter 2, in some sense, subtractive synthesis is a logical choice, as the entire concept of modes of vibration that our sonification system depends on is built on the foundation of subtractive synthesis. Moreover, by varying the input signal to the filter, a variety of different timbres can be generated. In contrast, additive synthesis produces one single result, which must then be additionally filtered and adjusted to produce pleasing outputs. We settled on using gray noise, which emphasizes the lower frequencies in its spectral output, to bring out the principal singular values more prominently [82]. It must be mentioned that the choice itself to map the principal singular values to lower frequencies—that is, a choice of polarization, as discussed in Chapter 2, is itself somewhat arbitrary. However, this choice rests on perceptual considerations, not aesthetic ones. The prominent sonic tones in any timbre are the fundamental frequencies, which are the lowest frequencies; hence, inverting the singular values to match this prominence is a natural and justifiable step to take.

6.5.3 Real-Time and Non-Real-Time Strategies

One of the potential drawbacks of our audiovisual system is its heavy computation cost. As a result, the capacity for real-time interaction is infeasible as of time of writing in 2017. However, the dynamics of non-real-time composition allow for a different type of compositional process than a real-time system would afford. In this section, we discuss a few pros and cons of each of these types of systems, illustrating our non-real-time approach while suggesting a roadmap to eventual real-time strategies.

By its nature, a non-real time system lends itself better toward careful structural planning. Pieces must be planned and executed ahead of time rather than as a flash of inspiration. Hence, formal structures flourish, and strategies for unifying a piece on different time scales become very attractive. The possibility of unifying the material of a piece in the micro-, meso-, and macro-time scales is aesthetically powerful, and only possible through non-real-time composition. As a result, pieces that are designed with a non-real-time process tend to be structurally sound and aesthetically coherent.

However, the high degree of structure of non-real-time strategies can also be a weakness. The capacity for free improvisation and interaction through real-time often produces unexpected and daring results. Feedback through interaction with the system and having the system "interact back" can be extremely rewarding, leading to new insights and promising directions that would otherwise be impossible to discover. Additionally, the ability of the audience to participate in a piece through interaction changes the very nature of the piece. A viewer/listener no longer must let the experience wash over them; instead, they participate actively in the creation of the piece themselves, defining their own new experience which is distinct from all other viewers' experiences.

Ultimately, the non-real-time strategies employed in this dissertation proved effective, as the audiovisual studies were only possible from designing a careful strategy of moving through the subspace according to either mathematical or physical rules. However, the capacity for a real-time interaction to generate unexpected new insights and results from this system remains to be seen, and is a promising direction for future work. Interaction could happen on many levels. For example, the interaction could adjust physical parameters, by introducing new obstacles to the simulation or adjusting the buoyancy or vorticity of the flow. Alternatively, the interaction could

be more gestural, allowing the user to control the energy of the system, creating dynamical swells and frequency sweeps as demonstrated by the earlier studies in this section.

6.5.4 Aesthetics

In our creation of these studies, we imposed our own aesthetic values to some degree on the sonification system to produce musical results. While some sonifications remain purely scientific, created only for the purpose of identifying patterns in data, our own system was always born out of a desire to create musical outputs. The tension between adhering to fixed scientific principles and generating aesthetically interesting musical output is a constant challenge in composing generative music of this kind. Our aesthetic take, as discussed in Chapter 2, is a middleground between these two extremes. While a system itself can be aesthetically pleasing in the mathematical sense, this is no guarantee of the quality of its musical output. Conversely, a total abandonment of the systematic approach simply for the reasons of adjusting the musical output largely negates the point of composing using such a system in the first place. To that end, our musical adjustments remained of a minor quality, not abandoning the logic of the system, but merely adjusting it in certain cases to compensate for the quality of the musical output. For example, when we take our amplitude mapping from the numerical data of the simulation, all sign information is neglected, as a positive and negative amplitude in sound merely represents an unpleasant phase distortion. While this approach arguably throws away a dimension of the physical data, it does not fundamentally detach the result from the system, and is thus considered acceptable from our aesthetic point of view.

One of our principal aesthetic goals toward future work is the idea of simulating smoke under more fanciful systems of time-evolution than standard physics. Hence,

many of our examples play with the idea of controlling the time evolution in different mathematical procedures rather than sticking with the correct physics. While this again is pushing the boundaries of our original system, we nonetheless retain the same link between the visual and audio results. Thus, we view this form of modulation as a robust way to explore new compositional forms. Changing the underlying rules which govern the time-to-time behavior of the audiovisual system is not the same as abandoning the system; in fact, it has proved fruitful in generating novel results. These strategies move us more in the direction of simulating fluids "as they might be" vs. fluids "as they actually are."

Chapter 7

Conclusions and future work

In this dissertation, we strove to develop a generative audiovisual compositional language. Toward this end, we presented a sonification system for computational fluid dynamics based on the method of subspaces. Additionally, we devised a data compression algorithm that alleviates the memory costs of subspace algorithms by an order of magnitude. Through systematic experimentation and exploration, our sonification system was tested and explored, producing a series of audiovisual etudes. The data compression algorithm was tested on a variety of different types of scenes, perturbations, and compression levels, demonstrating its general robustness and effectiveness. Continuing research in the direction of the sonification system is an important step in the development of new audiovisual grammars in media art, while further work in the area of data compression represents an important step toward making subspace simulations more computationally practical.

7.1 Summary of Results

We have devised a system of sonification of computational fluid dynamics by considering the empirical eigenvectors and eigenvalues of a subspace simulation. Based loosely on the ideas of Chladni plates and the general conception of cymatics, we map the characteristic resonant modal shapes of the fluid velocity field bases to their corresponding audio frequencies, producing a sound signal. The choice of a model-based sonification strategy allows us to unfold our visuals and sounds over time within the framework of the system. Hence, the natural sequence of fluid velocity fields as governed by the subspace Navier-Stokes equations determines a natural time progression of dynamic visual forms and sonic content.

Several sound synthesis strategies and transformations had to be considered. We carefully mapped the raw singular values into frequencies, choosing an intuitive polarization as well as a ratio-preserving transformation and offset to keep the values within the limits of human hearing. Following the spirit of physical modal vibrations, we selected a subtractive synthesis technique, in which a filterbank of resonant filters at the corresponding modal frequencies were excited in various strengths and decay times. Although many possible input signals could be used with the filter bank, we chose to use noise, as it was both spectrally rich and matched the aesthetic feel of fluid flow.

Our audiovisual etudes each explored different combinations of modal excitations. By isolating individual modes, superimposing them, and cross-fading between them, we demonstrated the versatility and compositional usefulness of having spectral control in both the spatial and audio domain. The gestures of crescendo and diminuendo were also performed by controlling the overall energy of the system, demonstrating the potential for different musical articulations. Finally, the general

principle of constructing time-evolving paths ungoverned by physics illustrated an important compositional possibility of working with fluids and sounds more fancifully, leading to the idea of smoke "as it might be" as opposed to smoke "as it is."

The data compression technique developed in Chapter 4 serves to address the memory pressure of subspace simulations. In an analogy to the JPEG compression scheme, we use a DCT-based transform compression algorithm to represent the subspace basis vectors in the Fourier domain, with the intuition that the energy at the higher frequencies can be dampened or discarded without distorting the original data greatly. Our algorithm, while intuitively based on JPEG, made several important alterations that were necessary for fluid data, including adapting the process to three dimensions, systematically generating damping quality matrices, performing an energy-based per-block quality selection, and devising a new zigzag scan. In addition, since naïve reconstruction during the decompression stage incurred heavy computational costs, it largely negated the advantage of the subspace approach to begin with. We circumvented this problem by devising a novel fast sparse frequency-domain reconstruction that enabled the algorithm to reduce memory costs without severely increasing time costs.

7.2 Limitations

There are several drawbacks to our current sonification and audiovisual system. The most challenging is that of computational speed. While such a system would ideally run in real time, allowing not only for simple compositional feedback and iteration but also for user interaction, both the simulation and rendering times as of 2017 are extremely far away from these speeds. This means that even short compositions

can take hours or days to generate at high visual quality. However, the restriction of working in non-real time does force the composer to make careful compositional choices ahead of time, producing aesthetically very different works from real-time, experimental, interactive systems.

Another basic drawback of the sonification system is inherent in the subspace method itself. Once a training set is chosen and the subspace basis is computed, simulations are locked into reproducing only dynamics that are reasonably similar to the original training set. Hence, if the user desires any significantly different set of motions or dynamics, a fresh simulation must be precomputed. While in principle this can be done repeatedly, the time costs are quite prohibitive. A more analytical approach such as using a particular mathematical basis such as Laplacian eigenfunctions is possible, but the fluid modes would then be more regular, undermining the visual variety of the training-based approach. However, the constraint of a particular space of dynamics can also be viewed as fruitful from a compositional standpoint. Indeed, limitations often drive artistic creativity. The main challenge is choosing a reasonable set of constraints in the first place.

The data compression scheme, while obtaining an order of magnitude compression, still leads to a slight increase in time costs, which may be unacceptable to a user who already has sufficient memory to compute the uncompressed subspaces. Furthermore, the system assumes a fluid simulation technique using a regular grid. Many of the techniques might generalize, but extending the technique to applications with tetrahedral meshes is still a direction for future work.

7.3 Future Work

We have explored several basic parameter modulations in our sonification system, producing a series of audiovisual etudes. However, further work in this direction could be considered. Other parameters such as fluid viscosity or buoyant forces could be modulated, leading to new insights into the interplay between form and sound. The sonifications presented in this dissertation all derived from one particular subspace training. However, we could retrain the subspace in a variety of ways, leading to the potential for new artistic expression. The introduction of obstacles would also perturb the basis functions, leading to a novel spectrum. The present rendering system remains static, presenting the smoke in each simulation with the same lighting and coloring; however, a mapping between the rendering system and sound could also be explored, yielding more variety in the visuals. Finally, more complete audiovisual works, deeper in macrostructure and musical form, remain to be seen.

In the data compression scheme, no matter what the simulation, the approach relies on using the discrete cosine basis, not leveraging the potential for more sparse representations depending on the simulation at hand. More general techniques, such as dictionary-based methods using orthogonal matching pursuit, could lead to sparser representations, and therefore both better memory compression as well as time reduction, due to the reliance of the algorithm on the sparse frequency-domain reconstruction during the decompression phase. Additionally, besides the run-length encoding step, no further lossless compression is applied to the data stream during the compressor. However, preliminary tests show that the data stream typically contains informational redundancies, implying that an additional step through entropy encoding such as Huffman or arithmetic coding could increase the compression effi-

ciency.

7.3.1 Outlook

The general strategy of using dynamical systems from mathematics and physics as an engine for generating sound and visuals shows much promise for generating novel audiovisual art. Outside of fluid dynamics, many other processes from physics such as Brownian motion, the heat equation, the wave equation, have potential for interest results. Chaotic systems such as the logistic map are fertile ground for art, as they are very sensitive to initial conditions and parameter modulation. This sensitivity means that they would generate contrasting results from slightly perturbed inputs, suggesting an interesting compositional direction of parameter perturbation. Entirely families of audiovisual pieces could be composed simply by setting up a baseline set of initial conditions and slowly modulating them.

As computing power and algorithms become more robust in the future, real-time interaction with physics-based audiovisual systems will become feasible. Interaction adds an extra layer of richness into the system. The user can modulate parameters in real-time and immediately see and hear the results, setting up a powerful feedback loop. Instead of composing using pre-planned compositional gestures, the system can be a fertile playground for improvisation and experimentation. It will also be possible to simulate both "realistic" results using the correct physics as well as more fanciful results. Users can explore even further into dynamical systems "as they might be" and the resulting audiovisual art.

The system has been described so far in essence takes a physics-based "engine" and generates sound and visuals from it. However, the reverse possibility is also feasible. That is, we can start with a sequence of sounds and use them to drive the physics. Not only does this create the possibility of generating interest novel

dynamics, it also opens up the idea of feedback. A sequence of sounds generated from one physics engine can be fed into another, or even back into itself, creating unexpected results. Even a simple dynamical system is likely to create interesting patterns when fed back into itself in such a manner.

Scientifically, the strategy of representing complex data with audiovisual streams multimodally has the potential for new intuitive insight. Hence, the mapping of complex dynamical systems into sound and visuals may give scientists a more intuitive and clearer way to grasp their logic. With rigorously defined systems of sonification, it is possible that input/output pairs can easily be identified. That is, given the audiovisual result, the user intuitively understands the underlying process that must have generated that result. If a sonification system can be understood on this level, then the representation of data through multiple modes can be highly desirable, as it allows scientists to think and perceive the data from several different levels of understanding.

Ultimately, a systematic mapping of dynamical systems into multimodal art has many positive implications in both art and science. We hope that these disciplines can move closer together along their commonly shared ideas. Both artists and scientists have much to learn from one another. The work in this dissertation is just one stepping stone toward a unified language of art and science, but we believe that the future is bright for this hybridized field.

Bibliography

[1] M. Gad-el Hak, "Fluid mechanics from the beginning to the third millennium," *International Journal of Engineering Education*, vol. 14, no. 3, pp. 177–185, 1998.

[2] I. Xenakis, *Formalized music: thought and mathematics in composition*. No. 6, Pendragon Press, 1992.

[3] K. Stockhausen and E. Barkin, "The concept of unity in electronic music," *Perspectives of New Music*, pp. 39–48, 1962.

[4] M. V. Mathews, "The digital computer as a musical instrument," *Science*, vol. 142, no. 3592, pp. 553–557, 1963.

[5] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the audio engineering society*, vol. 21, no. 7, pp. 526–534, 1973.

[6] J. Cage, *Silence: lectures and writings*. Wesleyan University Press, 2011.

[7] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *The physics of fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[8] R. A. Gentry, R. E. Martin, and B. J. Daly, "An eulerian differencing method for unsteady compressible flow problems," *Journal of Computational Physics*, vol. 1, no. 1, pp. 87–118, 1966.

[9] J. L. Hess and A. O. Smith, "Calculation of potential flow about arbitrary bodies," *Progress in Aerospace Sciences*, vol. 8, pp. 1–138, 1967.

[10] J. F. O'brien and J. K. Hodgins, "Dynamic simulation of splashing fluids," in *Computer Animation'95., Proceedings.*, pp. 198–205, IEEE, 1995.

[11] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, "A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method)," *Journal of computational physics*, vol. 152, no. 2, pp. 457–492, 1999.

[12] R. Bridson, J. Houriham, and M. Nordenstam, "Curl-noise for procedural fluid flow," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 46, 2007.

[13] J. Stam, "Real-time fluid dynamics for games,"

[14] J. U. Brackbill, D. B. Kothe, and H. M. Ruppel, "Flip: a low-dissipation, particle-in-cell method for fluid flow," *Computer Physics Communications*, vol. 48, no. 1, pp. 25–38, 1988.

[15] R. Ando, N. Thürey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," *ACM Trans. Graph.*, vol. 32, pp. 103:1–103:10, July 2013.

[16] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 104, 2014.

[17] T. Pfaff, N. Thuerey, and M. Gross, "Lagrangian vortex sheets for animating fluids," *ACM Trans. Graph. – SIGGAPH 2012 Papers*, vol. 31, pp. 112:1–112:8, July 2012.

[18] A. Lucier, "Origins of a form: Acoustical exploration, science and incessancy," *Leonardo Music Journal*, pp. 5–11, 1998.

[19] G. Kepes and A. Feininger, "The new landscape in art and science," 1958.

[20] A. Lauterwasser, *Wasser-Klang-Bilder:" die schöpferische Musik des Weltalls"*. AT Verlag, 2003.

[21] A. Treuille, A. Lewis, and Z. Popović, "Model reduction for real-time fluids," *ACM Transactions on Graphics*, vol. 25, pp. 826–834, July 2006.

[22] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. University of Illinois press, 1998.

[23] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[24] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[25] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[26] M. I. S. ISO, "Iec 11172: Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1, 5 mbit/s," *Part1: Systems, Part2: Video, Part3: Audio*, 1993.

[27] D. Le Gall, "Mpeg: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[28] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation," in *Computer Graphics (Proceedings of SIGGRAPH 89)*, pp. 215–222, July 1989.

[29] J. Seo, G. Irving, J. P. Lewis, and J. Noh, "Compression and direct manipulation of complex blendshape models," *ACM Trans. Graph.*, vol. 30, pp. 164:1–164:10, Dec. 2011.

[30] T. R. Langlois, S. S. An, K. K. Jin, and D. L. James, "Eigenmode compression for modal sound models," *ACM Trans. Graph.*, vol. 33, pp. 40:1–40:9, July 2014.

[31] T. Hermann, A. Hunt, and J. G. Neuhoff, *The sonification handbook*. Logos Verlag Berlin, 2011.

[32] G. Kramer, B. Walker, T. Bonebright, P. Cook, J. H. Flowers, N. Miner, and J. Neuhoff, "Sonification report: Status of the field and research agenda," 2010.

[33] M. A. Boden and E. A. Edmonds, "What is generative art?," *Digital Creativity*, vol. 20, no. 1-2, pp. 21–46, 2009.

[34] C. Roads, *Composing electronic music: a new aesthetic*. Oxford University Press, USA, 2015.

[35] J. Schillinger, "The schillinger system of musical composition," 1949.

[36] D. Lewin, *Generalized musical intervals and transformations*. Oxford University Press, USA, 2010.

[37] J. Demers, *Listening through the noise: the aesthetics of experimental electronic music*. Oxford University Press, 2010.

[38] M. A. Day, "The no-slip condition of fluid dynamics," *Erkenntnis*, vol. 33, no. 3, pp. 285–296, 1990.

[39] Y. Katznelson, *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[40] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.

[41] M. Babbitt, "Who cares if you listen?," *High Fidelity*, vol. 8, no. 2, pp. 38–40, 1958.

[42] E. Křenek, *Music here and now*. WW Norton, 1939.

[43] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge university press, 2000.

[44] I. G. Currie, *Fundamental mechanics of fluids*. CRC Press, 2012.

[45] H. Lamb, *Hydrodynamics*. Cambridge university press, 1932.

[46] J. Stam, "Stable fluids," in *SIGGRAPH 1999*, pp. 121–128, 1999.

[47] A. J. Chorin, J. E. Marsden, and J. E. Marsden, *A mathematical introduction to fluid mechanics*, vol. 3. Springer, 1990.

[48] R. Bridson, *Fluid simulation for computer graphics*. CRC Press, 2015.

[49] D. W. Peaceman and H. H. Rachford, Jr, "The numerical solution of parabolic and elliptic differential equations," *Journal of the Society for industrial and Applied Mathematics*, vol. 3, no. 1, pp. 28–41, 1955.

[50] J. Stam, "A simple fluid solver based on the fft," *Journal of graphics tools*, vol. 6, no. 2, pp. 43–52, 2001.

[51] J. Lumley, "The structure of inhomogeneous turbulent flows," *Atmospheric turbulence and radio wave propagation*, pp. 166–178, 1967.

[52] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual Rev. Fluid Mech*, pp. 539–575, 1993.

[53] J. Barbič and D. L. James, "Real-time subspace integration for st. venant-kirchhoff deformable models," in *ACM transactions on graphics (TOG)*, vol. 24, pp. 982–990, ACM, 2005.

[54] "Synthesizing sounds from physically based motion," in *ACM SIGGRAPH 2001 Video Review on Animation Theater Program*, SVR '01, (New York, NY, USA), pp. 59–, ACM, 2001. Director-O'Brien, James F.

[55] T. Kim and J. Delaney, "Subspace fluid re-simulation," *ACM Trans. Graph.*, vol. 32, pp. 62:1–62:9, July 2013.

[56] K. Carlberg, C. Bou-Mosleh, and C. Farhat, "Efficient non-linear model reduction via a least-squares petrov-galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, vol. 86, no. 2, pp. 155–181, 2011.

[57] S. S. An, T. Kim, and D. L. James, "Optimizing cubature for efficient integration of subspace deformations," *ACM Trans. on Graphics*, vol. 27, p. 165, Dec. 2008.

[58] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992.

[59] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1995.

[60] R. Bro and S. De Jong, "A fast non-negativity-constrained least squares algorithm," *Journal of chemometrics*, vol. 11, no. 5, pp. 393–401, 1997.

[61] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, "An efficient construction of reduced deformable objects," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 213, 2013.

[62] Z. Pan, H. Bao, and J. Huang, "Subspace dynamic simulation using rotation-strain coordinates," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 242, 2015.

[63] A. D. Jones, P. Sen, and T. Kim, "Compressing fluid subspaces," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '16, (Aire-la-Ville, Switzerland, Switzerland), pp. 77–84, Eurographics Association, 2016.

[64] M. Wicke, M. Stanton, and A. Treuille, "Modular bases for fluid dynamics," *ACM Trans. on Graphics*, vol. 28, p. 39, Aug. 2009.

[65] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965. URL: `http://cr.yp.to/bib/entries.html#1965/cooley`.

[66] B.-L. Yeo and B. Liu, "Volume rendering of DCT-based compressed 3D scalar data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 29–43, Mar. 1995.

[67] S. Guthe, M. Wand, J. Gonser, and W. Straßer, "Interactive rendering of large volume data sets," in *IEEE Visualization*, pp. 53–60, 2002.

[68] M. Treib, K. Bürger, F. Reichl, C. Meneveau, A. Szalay, and R. Westermann, "Turbulence visualization at the terascale on desktop PCs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 2169–2177, December 2012.

[69] T. de Witt, C. Lessig, and E. Fiume, "Fluid simulation using Laplacian eigenfunctions," *ACM Trans. Graph.*, vol. 31, no. 1, pp. 10:1–10:11, 2012.

[70] B. Long and E. Reinhard, "Real-time fluid simulation using discrete sine/cosine transforms," in *Symposium on Interactive 3D graphics and games*, pp. 99–106, ACM, 2009.

[71] K. Sayood, *Introduction to Data Compression*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 4 ed., 2012.

[72] B. Liu, G. Mason, J. Hodgson, Y. Tong, and M. Desbrun, "Model-reduced variational fluid simulation," *ACM Trans. Graph.*, vol. 34, pp. 244:1–244:12, Oct. 2015.

[73] R. Bridson, *Fluid Simulation for Computer Graphics*. CRC Press, 2015.

[74] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[75] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[76] M. Stanton, Y. Sheng, M. Wicke, F. Perazzi, A. Yuen, S. Narasimhan, and A. Treuille, "Non-polynomial Galerkin projection on deforming meshes," *ACM Trans. Graph.*, vol. 32, pp. 86:1–86:14, July 2013.

[77] J. K.-M. Aaron Demby Jones and T. Kim, "Seeing and hearing the eigenvectors of a fluid," in *Proceedings of Bridges 2017: Mathematics, Art, Music, Architecture, Education, Culture* (C. H. S. David Swart and K. Fenyvesi, eds.), (Phoenix, Arizona), pp. 305–312, Tessellations Publishing, 2017. Available online at `http://archive.bridgesmathart.org/2017/bridges2017-305.pdf`.

[78] D. Ryckelynck, "A priori hyperreduction method: an adaptive approach," *Journal of Computational Physics*, vol. 202, no. 1, pp. 346–366, 2005.

[79] S. Rosen and P. Howell, *Signals and systems for speech and hearing*, vol. 29. Brill, 2011.

[80] A. C. G. Cadwallader, D. A. Cadwallader, and D. Gagné, *Analysis of tonal music: a Schenkerian approach*. No. Sirsi) i9780195301762, 2007.

[81] C. Roads, *Microsound*. MIT press, 2004.

[82] S. Wilson, D. Cottle, and N. Collins, *The SuperCollider Book*. The MIT Press, 2011.