Spectral Exterior Calculus and Its Implementation

Thesis by
Dzhelil S. Rufat

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2017
Defended April 5, 2017

Dzhelil S. Rufat
ORCID: 0000-0001-8766-2338

To my mother Bakie.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Mathieu Desbrun for his mentorship and guidance, without which this piece of work would not have been possible.

I would also like to acknolwedge the contribution of the late Jerrold Marsden who first introduced me to the wonders of differential geometry and symmetry in mechanics, first as a teacher and then briefly as an advisor.

I am grateful to Tim Colonius for his guidance in numerical fluid mechanics.

I am thankful to Yiying Tong, Joris Vankerschaver, Patrick Mullen, and Gemma Mason for their suggestions and collaboration.

Additionally, the support and encouragement of my friends Daniel Sandoval, Vaclav Cvicek, Nikola Kamburov, Silviu Pufu, Ana Caraiani, and Andreas Hoenselaar have been invaluable to me during the years.

I also want to extend my deepest appreciation to staff Kevin Austin, Linda Schutz, Alice Sogomonian, Divina Bautista, Sheila Shull, and Maria Lopez.

Last but not least, I want to say thank you to my mother Bakie, my father Sabahatin, and my sister Refie for their love and unwavering dedication to me.

ABSTRACT

Preserving geometric, topological and algebraic structures at play in partial differential equations has proven to be a fruitful guiding principle for computational methods in a variety of scientific fields. However, structure-preserving numerical methods have traditionally used spaces of piecewise polynomial basis functions with local support to interpolate differential forms. When solutions are known to be smooth, a spectral treatment is often preferred instead as it brings exponential convergence. While recent works have established spectral variants of *discrete exterior calculus*, no existing approach offers the full breadth of exterior calculus operators and a clear distinction between vectors and covectors. We present such a unified approach to *spectral exterior calculus* (SPEX) and provide detail on its implementation. Notably, our approach leverages Poincaré duality through the use of a primal grid and its dual (with a natural handling of boundaries to facilitate the treatment of boundary conditions), and uses a twin representation of differential forms as both integrated and pointwise values. Through its reliance on the *fast Fourier transform*, the resulting framework enables computations in arbitrary dimensions that are both efficient and have excellent convergence properties.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Dzhelil Rufat et al. “The chain collocation method: A spectrally accurate calculus of forms”. In: *Journal of Computational Physics* 257 (Jan. 2014), pp. 1352–1372. DOI: 10.1016/j.jcp.2013.08.011.
Rufat participated in the conception of the project, derived the spectrally accurate basis functions, formulated the corresponding discrete operators, implemented them in code, and validated the results via extensive unit testing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The laws of physics are coordinate-independent as a consequence of general covariance. Many discretization schemes, however, rely on an arbitrary choice of a coordinate system to describe and simulate these laws. Consequently, such discretizations often suffer from spurious numerical modes and other numerical shortcomings that are artifacts of the particular computational method and do not exist in the underlying physical system.

Using the language of *exterior calculus* and *differential geometry* one can express the laws of physics in a coordinate-independent manner. Whether studying fluid mechanics, electromagnetism, wave propagation, diffusion, or advection, one can readily cast the governing laws and equations using the objects and operators of exterior calculus and differential geometry. Then, rather than discretizing the coordinate dependent equations of motion, one can instead discretize directly the intrinsic operators and objects of exterior calculus, which is precisely the approach taken by *discrete exterior calculus* (DEC).

In addition to coordinate independence, another important goal of DEC is structure preservation. DEC strives to preserve in the discrete setting as many as possible of the properties of the smooth operators from the continuous realm. This allows for important features of the continuous theory to be true in the discrete world by construction rather than as an ad hoc feature. However, many of the previous works on DEC are incomplete in terms of objects or operations that they consider, or they are low order approximations.

In this thesis, we introduce *spectral exterior calculus* (SPEX), which provides a complete hierarchy of operators that have a high degree of accuracy and that can be used to simulate a wide variety of physical problems. SPEX is a merger between well known spectral numerical methods and discrete exterior calculus in order to create a formulation that has excellent convergence properties and is fast and efficient in implementation, with operators that are discretized in a coordinate-free and structure-preserving manner.

1.1 Organization of this Thesis

- Chapter 2 reviews manifold theory, differential forms, vector fields, the operators associated with them, and their properties. All these objects and operators will be subject to discretization in the chapters that follow.

- Chapter 3 describes our first attempt at defining a spectrally accurate calculus of forms. We present some discrete spectral operators such as the Hodge star and the wedge product. The framework presented is limited to flat domains defined over either periodic or bounded intervals.

- Chapter 4 presents *spectral exterior calculus* (SPEX) with a complete hierarchy of operators defined on arbitrary curved domains that are logically rectangular. A twin representation of discrete forms is introduced, both as cochains and components, along with the ability to seamlessly map between them. Metric dependent operators are performed in the component representation by first transforming into an orthonormal frame, in which the operators have a very simple form.

- Chapter 5 provides detail of the SPEX implementation in terms of the *fast Fourier transform* and elementary array operations, describes how said implementation is validated, demonstrates its convergence, and presents results and applications to a variety of problems in physics.

Chapter 2

DIFFERENTIAL FORMS, VECTOR FIELDS, AND ASSOCIATED OPERATORS

In this chapter, we review basic manifold theory in order to introduce the objects that live on manifolds and their associated operators, that in later chapters will be subject to discretization. In particular, we will list the properties of these operators, all of which will naturally carry over to our spectrally accurate discretization.

Roughly speaking **manifolds** are topological spaces that locally resemble Euclidean spaces. Given a set $\mathcal{M}$, a chart of $\mathcal{M}$ is a bijective map $\varphi : U \to \varphi(U) \subset \mathbb{R}^n$ that takes a subset U of $\mathcal{M}$ and maps it to $(x^1, \ldots, x^d)$. We call the scalar values x^i the **coordinates** of the point $m \in U \subset \mathcal{M}$. A manifold is then a set $\mathcal{M}$ for which every point is covered by at least one chart, and charts are compatible with each other to ensure smoothness [1].

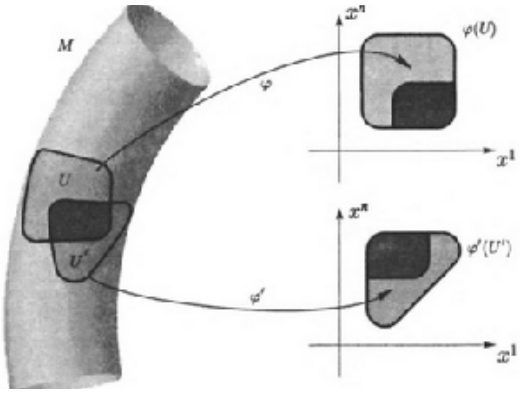


Figure 2.1: A manifold with charts.

2.1 Vectors and Forms

To each point $m \in \mathcal{M}$ we can associate a **tangent space** denoted by $T_m\mathcal{M}$. A **tangent vector** v to a manifold $\mathcal{M}$ at point m is an element of $T_m\mathcal{M}$. The values $(v^1, \ldots, v^d)$ are called the components of v with respect to some local coordinate chart. The collection of all tangent spaces forms a **tangent bundle** denoted by $T\mathcal{M}$:

$$T\mathcal{M} = \bigcup_{m \in \mathcal{M}} T_m\mathcal{M}.$$

To each tangent space $T_m\mathcal{M}$ we can associate a **cotangent space** $T_m^*\mathcal{M}$ whose elements are the linear functions $\alpha : T_m\mathcal{M} \to \mathbb{R}$ that are known as **covectors** or **1-forms**. The values $(\alpha_1, \ldots, \alpha_d)$ are known as the components of the 1-form α.

A **vector field** X on the manifold $\mathcal{M}$ is a map $X : \mathcal{M} \to T\mathcal{M}$ that assigns a vector $X(m)$ to each point m of the manifold. The space of all vector fields is denoted by $\mathfrak{X}(\mathcal{M})$. Similarly, a 1**-form field** is a map $\alpha : \mathcal{M} \to T^*\mathcal{M}$ that assigns a covector $\alpha(m)$ to each point of the manifold. The space of all 1-form fields is denoted by $\Lambda^1(\mathcal{M})$. **Scalar fields** are 0-forms, and are denoted by $\Lambda^0(\mathcal{M})$, whereas k**-forms**, which are antisymmetric tensors acting on k vectors, are denoted by $\Lambda^k(\mathcal{M})$, i.e., if $\omega^k \in \Lambda^k(\mathcal{M})$ then

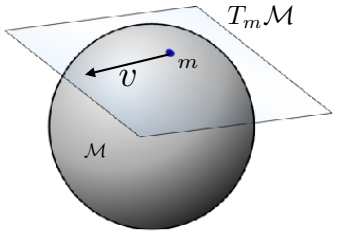


Figure 2.2: Tangent space.

$$\omega^k(m): \quad \underbrace{T_m\mathcal{M} \times \cdots \times T_m\mathcal{M}}_{k} \to \mathbb{R}.$$

Having defined the spaces of vector fields $\mathfrak{X}(\mathcal{M})$ and form fields $\Lambda^k(\mathcal{M})$, we now turn our attention to the operators that act upon them.

2.2 Exterior Derivative

The continuous exterior derivative,

$$\mathbf{d}: \quad \Lambda^k(\mathcal{M}) \to \Lambda^{k+1}(\mathcal{M}),$$

extends the notion of derivative to differential forms, turning a k-form ω^k to a $(k+1)$-form $\mathbf{d}\omega^k$. This metric-independent operator satisfies:

- **linearity**
$$\mathbf{d}(c_1\omega_1 + c_2\omega_2) = c_1\mathbf{d}\omega_1 + c_2\mathbf{d}\omega_2$$
- **nilpotency**
$$\mathbf{d}^2 = 0$$
- **locality**
$$\mathbf{d}(\omega|_U) = (\mathbf{d}\omega)|_U.$$

The exterior derivative encompasses the usual linear operators in vector calculus (gradient, curl, and divergence); its nilpotency reflects the classical identities $\nabla \cdot (\nabla \times \vec{\mathbf{v}}) = 0$ and $\nabla \times (\nabla f) = 0$. Moreover, a number of theorems (the fundamental theorem of calculus, as well as Green's, Kevin's, and Gauss's theorems) can be seen

as special cases of the general *Stokes' theorem* which states that $\mathbf{d}$ and the boundary operator $\boldsymbol{\partial}$ are dual operators, in the sense that:

$$\int_{\sigma} \mathbf{d}\omega = \int_{\boldsymbol{\partial}\sigma} \omega.$$

Fortunately, one can use the notion of chains and cochains from algebraic topology to provide discrete realization of this derivative operator [73, 49, 8, 21] that captures all its defining properties. The exterior derivative is invariant under diffeomorphisms

$$\varphi^* \mathbf{d}\alpha = \mathbf{d}\varphi^* \alpha.$$

Given its purely topological (metric-independent) definition, no special care is necessary to adapt the traditional discrete exterior derivative to our spectral framework (see Section 3.4).

2.3 Wedge Product

The continuous wedge product is also a metric-independent operator, assembling a $(k+l)$-form from a k-form and an l-form; it generalizes the notion of cross product to differential forms. The wedge operator

$$\wedge : \quad \Lambda^k(\mathcal{M}) \times \Lambda^l(\mathcal{M}) \to \Lambda^{k+l}(\mathcal{M})$$

has the following properties:

- **associativity**
$$\alpha \wedge (\beta \wedge \gamma) = (\alpha \wedge \beta) \wedge \gamma$$
- **bilinearity**
$$(a\alpha + b\beta) \wedge \gamma = a(\alpha \wedge \beta) + b(\beta \wedge \gamma)$$
$$\alpha \wedge (b\beta + c\gamma) = b(\alpha \wedge \beta) + c(\alpha \wedge \gamma)$$
- **anti-commutativity**
$$\alpha^k \wedge \beta^l = (-1)^{kl} \beta^l \wedge \alpha^k.$$

The wedge product also satisfies a Leibniz rule (also known as a product rule) with the exterior derivative

$$\mathbf{d}(\alpha^k \wedge \beta^l) = \mathbf{d}\alpha^k \wedge \beta^l + (-1)^k \alpha^k \wedge \mathbf{d}\beta^l. \tag{2.1}$$

Various discretizations of this operator have been proposed over the years, from primal-dual versions [35], to metric-independent definitions based on cup product [22, 7]. While most properties of the wedge product are retained by these discrete counterparts, associativity is only satisfied for closed forms, or in the limit of mesh refinement [40]. Additionally, the wedge product is a purely topological (metric-independent) operator that is invariant under a diffeomorphism:

$$\varphi^*(\alpha \wedge \beta) = (\varphi^*\alpha) \wedge (\varphi^*\beta).$$

This property is preserved in the discrete setting (see A.1).

2.4 Hodge Star

The continuous Hodge star operator, denoted by $\star$, is a metric-dependent operator that maps k-forms on a d-dimensional manifold to $(d-k)$-forms:

$$\star : \quad \Lambda^k(\mathcal{M}) \to \Lambda^{d-k}(\mathcal{M}).$$

The Hodge star is an **(anti-)involution** in that if it is applied twice, the result is the identity up to a sign; more precisely, on a d-dimensional manifold:

$$\star^k \star^{d-k} = (-1)^{k(d-k)}. \tag{2.2}$$

Under certain conditions, the discrete equivalent is also an involution (see A.2). In dimension two, the Hodge star operator in coordinates satisfies:

$$\star 1 = \mathbf{d}x \wedge \mathbf{d}y, \quad \star\mathbf{d}x = -\mathbf{d}y, \quad \star\mathbf{d}y = \mathbf{d}x, \quad \star(\mathbf{d}x \wedge \mathbf{d}y) = 1.$$

Along with the wedge product, the Hodge star defines a natural inner product on k-forms:

$$\alpha^k \cdot \beta^k = \star(\alpha^k \wedge \star\beta^k). \tag{2.3}$$

2.5 Contraction

The contraction (sometimes referred to as the interior product) is a metric independent operator

$$\lrcorner : \quad \mathfrak{X}(\mathcal{M}) \times \Lambda^k(\mathcal{M}) \to \Lambda^{k-1}(\mathcal{M})$$

that arises from the natural pairing of vectors and forms. It turns k-form ω^k into a $(k-1)$-form $X \lrcorner \omega^k$. Since a k-forms are linear maps that take k vectors as inputs, the contraction simply places the vector field X into the first slot of the k-form to reduce its degree by one:

$$(X \lrcorner \alpha^k)(v_2, \ldots, v_k) = \alpha^k(X, v_2, \ldots, v_k).$$

The contraction is **left-distributive** over the wedge product

$$X \lrcorner (\alpha \wedge \beta) = (X \lrcorner \alpha) \wedge \beta + (-1)^k \alpha \wedge (X \lrcorner \beta). \tag{2.4}$$

2.6 Lie Derivative

The notion of a Lie derivative $\mathcal{L}_X$ extends the usual concept of the derivative of a function along a vector field X to k-forms:

$$\mathcal{L}: \quad \mathfrak{X}(\mathcal{M}) \times \Lambda^k(\mathcal{M}) \to \Lambda^k(\mathcal{M}).$$

Although a dynamic definition of this operator can be given in terms of extrusions along the flow of a vector field [48], in this work we will rely on an algebraic definition given via Cartan's Magic formula:

$$\mathcal{L}_X \alpha = \mathbf{d} X \lrcorner \alpha + X \lrcorner \mathbf{d}\alpha.$$

The Lie derivative can be directly defined as a composition of the metric independent operators contraction $\lrcorner$ and exterior derivative $\mathbf{d}$, without the need for its own separate discrete definition. An important property, which will also hold in the discrete setting, is that the Lie derivative commutes with the exterior derivative $\mathcal{L}_X \mathbf{d} = \mathbf{d} \mathcal{L}_X$, which can be easily shown to hold true because of the fact that $\mathbf{d}^2 = 0$:

$$\begin{aligned} \mathbf{d}\mathcal{L}_X \alpha &= \mathbf{d}(\mathbf{d} X \lrcorner \alpha + X \lrcorner \mathbf{d}\alpha) \\ &= \mathbf{d}(X \lrcorner \mathbf{d}\alpha) \\ &= (\mathbf{d} X \lrcorner + X \lrcorner \mathbf{d}) \mathbf{d}\alpha \\ &= \mathcal{L}_X \mathbf{d}\alpha. \end{aligned}$$

A useful consequence of this is that discrete Lie advection of closed forms will remain closed by construction. As a consequence of Eq. (2.1) and Eq. (2.4), the Lie derivative, like the exterior derivative, also satisfies a Leibniz rule over the wedge product:

$$\mathcal{L}_X(\alpha \wedge \beta) = (\mathcal{L}_X \alpha) \wedge \beta + \alpha \wedge (\mathcal{L}_X \beta).$$

2.7 Laplace-Beltrami

The Laplace-Beltrami operator (sometimes referred to only as the Laplacian) is an elliptic operator that is ubiquitous in physics and mathematics. For example, it arises in the diffusion and wave equations:

$$\Delta: \quad \Lambda^k(\mathcal{M}) \to \Lambda^k(\mathcal{M}).$$

The Laplace-Beltrami operator acts on differential forms to produce forms of the same degree. Explicitly, it is a composition of Hodge stars and exterior derivatives:

$$\Delta = \star\mathbf{d}\star\mathbf{d} + \mathbf{d}\star\mathbf{d}\star$$

and its discrete version can be expressed as compositions of the discrete counterparts of the above operators. The Laplace-Beltrami operator is metric dependent because of the presence of the Hodge star in its definition.

2.8 Flat and Sharp

The flat and sharp are metric dependent operators that convert vectors to 1-forms and vice versa:

$$\flat: \quad \mathfrak{X}(\mathcal{M}) \to \Lambda^1(\mathcal{M}),$$
$$\sharp: \quad \Lambda^1(\mathcal{M}) \to \mathfrak{X}(\mathcal{M}).$$

Known also as the musical operators, they are inverses of each other:

$$\flat\sharp = \sharp\flat = \mathbf{id}$$

In indicial notation the flat raises the indices of a vector to convert it to a form, whereas the sharp lowers the indices of a form to create a vector. The musical operators along with the contraction define an inner product (or metric) on both vectors (X, Y) and forms (α, β) that is given by

$$X \cdot Y = X \lrcorner Y^\flat, \qquad \alpha \cdot \beta = \alpha^\sharp \lrcorner \beta.$$

To illustrate that the operators considered in this chapter form a complete set, we will cast some of the familiar operators of *vector calculus* into the ones that we have defined so far. Operators such as the gradient, curl, and divergence can be readily expressed in 3D as

$$\nabla f = (\mathbf{d}f)^\sharp \qquad \text{(gradient)},$$
$$\nabla \times \mathbf{v} = [\star(\mathbf{d}\mathbf{v}^\flat)]^\sharp \qquad \text{(curl)},$$
$$\nabla \cdot \mathbf{v} = \star\mathbf{d}(\star\mathbf{v}^\flat) \qquad \text{(divergence)}.$$

Additionally, the vector cross product in 3D can be written as

$$\mathbf{u} \times \mathbf{v} = [\star(\mathbf{u}^\flat \wedge \mathbf{v}^\flat)]^\sharp$$

. Keep in mind that associativity of the wedge product does not imply associativity of the cross product.

Chapter 3

FROM POINT COLLOCATION TO CHAIN COLLOCATION

3.1 Introduction

Recent years have seen the development of novel discretizations for a wide variety of systems of partial differential equations. In particular, preserving in the discrete realm the underlying geometric, topological, and algebraic structures at stake in differential equations has proven to be a fruitful guiding principle for discretization [2, 65, 3, 58]. This geometric approach has led to numerical methods, analyzed in, e.g., [36, 3], that inherit a variety of properties from the continuous world. However, geometric discretizations of elasticity, electromagnetism, or fluid mechanics have mostly been demonstrated using spaces of piecewise polynomial differential forms. Many problems where solutions are smoothly varying in space call for a spectral numerical treatment instead, as it produces low-error, exponentially converging approximations by leveraging fast implementations of transforms such as the *fast Fourier transform*. In an effort to provide structure-preserving numerical tools with spectral accuracy on logically rectangular grids over periodic or bounded domains, we present a spectral extension of the discrete exterior calculus described in [8, 21, 7, 31]—and point out that the resulting computational tools extend well-known spectral collocation methods.

Review of Previous Work

Computational methods preserving geometric structures have become increasingly popular over the past few yeears, gaining acceptance among both engineers and mathematicians [4]. Computational electromagnetism [8, 65], mimetic (or natural) discretizations [52, 7], finite-dimensional exterior calculus (including *discrete exterior calculus* (DEC, [20, 21]), and *finite element exterior calculus* (FEEC, [2, 3])) have all proposed discretizations that preserve vector calculus identities in order to improve numerics. In particular, the relevance of exterior calculus (Cartan's calculus of differential forms [18]) and algebraic topology [49] to computations came to light.

Exterior calculus is a concise mathematical formalism to express differential and integral equations on smooth and curved spaces, while revealing the geometric

structures at play and clarifying the nature of the physical quantities involved. At the heart of exterior calculus is the notion of differential forms, denoting antisymmetric tensors of arbitrary order. As integration of differential forms is an abstraction of the measurement process, this calculus of forms provides an intrinsic, coordinate-free approach particularly relevant to the neat description of a multitude of physical models making heavy use of line, surface and volume integrals [1, 15, 25, 26, 60]. Moreover, physical measurements, such as fluxes, represent local integrations over a small surface of the measuring instrument. Pointwise evaluation of such quantities does not have physical meaning; instead, one should manipulate these quantities only as geometrically-meaningful entities integrated over appropriate submanifolds.

Algebraic topology, specifically the notion of chains and cochains [73, 49] has been used to provide a natural discretization of differential forms and to emulate exterior calculus on finite grids: a set of values on vertices, edges, faces, and cells are proper discrete versions of respectively pointwise functions, line integrals, surface integrals, and volume integrals [8]. This point of view is entirely compatible with the treatment of volume integrals in finite volume methods, or scalar functions in finite element methods; however, it also involves the "edge elements" and "facet elements" (as first introduced in computational electromagnetism) as special H_{div} and H_{curl} basis elements [51]. Equipped with such discrete forms of arbitrary degree, Stokes' theorem connecting differentiation and integration is automatically enforced if one thinks of differentiation as the dual of the boundary operator—a particularly simple operator on meshes. With these basic building blocks, important structures and invariants of the continuous setting directly carry over to the discrete world, culminating in a discrete Hodge theory [21, 3]. As a consequence, such a discrete exterior calculus has already proven useful in many areas such as electromagnetism [8, 65], fluid simulation [58], (re)meshing of surfaces [68, 47], and graph theory [31] to mention a few. So far, only piecewise polynomial basis functions [2, 72] have been employed in these applications, thus limiting their computational efficiency in terms of convergence rates.

Spectral Methods

Spectral methods are a class of spatial discretizations of differential equations widely recognized as crucial in fluid mechanics, electromagnetics, and other applications where solutions are expected to be smooth. Central to the efficiency of this large family of numerical methods is the fact that the approximation of a periodic C^∞ function by its trigonometric interpolation over evenly spaced points converges faster

than any polynomial order of the step size. This is sometimes referred to as "spectral accuracy" or "super-convergence." In practice, spectral accuracy can be achieved for bounded domains through continuation methods [14] or using Gauss-Lobatto quadrature on Legendre or Chebyshev grids [17]. A larger number of spectral methods have been designed, varying in the mesh they consider (primal grids only, or staggered grids [39]), and the locations at which they enforce partial differential equations (PDE). Be it for Galerkin, Petrov-Galerkin, or collocation-based spectral schemes, it has however been noticed that besides constructing spectrally accurate approximations of the relevant fields and their derivatives involved in a PDE, numerically preserving conservation properties helps in obtaining stable and/or physically adequate results [17]. Yet, numerical schemes are often proven conservative a posteriori, as a formal approach to guarantee conservation properties by design remains elusive.

Motivations and Contributions

Despite an increasingly large body of work on numerical approaches based on exterior calculus, developing a spectrally accurate calculus of discrete forms has received very little attention—with a few recent exceptions [59, 10, 28, 55] that we will build upon. We present a discrete exterior calculus of differential forms on periodic or bounded domains, including wedge product, Hodge star, and exterior derivative, all of which converge spectrally under grid refinement while utilizing fast Fourier methods to remain computationally efficient. In order to construct a spectral representation of the operators on differential forms, we expand the conventional tools of spectral methods to give spectrally accurate approximations of fields for which integral values over specified domains are known—a process referred to as histopolation [59, 72]. In this chapter we construct a histopolation using trigonometric polynomials on periodic domains, and consider the extension to bounded domains using a Chebyshev grid, thereby allowing the use of the fast Fourier transform for efficient calculation.

Our work lays out a set of spectral, structure-preserving computational tools with the following distinguishing features:

- We leverage existing work in algebraic topology to discretize space through *chains* (linear combination of mesh elements) and differential forms through cochains (discrete forms). The resulting discrete de Rham complex, that by construction satisfies Stokes' theorem, offers a consistent, "structure-

preserving" manipulation of integrals and differentials which respect important conservation laws. This approach, used mainly so far in non-spectral computations [2], was identified in [10, 28] as a significant departure in the construction of conservative schemes from traditional spectral methods, since divergences, gradients, and curls are no longer computed through derivation but directly evaluated via a metric-independent exterior derivative without having recourse to approximations—thus exactly enforcing the divergence theorem, Green's theorem, etc.

- We extend the basic DEC operators to ensure spectral accuracy. Using a framework with some similarities to that of Bochev [7] and DEC [21], we construct operations on discrete differential forms by interpolating their finite-dimensional (point-sampled or integrated) representation, performing exterior calculus operations in continuous space, and point-sampling or integrating the results to project back to their finite-dimensional representation. We differ from previous spectral DEC approaches [55, 10, 28] in that we provide a natural Hodge star leveraging mesh duality, introduce a wedge product, satisfy consistency conditions between the various operators and basis functions, and provide their explicit expressions to facilitate implementation.

- Our spectrally accurate operators result in a *spectral chain collocation method* to solving partial differential equations: a differential equation is not enforced at a series of points, but at a series of mesh elements (points, edges, faces, etc), extending point collocation methods to respect the natural geometric structure of the continuous equation. We also provide a discussion on the implementation of this spectral computational framework, and show results on Poisson problems with various boundary conditions.

- Finally, our approach is versatile: since it builds on the notion of integrated values (cochains to be precise), one can use the same computational tools on curved grids with only minor changes.

For simplicity of presentation, we will first treat regular and Chebyshev one-dimensional (1D) grids before extending our approach to grids of arbitrary dimension via tensor products. Other Chebyshev-like grids are easy to derive as well, since integrals of differential forms (the building blocks of our approach) are unchanged by pushforward or pullback. For the reader's convenience, Table 3.1 lists the main notations we will use throughout this chapter.

Symbol	Meaning
d	Dimension of the domain
Ω	Spatial domain in $\mathbb{R}^d$
K	Computational grid of Ω
$\widetilde{K}$	Dual grid to K
σ^k	Primal k-dim element of grid K
$\widetilde{\sigma}^k$	Dual k-dim element of grid $\widetilde{K}$
$*$	Duality operator ($*\sigma^k = \widetilde{\sigma}^{n-k}$)
$\boldsymbol{\partial}$	Boundary operator on mesh elements
Λ^k	Space of differential k-forms ω^k over Ω
$\overline{\Lambda}^k$	Space of discrete k-forms $\bar{\omega}^k$ on K
$\mathbf{D}$	Discrete exterior derivative operator $\mathbf{d}$
$\mathbf{W}$	Discrete wedge operator $\wedge$
$\mathbf{H}$	Discrete Hodge star operator $\star$
$\mathcal{F}$	Discrete Fourier transform
$\hat{\cdot}$	Quantities expressed in Fourier space

Table 3.1: Meaning of the basic notations.

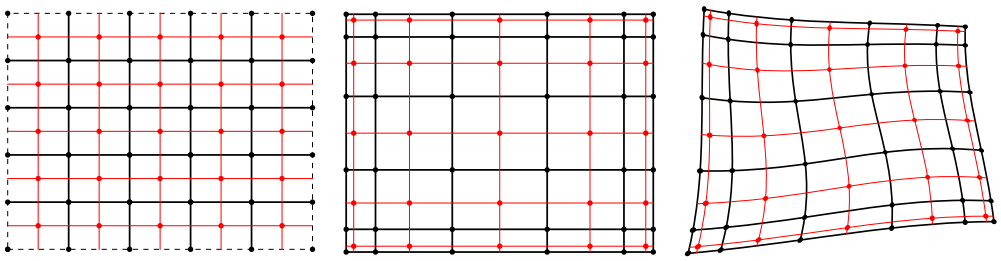
3.2 Discrete Differential Forms and Operators

Differential forms were pioneered by Cartan [18] in an effort to provide a unified approach to defining differentiation and integration over curves, surfaces, volumes, and higher-dimensional manifolds. Since then, forms have been shown to be crucial in elucidating the structures and invariants in physics [26]. Since most of our measurements of the world are of integral nature, forms have been also at the core of a number of numerical approaches targeting the definition of numerical counterparts of differential operators that are compatible with the geometric and topological structures underlying well-posed partial differential equations.

Most computational methods based on exterior calculus use the notion of simplicial k-cochain as a fundamental object that assigns a number to each simplex of dimension k of the simplicial complex. Cochains are, in a sense, a natural discretization of differential forms of degree k, which are objects to be integrated over k-dimensional domains to return a number; in fact, de Rham's theorem states that the (discretization) map from the de Rham complex to the simplicial cochain complex induces an isomorphism on cohomology. Conversely, one can go from cochains back to differential forms using Whitney forms [73, 8] or higher-order piecewise polynomial finite element spaces [2, 72]. These mimetic properties of finite-dimensional

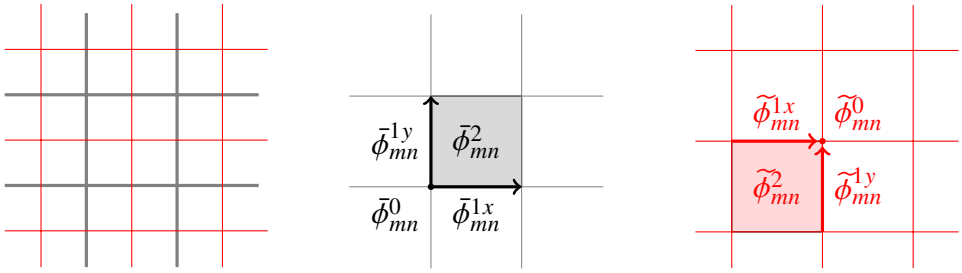
approximations to differential forms have led to the term *discrete differential forms* to describe cochains [21].

In this section, we review the principles and approaches used in discrete versions of exterior calculus in preparation for extending them to spectral computations. By construction, the calculus of discrete differential forms automatically preserves a number of important geometric structures, including integration by parts (with a proper treatment of boundaries), Stokes' theorem, the de Rham complex, Poincaré duality, Poincaré's lemma, and Hodge theory. Therefore, it provides a suitable foundation for a coordinate-free discretization of geometric field theories and associated computations, with applications including electromagnetism [65], incompressible Euler and complex fluids [58], and meshing algorithms [68, 19]. The particular "flavor" of discrete differential forms and operators we will be leveraging is known as *discrete exterior calculus*, or DEC for short; see [35, 43]. (For related work in this direction, see also [33] and [2]; DEC most significantly differs from these other discrete calculus approaches in the way the Hodge duality is defined through the use of a dual mesh.) We refer the reader to [21] for a brief introduction to DEC.



(a) Periodic **(b)** Chebyshev **(c)** Logically rectangular

Figure 3.1: Logically rectangular grids such as (a) regular grids over periodic domains, (b) Chebyshev grids over bounded domains, or (c) an arbitrarily mapped rectangular grid. Primal elements are displayed in black, while their duals are in red.



(a) Primal/dual grid **(b)** Primal grid **(c)** Dual grid

Figure 3.2: Illustration of a regular 2D grid (in black) along with its dual (in red), along with our mesh orientation and index convention.

Mesh and Dual Mesh as Spatial Discretization

In our approach, the continuous domain (usually a smooth d-dimensional manifold Ω) is approximated by a mesh—more precisely, by a cell complex that is manifold, admits a metric, and is orientable. We will generally denote the complex by K, and a cell in the complex by σ. Superscripts will be used at times to denote the dimension of a cell; i.e., σ^0 represents a vertex of K, σ^1 represents an edge, etc. Moreover, we restrict our study to the case of logically rectangular grids over periodic and bounded domains that include regular grids and Chebyshev grids in $\mathbb{R}^n$ (see Figure 3.1); that is, each k-cell for $k \geq 1$ has $2k$ non-degenerate faces, and the mapping function between the rectangular grid of a rectangular domain and the actual spatial grid is assumed to be bijective and C^∞. Consequently, the mesh topology is easily encoded using indices as indicated in Figure 3.2; we will also assume prescribed orientations as the figure indicates as well.

From a mesh K, one can construct a *dual mesh*—where duality is to be understood in the sense of graph duality. The dual mesh is defined via the location of its dual vertices, whereas its connectivity is directly induced by the primal connectivity. Once a primal mesh and its dual are defined, one can define the *duality operator* $\star$ that maps each k-cell σ^k of the primal mesh K to a dual $(n–k)$-cell $\star\sigma^k$ in its dual cell complex $\star K$. Note that a refined definition of the dual mesh for bounded domains (where dual cells at the boundary are restricted ("clipped") to K to allow proper enforcement of boundary conditions) is assumed as well (see, for instance, [22, 65]).

The canonical placement of primal and dual nodes for Chebyshev grids will be given in Section 3.3; but arbitrarily distorted grids will be straightforward to handle (see Section 3.6), owing to the fact that the exterior derivative and the pullback operator commute.

Discretization of Differential Forms

The fundamental objects of DEC are discrete differential forms. A discrete k-form $\bar{\omega}^k$, the discrete counterpart of a continuous k-form ω^k, assigns a real number $\bar{\omega}^k_i$ to each oriented k-dimensional cell σ^k_i in the mesh K—and zero for all other mesh elements. (The superscripts k are not actually required by the notation, but they are often useful as reminders of what dimension of form or cell we are dealing with.) The value of the form on σ^k_i is sometimes denoted by $\left\langle \omega^k, \sigma^k_i \right\rangle$, and represents the value of the continuous form ω^k integrated over the mesh element σ^k_i, i.e.,

$$\bar{\omega}^k_i \equiv \left\langle \omega^k, \sigma^k \right\rangle = \int_{\sigma^k} \omega^k.$$

For implementation purposes, a discrete k-form can therefore be thought of as an array of values, one per k-dimensional cell in the mesh K. For example, discrete 0-forms sample values of an associated continuous 0-form at vertices, discrete 1-forms sample values of a continuous 1-form on all edges, etc. Note that we can even integrate these discrete forms over discrete paths by linearity: simply add the form's values on each cell in the path, taking care to flip the sign if the path is oriented opposite the cell. Formally, these "paths" of k-dimensional elements are called *chains*, and discrete differential forms are *cochains*, where $\langle \cdot, \cdot \rangle$ is the pairing between cochains and chains [49].

Discrete differential forms can be defined either on the mesh K or on its dual $*K$. We will refer to these as *primal forms* and *dual forms* respectively, and denote the space of primal discrete forms by $\bar{\Lambda}$ and the space of dual discrete forms by $\widetilde{\Lambda}$. Note that there is a natural correspondence between primal k-forms $\bar{\omega}^k$ and dual $(n-k)$-forms $\widetilde{\omega}^{n-k}$, corresponding to the mesh duality discussed in Section 3.2, for which each primal k-cell σ^k has an associated dual $(n-k)$-cell $*\sigma^k$ (Figure 3.2). This important property will be used below to define the discrete Hodge star operator.

Reduction and Reconstruction Maps

The reduction and reconstruction maps provide a way to go back and forth between continuous forms and their discrete realizations.

Reduction. The reduction map (also called the de Rham map) is a linear operator $\mathcal{P}$ that projects a continuous form to its discrete realization on the grid through integration over mesh elements:

$$\begin{aligned} \mathcal{P}: \quad & \Lambda^k \to \bar{\Lambda}^k \\ & \omega^k \to \bar{\omega}^k \quad \text{with} \quad \bar{\omega}^k_i = (\mathcal{P}\omega^k)_i \equiv \int_{\sigma^k_i} \omega^k. \end{aligned}$$

We will denote by $\widetilde{\mathcal{P}}$ the analogous operator mapping continuous forms to their *dual* discrete counterparts in a similar fashion:

$$\begin{aligned} \widetilde{\mathcal{P}}: \quad & \Lambda^k \to \widetilde{\Lambda}^k \\ & \omega^k \to \widetilde{\omega}^k, \quad \text{with} \quad \widetilde{\omega}^k_i = (\widetilde{\mathcal{P}}\omega^k)_i \equiv \int_{\widetilde{\sigma}^k_i} \omega^k. \end{aligned}$$

Note that this definition of reduction extends the notion of *point sampling*: while the reduction of a 0-form is found by simply point-sampling its value at each vertex of the grid, the reduction of a general k-form is its evaluation (i.e., integral) on all

the k-dimensional elements (vertices for $k=0$, edges for $k=1$, faces for $k=2$, etc) of the grid.

Reconstruction. Conversely, the reconstruction map $(\mathcal{R})$ is a map which reconstructs a continuous k-form from its discrete realization by interpolation for $k=0$, and by histopolation otherwise:

$$\begin{aligned} \mathcal{R}: \quad & \bar{\Lambda}^k \to \Lambda^k \\ & \bar{\omega}^k \to \omega^k. \end{aligned}$$

We will denote by $\widetilde{\mathcal{R}}$ the analogous operator mapping *dual* discrete forms to continuous forms in a similar fashion. Note that we will sometimes omit the dual sign $\widetilde{\cdot}$ for clarity, as which reconstruction operator is meant is unambiguously implied by the (primal or dual) nature of the discrete form it is applied on.

If one has a set of basis functions $\{\phi_0^k(x), \phi_1^k(x), \dots\}$ for k-forms that satisfy the property

$$\forall i, j, \quad \int_{\sigma_i^k} \phi_j^k = \delta_{ij} \tag{3.1}$$

then $\mathcal{R}$ can trivially be defined as

$$\omega^k = \mathcal{R}\bar{\omega}^k \equiv \sum_i \bar{\omega}_i^k \phi_i^k.$$

One can readily verify that this reduction map is a left inverse of the reconstruction map:

$$\mathcal{P}\mathcal{R} = \mathrm{Id}.$$

However, the converse is not true; the reconstruction map is only approximately the right inverse of the reduction map, with equality in the limit when the mesh element size h approaches 0:

$$\|\omega - \mathcal{R}\mathcal{P}\omega\| \xrightarrow[h\to 0]{} 0,$$

with a rate of convergence determined by the chosen norm on forms and the degree of the basis functions. While Whitney first introduced a one-sided inverse of the de Rham map using what amounts to piecewise linear basis functions [73], we will instead use global basis functions satisfying Eq. (3.1) to provide spectrally accurate reconstructions.

3.3 Basic Spectral Tools

Before delving into the design of spectrally accurate discrete operators, we must define a series of basic tools and conventions which will be particularly useful for our task. We start by defining proper periodic basis functions on regular grids, before describing the case of Chebyshev grids.

1D Periodic Interpolator and Histopolator Functions

To build our spectral wedge and Hodge star operators to work on discrete forms of arbitrary degree, we will need not only spectral interpolating basis functions, but also spectral *histopolating* basis functions, i.e., basis functions which integrate to one over assigned intervals [10]. To this end, we consider a one-dimensional periodic domain of width 2π with N *regularly-spaced nodes*, and define over this canonical domain two scalar functions α_N and β_N—that we will respectively call *interpolator* and *histopolator*—as follows:

$$\boxed{\alpha_N(x) = \frac{1}{N}\begin{cases}\cot\frac{x}{2}\,\sin\frac{Nx}{2} & \text{if N even,}\\ \csc\frac{x}{2}\,\sin\frac{Nx}{2} & \text{if N odd,}\end{cases}} \tag{3.2}$$

and

$$\boxed{\beta_N(x) = \begin{cases}\frac{1}{2\pi} - \frac{1}{4}\cos\frac{Nx}{2} + \frac{1}{N}\sum_{n=1}^{N/2}\frac{n\cos nx}{\sin\frac{n\pi}{N}} & \text{if N even,}\\ \frac{1}{2\pi} + \frac{1}{N}\sum_{n=1}^{(N-1)/2}\frac{n\cos nx}{\sin\frac{n\pi}{N}} & \text{if N odd.}\end{cases}} \tag{3.3}$$

For convenience, let us denote any translation of the above functions using the notation below:

$$\alpha_{N,n}(x) = \alpha_N(x - hn), \quad \text{and} \quad \beta_{N,n}(x) = \beta_N(x - hn),$$

where $h = 2\pi/N$ is the mesh's edge element size and the nodes are enumerated by $x_n = nh$. For any given number N of nodes, these two functions satisfy the following important properties mentioned in Eq. (3.1) (where δ_{mn} refers to the Kronecker delta):

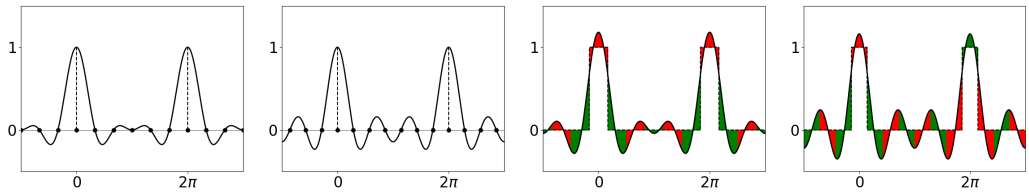
$$\alpha_{N,n}(x_m) = \delta_{mn}, \quad \text{and} \quad \int_{x_m-\frac{h}{2}}^{x_m+\frac{h}{2}} \beta_{N,n}(x)\,dx = \delta_{mn}.$$

In other words, α_N provides a smooth interpolation of a discrete function with 1 at node x_0 and 0 at every other node, while β_N integrates to 1 over the interval $[x_0 - \frac{h}{2}, x_0 + \frac{h}{2}]$, and to 0 over all other intervals (see Figure 3.3). Notice that these

functions are the only Fourier series with N sinusoidal components satisfying the point-wise (resp., interval-wise) constraints. Note also that α_N and β_N are related to each other:

$$\int_{x-\frac{h}{2}}^{x+\frac{h}{2}} \beta_N'(\xi)d\xi = \alpha_N(x), \quad \text{or equivalently,} \quad \beta_N(x+\frac{h}{2}) - \beta_N(x-\frac{h}{2}) = \alpha_N'(x).$$

Next, we show that these two functions provide building blocks to construct basis functions for arbitrary forms on regular, periodic grids—from which we will derive basis functions for bounded domains via pushforward.



(a) $\alpha_6(x)$ **(b)** $\alpha_7(x)$ **(c)** $h\beta_6(x)$ **(d)** $h\beta_7(x)$

Figure 3.3: Examples of periodic interpolators for $N = 6$ (a) and 7 (b), and corresponding periodic histopolators (c) and (d), scaled by $h = 2\pi/N$ for clarity. While the interpolator α_N satisfies $\alpha_N(nh \mod 2\pi) = \delta_{0n}\ \forall n$, the histopolator β_N integrates to 1 over the dual cell straddling $x = 0$, and to 0 over other dual cells in the range $[0, 2\pi]$. Note that the alternating red and green colors are used to mark out dual cells, and to illustrate that the integral of β_N over each of these dual cells sums to zero or one.

Spectral Basis Functions on Regular Grids

Basis functions for periodic grids in arbitrary dimensions can be easily built through tensor products of (translated) interpolators α_N and histopolators β_N, where the number of β_N used in the tensor product is equal to the degree of the form. To make this point clear, we will now introduce our notation for 1-, 2-, and 3-dimensional basis functions, with the extension to higher dimensions being straightforward. We will denote the basis functions as $\phi^{p[\text{comp}]}_{[\text{ind}]}$ with superscripts used to indicate the degree p, followed by the component of the form when appropriate (e.g., x, xy, etc), and subscripts used for grid indices; for instance, $\phi^{1y}_{mnk}(.)$ is the function of $\mathbb{R}^3$ representing the $\mathbf{d}y$-component of the 1-form basis, located on the edge parallel to the y axis indexed by (m, n, k). See Figure 3.2 for an illustration of the 2D notations.

Primal and Dual Nodes. On a 1D regular grid the N primal nodes are equally spaced, and the dual nodes are placed in the middle between two adjacent primal nodes. Consequently, the primal nodes are: $x_n = 2\pi n/N$, and the dual nodes are at $\widetilde{x}_n = 2\pi(n + \frac{1}{2})/N$.

Basis Functions for Forms in 1D. Because of their interpolation (resp., histopolation) property, translated versions of the functions α_N and β_N can directly be used as basis functions for 0- and 1-forms respectively as follows:

$$\phi^0_{N,n}(x) = \alpha_{N,n}(x), \qquad \text{and} \qquad \phi^1_{N,n}(x) = \beta_{N,n+\frac{1}{2}}(x)\,\mathbf{d}x,$$

where $n \in \{0, 1, \ldots, N-1\}$. Indeed, a discrete 0-form $\bar{\omega}^0$ (resp., a discrete 1-form $\bar{\omega}^1$) can be reconstructed as a smooth form through $\omega^0 = \sum_i \bar{\omega}^0_i \phi^0_i$ (resp., $\omega^1 = \sum_i \bar{\omega}^1_i \phi^1_i$); the reconstructed form then satisfies $\mathcal{P}\omega^0 = \bar{\omega}^0$ (resp., $\mathcal{P}\omega^1 = \bar{\omega}^1$). Similarly, dual basis functions are easily designed as well through

$$\tilde{\phi}^0_{N,n}(x) = \alpha_{N,n+\frac{1}{2}}(x) \qquad \text{and} \qquad \tilde{\phi}^1_{N,n}(x) = \beta_{N,n}(x)\,\mathbf{d}x.$$

Basis Functions for Forms in 2D. In two dimensions, tensor products of the one-dimensional bases provide basis functions for 0-, 1-, and 2-forms on a regular $M \times N$ grid. These functions are expressed as follows:

$$\begin{aligned}
\phi^0_{MN,mn}(x,y) &= \phi^0_{M,m}(x) \wedge \phi^0_{N,n}(y),\\
\phi^{1x}_{MN,mn}(x,y) &= \phi^1_{M,m}(x) \wedge \phi^0_{N,n}(y),\\
\phi^{1y}_{MN,mn}(x,y) &= \phi^0_{M,m}(x) \wedge \phi^1_{N,n}(y),\\
\phi^2_{MN,mn}(x,y) &= \phi^1_{M,m}(x) \wedge \phi^1_{N,n}(y).
\end{aligned}$$

One can easily check that these functions are 1 on their associated degree of freedom and 0 on all others (vertices for 0-forms, edges for 1-forms, and faces for 2-forms), thus offering a proper set of bases for smooth reconstructions of discrete forms. Formulas for the dual basis functions are strictly analogous, where $\widetilde{\phi}$ is used in lieu of ϕ.

Basis Functions for Forms in 3D. The same construction can be used in three or higher dimensions. For completeness, we describe the three-dimensional basis functions for primal forms:

$$\begin{aligned}
\phi^0_{MNK,mnk}(x,y,z) &= \phi^0_{M,m}(x) \wedge \phi^0_{N,n}(y) \wedge \phi^0_{K,k}(z),\\
\phi^{1x}_{MNK,mnk}(x,y,z) &= \phi^1_{M,m}(x) \wedge \phi^0_{N,n}(y) \wedge \phi^0_{K,k}(z),\\
\phi^{1y}_{MNK,mnk}(x,y,z) &= \phi^0_{M,m}(x) \wedge \phi^1_{N,n}(y) \wedge \phi^0_{K,k}(z),\\
\phi^{1z}_{MNK,mnk}(x,y,z) &= \phi^0_{M,m}(x) \wedge \phi^0_{N,n}(y) \wedge \phi^1_{K,k}(z),\\
\phi^{2xy}_{MNK,mnk}(x,y,z) &= \phi^1_{M,m}(x) \wedge \phi^1_{N,n}(y) \wedge \phi^0_{K,k}(z),\\
\phi^{2xz}_{MNK,mnk}(x,y,z) &= \phi^1_{M,m}(x) \wedge \phi^0_{N,n}(y) \wedge \phi^1_{K,k}(z),\\
\phi^{2yz}_{MNK,mnk}(x,y,z) &= \phi^0_{M,m}(x) \wedge \phi^1_{N,n}(y) \wedge \phi^1_{K,k}(z),\\
\phi^3_{MNK,mnk}(x,y,z) &= \phi^1_{M,m}(x) \wedge \phi^1_{N,n}(y) \wedge \phi^1_{K,k}(z).
\end{aligned}$$

Note that since the wedge products above are the continuous (as opposed to discrete) ones, associativity holds, and no ambiguity is introduced by the omission of parentheses. Here again, it is easy to check that these functions provide smooth reconstructions from values of vertices, edges, faces, or volumes on a regular grid indexed by m, n, and k, in a periodic domain. Formulas for the dual basis functions are strictly analogous, where $\widetilde{\phi}$ is used in lieu of ϕ.

Chebyshev Grids over Bounded Domains

For bounded domains, a popular choice of spatial discretization in spectral methods is the use of Chebyshev computational grids [17]. It is well known that Chebyshev polynomials can be derived as the pullback of Fourier basis functions from a circle onto its diameter; see Figure 3.4. We can, in fact, perform the same pullback of our regular-grid basis functions of forms to obtain new spectral bases of forms applicable for Chebyshev grids over non-periodic domains.

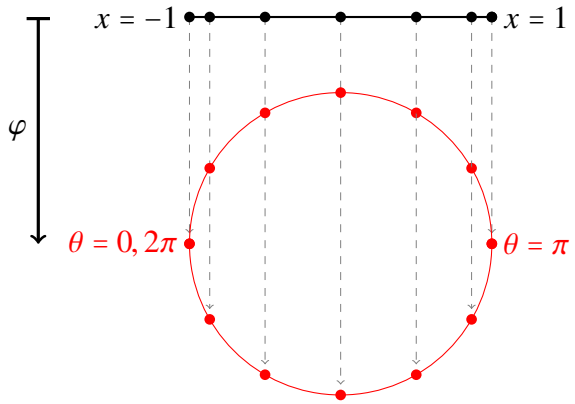


Figure 3.4: The map φ mapping the canonical interval $[-1, 1]$ to the bottom unit hemicircle $(0 \le \theta \le \pi)$.

Denote by φ the map from the interval between -1 and 1 on the real line (with Cartesian coordinate x) to the unit semicircle (with polar angle θ, see Figure 3.4):

$$\begin{aligned} \varphi: \quad [-1,1] \subset \mathbb{R} &\to [0,\pi] \subset \mathcal{S}^1 \\ x &\mapsto \theta = \operatorname{acos}(-x). \end{aligned}$$

Primal and Dual Nodes. Let the 1D grid consist of N points (including the boundaries) in the interval $[-1, 1]$. The corresponding unit circle will then have $2N-2$ points (see Figure 3.4). The primal nodes $\{x_n\}$ $(n = 0..N-1)$ and dual nodes $\{\widetilde{x}_n\}$ $(n = 0..N-2)$ of the Chebyshev grid are thus

$$x_n = -\cos\frac{n\pi}{N-1} \quad \text{and} \quad \widetilde{x}_n = -\cos\frac{(n+\frac{1}{2})\pi}{N-1}.$$

Basis functions. To design our basis functions, we first define functions on the semicircle ($\theta \in [0, \pi]$) by mirroring/antimirroring the regular basis functions ϕ on the whole circle (see [69] for the usual case of primal 0-forms), in order to satisfy the interpolation/histopolation properties (Eq. (3.1)) on the semicircle. The resulting functions κ are expressed as:

$$\kappa^0_{N,n} = \begin{cases} \phi^0_{2N-2,n}, & n = 0 \text{ or } n = N-1 \text{ (endpoints)} \\ \phi^0_{2N-2,n} + \phi^0_{2N-2,2N-2-n}, & n \in \{1, \dots, N-2\} \text{ (midpoints)} \end{cases}$$

for primal 0-forms,

$$\kappa^1_{N,n} = \phi^1_{2N-2,n} - \phi^1_{2N-2,2N-3-n}, \quad n \in \{0, \dots, N-2\}$$

for primal 1-forms,

$$\widetilde{\kappa}^0_{N,n} = \widetilde{\phi}^0_{2N-2,n} + \widetilde{\phi}^0_{2N-2,2N-3-n} \quad n \in \{0, \dots, N-2\}$$

for dual 0-forms, and

$$\widetilde{\kappa}^1_{N,n}(\theta) = \begin{cases} \delta_N(\theta) & n = 0 \\ \left[\widetilde{\phi}^0_{2N-2,n} - \widetilde{\phi}^0_{2N-2,2N-2-n} - \rho_{N,n}\right](\theta) & n \in \{1, \dots, N-2\} \\ \delta_N(\pi - \theta) & n = N-1 \end{cases}$$

for dual 1-forms, where

$$\delta_N(\theta) = \left((N-1)^2 \alpha_{2N-2,0}(\theta) + \frac{1}{2}\cos\left((N-1)\theta\right)\right) \sin\theta \, \mathbf{d}\theta,$$
$$\text{and} \quad \rho_{N,n}(\theta) = 2\left(\gamma_{2N-2,n}\delta_N(\theta) + \gamma_{2N-2,N-n-1}\delta_N(\pi-\theta)\right) \mathbf{d}\theta,$$

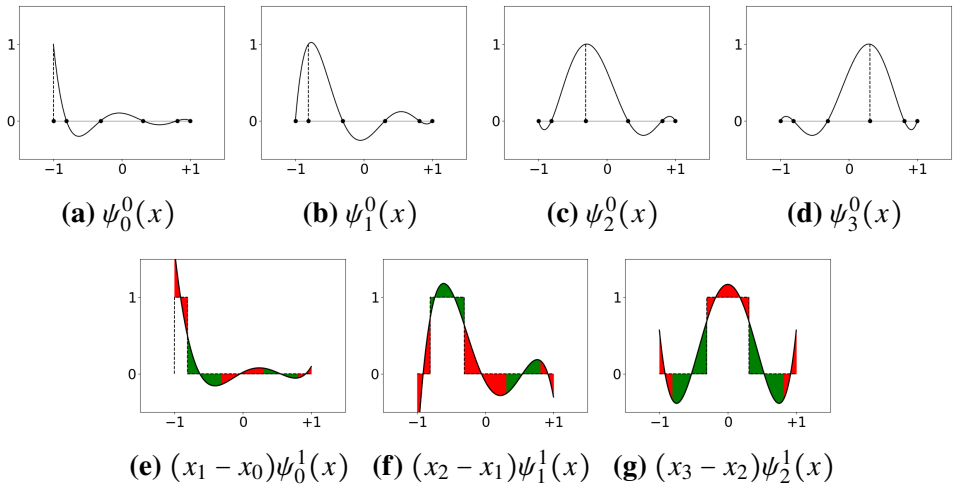
$$\text{with: } \gamma_{N,n} = \int_0^{\frac{\pi}{N}} \beta_{N,n}(\theta)\, d\theta$$
$$= \begin{cases} \frac{1-(-1)^k}{2N} + \frac{1}{N}\sum_{n=1}^{N/2} \frac{\sin(2kn\pi/N) - \sin((2k-1)n\pi/N)}{\sin\frac{n\pi}{N}} & \text{if N even,} \\ \frac{1}{2N} + \frac{1}{N}\sum_{n=1}^{(N-1)/2} \frac{\sin(2kn\pi/N) - \sin((2k-1)n\pi/N)}{\sin\frac{n\pi}{N}} & \text{if N odd.} \end{cases}$$

The function δ is used to deal with the special case of the two boundary dual (half)edges, and the function ρ adds contributions to intermediate basis functions

so that they integrate to zero at both boundary dual edges. Finally, we pull back the functions κ by φ to obtain the form basis functions ψ on the Chebyshev 1D grid ($\psi = \varphi^* \kappa$):

$$\begin{aligned}
\psi^0_{N,n}(x) &= \kappa^0_{N,n}(\operatorname{acos}(-x)), \\
\psi^1_{N,n}(x) &= \kappa^1_{N,n}(\operatorname{acos}(-x)) \frac{\mathbf{d}x}{\sqrt{1-x^2}}, \\
\tilde{\psi}^0_{N,n}(x) &= \tilde{\kappa}^0_{N,n}(\operatorname{acos}(-x)), \\
\tilde{\psi}^1_{N,n}(x) &= \tilde{\kappa}^1_{N,n}(\operatorname{acos}(-x)) \frac{\mathbf{d}x}{\sqrt{1-x^2}}.
\end{aligned}$$

One can easily check that the functions above satisfy the property in Eq. (3.1) required for basis functions, as the functions κ were designed to satisfy these properties, and the pullback φ^* commutes with integration.



(a) $\psi^0_0(x)$ **(b)** $\psi^0_1(x)$ **(c)** $\psi^0_2(x)$ **(d)** $\psi^0_3(x)$

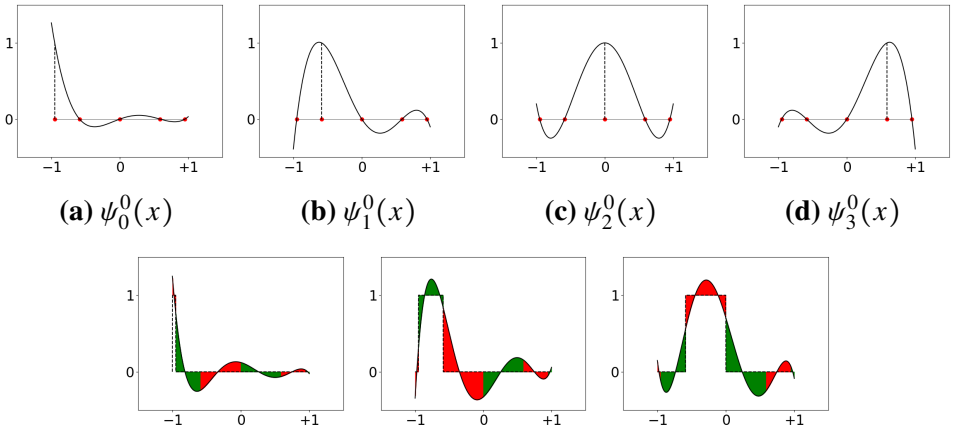
(e) $(x_1 - x_0)\psi^1_0(x)$ **(f)** $(x_2 - x_1)\psi^1_1(x)$ **(g)** $(x_3 - x_2)\psi^1_2(x)$

Figure 3.5: Chebyshev primal basis functions for a grid with $N = 7$. We normalize the one-form basis functions by $x_n - x_{n-1}$ to have approximately the same scale in our visualizations.

Note that the basis functions for primal zero-forms turn out (unsurprisingly) to be the Lagrange polynomials of order N, i.e.,

$$\psi^0_{N,n}(x) = \prod_{\substack{m=0 \\ m \neq n}}^{N-1} \frac{x - x_m}{x_n - x_m},$$

where $\{x_n\}_{n=0,\ldots,N-1}$ represent the coordinates of the primal points. All other basis functions of forms are *also* polynomials for any choice of N. Examples for the primal and dual basis functions for 0- and 1-forms are provided in Figures 3.5 and 3.6 for $N = 7$.



(a) $\psi_0^0(x)$ **(b)** $\psi_1^0(x)$ **(c)** $\psi_2^0(x)$ **(d)** $\psi_3^0(x)$

(e) $(x_1 - x_0)\psi_0^1(x)$ **(f)** $(x_2 - x_1)\psi_1^1(x)$ **(g)** $(x_3 - x_2)\psi_2^1(x)$

Figure 3.6: Chebyshev dual basis functions for a grid with $N = 7$

Finally, basis functions in higher dimensions for arbitrary forms are derived using tensor products of these two primal and two dual $1D$ basis functions, just as we explained in Section 3.3.

3.4 Spectrally Accurate Discrete Operators

Equipped with these basis functions on regular and Chebyshev grids, we can now derive the discrete, spectrally accurate version of the exterior derivative $\mathbf{d}$, the wedge product $\wedge$, and the Hodge star $\star$. Emphasis is placed on the mathematical definitions; fast evaluations of these operators will be discussed subsequently in Section 3.5.

Discrete Exterior Derivative D

The most common discrete realization of the exterior derivative based on algebraic topology [49] using chains and cochains is a linear operator $\mathbf{D}$:

$$\mathbf{D}: \quad \bar{\Lambda}^k \to \bar{\Lambda}^{k+1},$$

which is the (signed) incidence matrix between $(k+1)$ elements and k elements of the grid, with a sign determined by the relative orientation of the elements. In other words,

$$\boxed{\mathbf{D} = \boldsymbol{\partial}^t,}$$

where $\boldsymbol{\partial}$ refers to the *boundary operator* acting on chains (see [49, 21]). Note that the operator $\mathbf{D}$ is thus implemented via a sparse matrix whose non-zero elements have only $+1$ and -1 values to indicate incidence between mesh cells. It also satisfies $\mathbf{D}^2 = 0$ like its continuous equivalent (since the boundary of a boundary is always

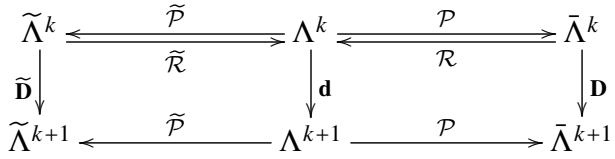
the empty set), and that a *discrete Stokes' theorem* is also enforced on chains since the very definition of $\mathbf{D}$ is tantamount to enforcing

$$\int_{\sigma} \mathbf{d}\omega = \int_{\partial\sigma} \omega$$

for every mesh element σ. This discrete realization of the exterior derivative is exact, in the sense that the operator $\mathbf{D}$ commutes with the reduction operator:

$$\mathbf{d}\mathcal{P} = \mathcal{P}\mathbf{D}. \tag{3.4}$$

This implies that the following diagram fully commutes:



Discrete Dual Forms | *Differential Forms* | *Discrete Primal Forms*

Therefore, as mentioned in Section 2.2, the classic discrete exterior derivative used in mimetic methods, DEC, or finite-dimensional exterior calculus needs no special treatment to obtain spectral accuracy. However, its use for spectral computations represents a clear departure from the conventional spectral methods [10], since all operators of classical vector calculus (divergence, gradient, and curl) can be achieved exactly via $\mathbf{D}$ [28]. In our framework, we also use the exterior derivative on dual forms:

$$\widetilde{\mathbf{D}}: \quad \widetilde{\Lambda}^k \to \widetilde{\Lambda}^{k+1}.$$

Its implementation and properties are no different from its primal version, since the adjacency of the dual mesh is directly derived from the adjacency on the primal:

$$\boxed{\widetilde{\mathbf{D}} = \widetilde{\boldsymbol{\partial}}^t}.$$

We now turn to the definition of a discrete wedge product and discrete Hodge star operator.

Discrete Wedge Product W

Various discrete definitions of the wedge product have been proposed, sometimes mixing primal and dual elements [35]. We instead follow Bochev and Hyman's treatment [7]: by utilizing the de Rham and reconstruction maps, one can define a

discrete wedge product $\mathbf{W}(\bar{\alpha}, \bar{\beta})$ in a general way that applies to any pair of discrete (primal or dual) forms $\bar{\alpha}$ and $\bar{\beta}$ through:

$$\boxed{\mathbf{W}(\bar{\alpha}, \bar{\beta}) = \mathcal{P}(\mathcal{R}\bar{\alpha} \wedge \mathcal{R}\bar{\beta})}. \tag{3.5}$$

That is, we first reconstruct the two discrete forms with our trigonometric or Chebyshev basis functions, we then apply the wedge product to the two continuous forms that we have created, and finally, integrate (or, in the case of 0-forms, point-sample) the result to get the final discrete form $\mathbf{W}(\bar{\alpha}, \bar{\beta})$. The resulting operator satisfies a discrete Leibniz rule, equivalent to Eq. (2.1):

$$\mathbf{DW}(\bar{\alpha}, \bar{\beta}) = \mathbf{W}(\mathbf{D}(\bar{\alpha}), \bar{\beta}) + (-1)^k \mathbf{W}(\bar{\alpha}, \mathbf{D}(\bar{\beta})),$$

where $\bar{\alpha}$ is a discrete k-form, and $\bar{\beta}$ is an arbitrary discrete form. Indeed, because the exterior derivative commutes with the reduction and reconstruction maps, this discrete Leibniz rule can be derived directly from the continuous Leibniz rule. Our spectrally accurate definition of the discrete wedge product is bilinear and anticommutative, but it is *not* associative. It is, however, associative in the limit as the mesh size approaches zero. That is, the associator $\mathbf{W}(\bar{\alpha}, \mathbf{W}(\bar{\beta}, \bar{\gamma})) - \mathbf{W}(\mathbf{W}(\bar{\alpha}, \bar{\beta}), \bar{\gamma})$ (i.e., the measure of nonassociativity) will approach zero exponentially fast as the step size is reduced. The difficulty of creating an associative discrete wedge product has been previously noted by, e.g., Kotiuga [40], who refers to it as the "commutative cochain problem" and discusses some deeper topological reasons behind it. The spectral accuracy of our wedge operator will mitigate this lack of associativity exponentially fast under mesh refinement.

Note that the discrete wedge product of a (primal or dual) p-form and a (primal or dual) q-form can also be seen as a three tensor

$$\mathbf{W}^{p,q}_{ijk} = \int_{\sigma_i^{p+k}} \phi^p_j \wedge \phi^q_k,$$

since we can write the value of the wedge product on element σ_i^{p+q} as :

$$\left[\mathbf{W}(\bar{\alpha}, \bar{\beta})\right]_i = \sum_{j,k} \mathbf{W}_{ijk} \bar{\alpha}_j \bar{\beta}_k.$$

Discrete Hodge Star H

The different methods to discretize the Hodge star operator fall mostly in two categories. A majority of approaches start from a discretization of the inner product of forms (Eq. (2.3)), then derive a discrete Hodge star from it. This leads to

what is called the "*derived* discrete $\star$" in [7]. A few authors, instead, use a direct (called "*natural*" in [7]) definition of the Hodge star, which exploits the existence of primal/dual structures on meshes; i.e., the discrete Hodge star $\mathbf{H}$ now maps discrete *primal* forms to discrete *dual* forms, and vice-versa through the inverse of the Hodge star. Discretizations of the Hodge star typically rely on local reconstructions and integration, the simplest one being what is often referred to as the *diagonal Hodge star* [8], corresponding to a piecewise constant reconstruction. Its implementation involves each primal value being multiplied by the ratio of the measure of the primal and dual mesh elements to obtain the dual value. It is hence represented by a diagonal matrix. Higher-order Galerkin Hodge stars have also been considered, resulting in better accuracy at the cost of denser matrices. As we will see, by choosing proper basis functions, we will be able to construct a spectrally accurate *natural* discrete Hodge star that is computationally efficient, as the star will become a Toeplitz matrix in the periodic case.

As explained earlier, our discrete Hodge star $\mathbf{H}$ exploits the notion of mesh duality in that the discrete Hodge star of a primal k-form is a dual $(d-k)$-form, and vice-versa. The discrete Hodge star for a discrete form is realized conceptually by first reconstructing the continuous form with our primal (resp., dual) spectral bases, applying the continuous Hodge star to this form, and then projecting this form back to the dual (resp., primal) grid. In our notation, this can be written as

$$\mathbf{H}^k = \widetilde{\mathcal{P}} \star^k \mathcal{R}, \tag{3.6}$$

and is easily understood from the following diagram:

$$\begin{array}{ccc} \Lambda^k & \underset{\mathcal{R}}{\overset{\mathcal{P}}{\rightleftarrows}} & \bar{\Lambda}^k \\ \downarrow{\scriptstyle \star^k} & & \vdots{\scriptstyle \mathbf{H}^k} \\ \Lambda^{n-k} & \xrightarrow{\widetilde{\mathcal{P}}} & \widetilde{\Lambda}^{n-k} \end{array}$$

The operator $\mathbf{H}$ can be built as a matrix with easily precomputed entries. For instance, for k-forms, this matrix contains the terms

$$\mathbf{H}^k_{ij} = \int_{\widetilde{\sigma}_i^{n-k}} \star \phi_j^k. \tag{3.7}$$

Similarly, the dual Hodge star operator is given by:

$$\widetilde{\mathbf{H}}^k_{ij} = \int_{\sigma_i^{n-k}} \star \widetilde{\phi}_j^k. \tag{3.8}$$

Notice that if we require our discrete Hodge stars to satisfy the discrete equivalent of Eq. (2.2):

$$\mathbf{H}^k \widetilde{\mathbf{H}}^{n-k} = \widetilde{\mathbf{H}}^{n-k} \mathbf{H}^k = (-1)^{k(n-k)} \mathbf{Id}, \tag{3.9}$$

this imposes a constraint on dual basis functions. Indeed, for Eq. (3.9) to hold true, they must be a linear combination of the primal basis functions:

$$\tilde{\phi}_i^k = \sum_j [(\mathbf{H}^{n-k})^{-1}]_{ji} \star \phi_j^{n-k}. \tag{3.10}$$

Both the regular (ϕ) and Chebyshev (ψ) basis functions that we defined above satisfy this constraint.

Finally, our Hodge matrices are dense; however, each matrix is *circulant* for a regular grid on periodic domains (due to the invariance of the grid under translation), and *centrosymmetric* for a Chebyshev grid (due to mirror symmetry around its center). While circulant matrices are special cases of Toeplitz matrices, for which multiplications by vectors can be done $\mathcal{O}(N \log N)$, we will show that *all* discrete Hodge stars for arbitrary dimensions can be efficiently implemented in Fourier space in $\mathcal{O}(N \log N)$ time complexity where N is the total number of spatial points.

Other Operators

There are important additional operators, including contraction by vector field, and Lie derivative (extending directional derivatives to differential forms). While these operators play a crucial role in certain applications such as computational fluids and other advection problems, their extension to our spectral framework brings additional difficulty due to their dependence on a vector field. They can, however, be handled in our framework, but *only through the use of coordinates*, whereas in the continuous world the application of a Lie derivative on a differential form is a coordinate-free metric-independent operation. A first step towards a geometric discretization of contractions and Lie derivatives was proposed in [48]; but a spectrally accurate discretization preserving key properties of these other operators is left for future work.

3.5 Implementation in Fourier Space

We now describe how to efficiently implement the operators described in the last section. We will see that the Hodge star and wedge operators can have very fast implementations as they can be expressed as the product of fast Fourier transforms and sparse matrices. On periodic, regular grids of size N^d, the basic Dec operators

can be expressed in terms of the discrete equivalents of two elementary operators, namely a convolution with a box function over $[a, b]$ (denoted $\mathbf{I}^{a,b}$) and a translation by a (denoted $\mathbf{T}^{a}$). Both of these operations are expressed as diagonal matrices in Fourier space. The wedge product and Hodge star for Chebyshev grids require additional sparse operators that we describe next.

Sparse Matrices for Periodic 1D Domains

We define our convolution and translation operators to act on an array of function values $\bar{f}_i$ at evenly spaced points x_i on a periodic grid by first constructing a trigonometric interpolation $\mathcal{R}\bar{f}$ as described in Section 3.2, then by transforming it as follows:

$$\mathbf{I}^{a,b} : \bar{f}_i \mapsto \int_{x_i+a}^{x_i+b} \left(\mathcal{R}\bar{f}\right)(\xi)\, d\xi, \qquad \text{(integration)} \tag{3.11}$$

$$\mathbf{T}^{a} : \bar{f}_i \mapsto \left(\mathcal{R}\bar{f}\right)(x_i + a). \qquad \text{(translation)} \tag{3.12}$$

These two operators are implemented through *discrete Fourier transforms* as follows:

$$\mathbf{I}^{a,b} = \mathcal{F}^{-1}\widehat{\mathbf{I}}^{a,b}\mathcal{F}, \qquad \mathbf{T}^{a,b} = \mathcal{F}^{-1}\widehat{\mathbf{T}}^{a,b}\mathcal{F}.$$

The operator $\mathcal{F}$ represents the shifted *discrete Fourier transform* and $\mathcal{F}^{-1}$ represents its inverse: rather than placing the zero-frequency term (the mean of the signal) first, it is shifted to the middle with the negative Nyquist frequency becoming the first element, and the positive Nyquist frequency becoming the last.

In Fourier space (denoted using a hat $\widehat{\cdot}$), each operator is simply encoded by a diagonal matrix with the following coefficients:

$$\widehat{\mathbf{I}}^{a,b}_{kk} = \frac{e^{ikb} - e^{ika}}{ik}, \quad \text{and} \quad \widehat{\mathbf{T}}^{a}_{kk} = e^{ika},$$

where the special case $k = 0$ is treated as:

$$\lim_{k\to 0} \frac{e^{ikb} - e^{ika}}{ik} = b - a.$$

Note that since these two matrices are diagonal, the order in which they are applied does not matter.

Sparse Matrices for 1D Chebyshev Grids

For conciseness in later expressions, we also define a number of sparse matrices representing simple linear operators that will be particularly useful for the Hodge stars and wedge products on Chebyshev grids.

Mirroring The mirroring matrices will allow us to map quantities from the Chebyshev grid to the unit circle and back (see Fig 3.4). We will need four versions of mirroring: a mirroring $\mathbf{M}_0^+$ and anti-mirroring $\mathbf{M}_0^-$ operator for 0-forms mapping N nodal values of the Chebyshev grid to $2N-2$ values on the circle; and similarly, a mirroring $\mathbf{M}_1^+$ and anti-mirroring $\mathbf{M}_1^-$ operator mapping N nodal values of the Chebyshev grid to $2N$ values on the circle will be useful for 1-forms. Their expressions are:

$$\mathbf{M}_0^{\pm}: \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ \pm x_{N-2} \\ \vdots \\ \pm x_1 \end{bmatrix}, \qquad \mathbf{M}_1^{\pm}: \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ \pm x_{N-1} \\ \pm x_{N-2} \\ \vdots \\ \pm x_1 \\ \pm x_0 \end{bmatrix}.$$

The left inverses of the above operators, satisfying $\mathbf{M}_0^{\dagger}\mathbf{M}_0^{\pm} = \mathbf{Id}$ and $\mathbf{M}_1^{\dagger}\mathbf{M}_1^{\pm} = \mathbf{Id}$, are then given by

$$\mathbf{M}_0^{\dagger}: \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \\ \vdots \\ x_{2N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}, \qquad \mathbf{M}_1^{\dagger}: \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \\ \vdots \\ x_{2N} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}.$$

Extensions Extension matrices will allow us to resample forms at higher or lower resolutions, an operation needed in the wedge product as we will mention in Section 3.5. We define the upsampling extension matrix $\widehat{\mathbf{E}}^n$ that will act in Fourier

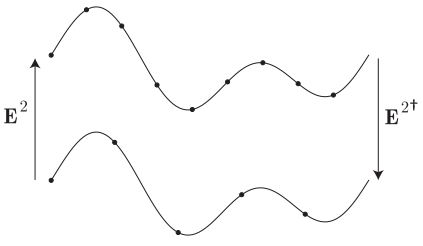
space to add $2n$ samples as follows:

$$\forall n > 0, \quad \widehat{\mathbf{E}}^n : \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \frac{N+2n}{N} \begin{bmatrix} \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} n \\ x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} n \end{bmatrix}.$$

The left inverse of this extension matrix is a downsampling matrix $\widehat{\mathbf{E}}^{n\dagger}$ such that $\widehat{\mathbf{E}}^{n\dagger}\widehat{\mathbf{E}}^n = \mathbf{Id}$, and is expressed as

$$\widehat{\mathbf{E}}^{n\dagger} : \begin{bmatrix} \left.\begin{matrix} x_{-n} \\ \vdots \\ x_{-1} \end{matrix}\right\} n \\ x_0 \\ \vdots \\ x_{N-1} \\ \left.\begin{matrix} x_N \\ \vdots \\ x_{N+n-1} \end{matrix}\right\} n \end{bmatrix} \mapsto \frac{N-2n}{N} \begin{bmatrix} \left.\begin{matrix} x_0 + x_N \\ \vdots \\ x_{n-1} + x_{N+n-1} \end{matrix}\right\} n \\ x_n \\ \vdots \\ x_{N-n-1} \\ \left.\begin{matrix} x_{N-n} + x_{-n} \\ \vdots \\ x_{N-1} + x_{-1} \end{matrix}\right\} n \end{bmatrix}.$$

With these extensions matrices, we can upsample a discrete form $\bar{\omega}$ from a N-point grid to a $(N{+}2n)$-point grid by first performing a Fourier transform $\mathcal{F}_N$, then applying $\widehat{\mathbf{E}^n}$, and applying an inverse Fourier transform $\mathcal{F}^{-1}_{N+2n}$ (see inset for an illustration of $\mathbf{E}^2$ and its left inverse):



$$\mathbf{E}^n\bar{\omega} = \mathcal{F}^{-1}_{N+2n}\,\widehat{\mathbf{E}}^n\,\mathcal{F}_N\,\bar{\omega}.$$

Downsampling is done similarly through $\mathbf{E}^{n\dagger}$, with now $\mathcal{F}^{-1}_N$ and $\mathcal{F}_{N+2n}$.

Scaling The metric dependent matrix $\mathbf{\Omega}^N$ is a diagonal matrix which has, as its diagonal elements, the weight by which each nodal value needs to be scaled in order for the integration over its corresponding dual region on the unit circle to faithfully represent integration over its corresponding dual region on the original Chebyshev grid. Unlike previous matrices in this section, Ω^N is metric-dependent in the sense that it does not only depend on the grid topology, but on the volume change between the grid on the circle and the Chebyshev grid:

$$\mathbf{\Omega}^N_{kk} = \sin\left(\frac{2\pi}{N}(k + \tfrac{1}{2}))\right) = \sqrt{1 - \widetilde{x}_k^2}.$$

Other Matrices Finally, we will need simple, additional matrices denoted $\mathbf{A}$, $\mathbf{B}$, $\mathbf{B}^\dagger$, and $\mathbf{C}$; they respectively lump every consecutive pair of non-boundary dual edge values, evaluate the nodal values of the 1-form basis functions of the left and right boundary dual edges (i.e., $\delta_N(\theta)$ and $\delta_N(\pi - \theta)$ in Section 3.3), and remove the end node values. Their expressions are thus:

$$\mathbf{A} = \underbrace{\begin{pmatrix} 1 & & & & & & \\ & 1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & \ddots & \ddots & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \\ & & & & & & 1 \end{pmatrix}}_{N\times(2N-2)} \quad \mathbf{C} = \underbrace{\begin{pmatrix} 0 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & 1 \\ & & & & & & 0 \end{pmatrix}}_{N\times N},$$

$$\mathbf{B} = \underbrace{\begin{pmatrix} (N-1)^2 + 1/2 & & \\ -1/2 & & \\ 1/2 & & \\ -1/2 & \mathbf{0} & \\ 1/2 & & \\ \vdots & & \\ (-1)^{N+1}/2 & & \end{pmatrix}}_{N\times N} \quad \mathbf{B}^\dagger = \underbrace{\begin{pmatrix} & & (-1)^{N+1}/2 \\ & & \vdots \\ & & 1/2 \\ \mathbf{0} & & -1/2 \\ & & 1/2 \\ & & -1/2 \\ & & (N-1)^2 + 1/2 \end{pmatrix}}_{N\times N}.$$

Wedge Product W

Let $\odot$ denote the element-wise multiplication (typically used in Fourier space) such that, for two vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a} \odot \mathbf{b}$ is the vector with components

$$(\mathbf{a} \odot \mathbf{b})_i = \mathbf{a}_i \mathbf{b}_i.$$

Remember that we are (conceptually) working with bandlimited functions $f(x)$ whose frequencies are below the Nyquist rate for the spatial grid resolution; i.e., with

functions that can be reconstructed exactly from their sample values $\{a_i = a(x_i)\}$ on the grid. However, when the element-wise product $\odot$ is applied to such functions, the result is no longer necessarily below the Nyquist rate, as the highest frequency can potentially be doubled by the multiplication. In order to compute wedge products *exactly within the Nyquist range*, we will have recourse to a temporary *upsampling* of the operands to a higher grid resolution, perform the multiplication at the higher resolution, and *downsample* back to the original grid resolution.

1D Wedge Based on our definition in Eq. (3.5), a wedge product of two 0-forms in 1D is expressed as

$$\bar{\omega}^0 = \mathbf{W}(\bar{\alpha}^0, \bar{\beta}^0) = \bar{\alpha}^0 \odot \bar{\beta}^0.$$

Note that there is no reason to use the up/downsampling operators here: while we obviously cannot capture the wedge product content past the Nyquist rate, all the frequencies below are exactly evaluated, thus leading to spectral accuracy. However, the wedge product of a 0-form and a 1-form differs, in the sense that to avoid losing accuracy, we must use the following definition:

$$\bar{\omega}^1 = \mathbf{W}(\bar{\alpha}^0, \bar{\beta}^1) = \mathbf{E}^{N\dagger}\mathbf{J}\left((\mathbf{E}^N\bar{\alpha}^0) \odot (\mathbf{E}^N\mathbf{J}^{-1}\bar{\beta}^1)\right),$$

where $\mathbf{J}$ maps from 0-forms to 1-forms (on the periodic domain) through

$$\mathbf{J} = \mathbf{I}^{-\frac{h}{2},\frac{h}{2}}\mathbf{T}^{\frac{h}{2}}.$$

Indeed, $\mathbf{J}$ being an integration, the wedge without $\mathbf{E}^N$ would lose accuracy at the Nyquist rate. Should dual forms be used, they could easily be converted to primal forms via the translation operator $\mathbf{T}$ before the wedge product is applied. Therefore, wedge products in 1D can always be implemented with efficient evaluation through discrete fast Fourier transforms, both for periodic and Chebyshev grids, since their expressions hold in each case.

2D and Beyond The various wedge products between forms can be expressed in terms of the following functions:

$$\begin{aligned}
\mathbf{w}^{00}(\mathbf{a}, \mathbf{b}) &= \mathbf{a} \odot \mathbf{b}, \\
\mathbf{w}^{01}\left(\mathbf{a}, \begin{pmatrix}\mathbf{b}_x \\ \mathbf{b}_y\end{pmatrix}\right) &= \begin{pmatrix}\mathbf{a}\odot\mathbf{b}_x \\ \mathbf{a}\odot\mathbf{b}_y\end{pmatrix}, \\
\mathbf{w}^{11}\left(\begin{pmatrix}\mathbf{a}_x \\ \mathbf{a}_y\end{pmatrix}, \begin{pmatrix}\mathbf{b}_x \\ \mathbf{b}_y\end{pmatrix}\right) &= \mathbf{a}_x \odot \mathbf{b}_y - \mathbf{a}_y \odot \mathbf{b}_x,
\end{aligned}$$

and similarly for higher dimensions.

Now, let:

$$\mathbf{J_x} \equiv \mathbf{J} \otimes \mathbf{Id}, \qquad \mathbf{J_y} \equiv \mathbf{Id} \otimes \mathbf{J}, \qquad \mathbf{J_{xy}} \equiv \mathbf{J} \otimes \mathbf{J},$$
$$\mathbf{E} \equiv \mathbf{E}^N \otimes \mathbf{E}^N, \qquad \mathbf{E}^\dagger \equiv \mathbf{E}^{N\dagger} \otimes \mathbf{E}^{N\dagger}.$$

Then we can define the wedge product in 2D for all possible pairs of forms as:

$$\begin{aligned}
\mathbf{W}(\bar{\alpha}^0, \bar{\beta}^0) &= \mathbf{w}^{00}\left(\bar{\alpha}^0, \bar{\beta}^0\right), \\
\mathbf{W}\left(\bar{\alpha}^0, \begin{pmatrix} \bar{\beta}^1_x \\ \bar{\beta}^1_y \end{pmatrix}\right) &= \begin{pmatrix} \mathbf{E}^\dagger \mathbf{J_x} & \\ & \mathbf{E}^\dagger \mathbf{J_y} \end{pmatrix} \mathbf{w}^{01}\left(\mathbf{E}\bar{\alpha}^0, \begin{pmatrix} \mathbf{E}\mathbf{J_x}^{-1}\bar{\beta}^1_x \\ \mathbf{E}\mathbf{J_y}^{-1}\bar{\beta}^1_y \end{pmatrix}\right), \\
\mathbf{W}\left(\begin{pmatrix} \bar{\alpha}^1_x \\ \bar{\alpha}^1_y \end{pmatrix}, \begin{pmatrix} \bar{\beta}^1_x \\ \bar{\beta}^1_y \end{pmatrix}\right) &= \mathbf{E}^\dagger \mathbf{J_{xy}} \mathbf{w}^{11}\left(\begin{pmatrix} \mathbf{E}\mathbf{J_x}^{-1}\bar{\alpha}^1_x \\ \mathbf{E}\mathbf{J_y}^{-1}\bar{\alpha}^1_y \end{pmatrix}, \begin{pmatrix} \mathbf{E}\mathbf{J_x}^{-1}\bar{\beta}^1_x \\ \mathbf{E}\mathbf{J_y}^{-1}\bar{\beta}^1_y \end{pmatrix}\right), \\
\mathbf{W}(\bar{\alpha}^0, \bar{\beta}^2) &= \mathbf{E}^\dagger \mathbf{J_{xy}} \mathbf{w}^{00}\left(\mathbf{E}\bar{\alpha}^0, \mathbf{E}\mathbf{J_{xy}}^{-1}\bar{\beta}^2\right).
\end{aligned}$$

Here again, the use of up/downsampling is purely to ensure no loss of accuracy for frequencies at or below the Nyquist rate. Extension of the wedge product to dual forms or in higher dimensions is straightforward as the same principles apply. While the above derivation was for periodic domains, the wedge product is preserved under pullback; so the equations hold for any topologically equivalent space. For Chebyshev grids, $\mathbf{J}$ is no longer square, and there are only $N - 1$ edges—but the formulas are basically unchanged.

Hodge Star H

We can also express our discrete Hodge stars in a way that lends itself to fast evaluations.

Periodic Grid in 1D The Hodge star for a periodic grid is very simple, as it is simply expressed as

$$\mathbf{H}^0 = \widetilde{\mathbf{H}}^0 = \mathbf{I}^{-\frac{h}{2},\frac{h}{2}} \qquad \mathbf{H}^1 = \widetilde{\mathbf{H}}^1 = \mathbf{I}^{-\frac{h}{2},\frac{h}{2}{}^{-1}}. \tag{3.13}$$

Chebyshev Grid in 1D To define the Hodge-Star operator on a Chebyshev grid, we make use of the matrices we defined earlier in Section 3.5. In particular, we get

$$\begin{aligned}
\widetilde{\mathbf{H}}^0 &= \mathbf{M_1}^\dagger \mathbf{I}^{-\frac{h}{2},\frac{h}{2}} \mathbf{\Omega} \mathbf{M}_1^+, \\
\mathbf{H}^1 &= \mathbf{M_1}^\dagger \mathbf{\Omega}^{-1} \mathbf{I}^{-\frac{h}{2},\frac{h}{2}{}^{-1}} \mathbf{M}_1^-.
\end{aligned}$$

The expressions for the other two stars are slightly more complicated because of the special treatment needed for the dual (half-)edges at the two ends of the domain:

$$\begin{aligned}
\mathbf{H}^0 &= \mathbf{A}\mathbf{M}_0^\dagger \mathbf{I}^{0,+\frac{h}{2}} \mathbf{\Omega} \mathbf{E}^{N-1} \mathbf{M}_0^+ \\
\widetilde{\mathbf{H}}^1 &= \mathbf{M}_0^\dagger \left(\mathbf{T}^{-\frac{h}{2}} \mathbf{\Omega}^{-1} \mathbf{T}^{\frac{h}{2}} - \mathbf{B}\mathbf{I}^{0,\frac{h}{2}} - \mathbf{B}^\dagger \mathbf{I}^{-\frac{h}{2},0}\right) \mathbf{I}^{-\frac{h}{2},\frac{h}{2}{}^{-1}} \mathbf{M}_0^- \mathbf{C} + \mathbf{B} + \mathbf{B}^\dagger.
\end{aligned}$$

2D and beyond The extension of the fast 1D implementation to arbitrary dimension is rather direct: we construct Hodge stars on higher-dimensional grids by taking the Cartesian products of stars on one-dimensional grids. Using 2D for illustration purposes, we first introduce the following operators

$$\mathbf{H_x} \equiv \mathbf{H} \otimes \mathbf{Id}, \qquad \mathbf{H_y} \equiv \mathbf{Id} \otimes \mathbf{H}.$$

where $\mathbf{H_x}$ acts along the first index (x-direction), and $\mathbf{H_y}$ acts along the second index (y-direction). Then from these two 1D Hodge stars, the 2D Hodge star operators can be expressed as

$$\begin{aligned} \mathbf{H^0} &= \mathbf{H^0_x H^0_y}, \\ \mathbf{H^1} &= \begin{pmatrix} \mathbf{0} & -\mathbf{H^0_x H^1_y}, \\ \mathbf{H^1_x H^0_y} & \mathbf{0} \end{pmatrix} \\ \mathbf{H^2} &= \mathbf{H^1_x H^1_y}. \end{aligned}$$

Equivalent equations hold for the dual Hodge stars. Note that we have $\mathbf{H^1 H^1} = -\mathbf{Id}$ as desired. For completeness, we also provide the 2D exterior derivative in terms of the Cartesian products of the 1D $\mathbf{D_x}$ and $\mathbf{D_y}$ operators:

$$\mathbf{D^0}\bar{\alpha}^0 = \begin{pmatrix} \mathbf{D_x}\bar{\alpha}^0 \\ \mathbf{D_y}\bar{\alpha}^0 \end{pmatrix}, \qquad \mathbf{D^1}\begin{pmatrix} \bar{\alpha}^1_x \\ \bar{\alpha}^1_y \end{pmatrix} = -\mathbf{D_y}\bar{\alpha}^1_x + \mathbf{D_x}\bar{\alpha}^1_y$$

.

3.6 Numerical Tests

We now discuss how our spectral operators can be used for practical computations. In particular, we present numerical tests involving Poisson equations on 0- and 1-forms in 1D and 2D.

Solving Differential Equations

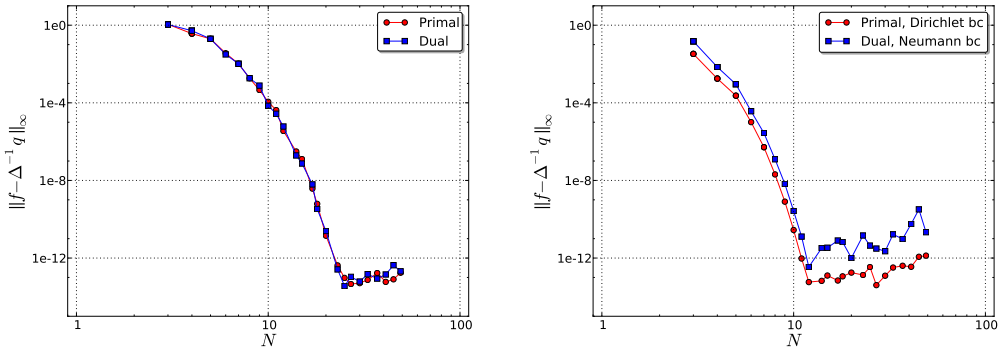
Our spectral framework can be used in various ways, either as is, or with minor modifications.

Chain Collocation Our computational framework lends itself naturally to a spectral method that we call the *Chain Collocation Method*. Just as the point collocation method enforces a partial differential equation on all grid nodes, we now enforce partial differential equations on all appropriate chains. If the unknown is a scalar function, then the PDE is enforced at every node as usual; but for a k-form, the PDE

will now be enforced on each k-chain. Since any differential equation can be rewritten with exterior forms, our operators can be used as is on *discrete* approximations of these forms, and the conversion of the continuous operators into discrete operators combining $\mathbf{D}, \mathbf{H}$, and $\mathbf{W}$ will enforce the differential equation on each appropriate mesh element. For instance, a 2D Poisson equation $\Delta \mathbf{v} = \mathbf{f}$ for a *vector field* $\mathbf{v}$ can be rewritten with, for instance, a 1-form ω as $\Delta\omega = \gamma$ where ω is now the circulation of the continuous field $\mathbf{v}$ on all edges, while γ is the circulation of $\mathbf{f}$—leading to a coordinate-free setup, drastically different from the usual coordinate-based treatment. Modulo boundary conditions, a direct discretization of the Laplace operator $\Delta = \mathbf{d} \star \mathbf{d} \star + \star \mathbf{d} \star \mathbf{d}$ will now enforce that a discrete form $\bar{\omega}$ satisfies the PDE in a discrete sense, i.e., on each edge. (Various 1D and 2D examples of this approach are shown at the end of this section.) Because of the algebraic topology properties of our framework, conservative differential equations will lead to conservative discrete equations, and structure-preservation will be enforced.

Other Numerical Approaches Our framework can be easily altered to accommodate other numerical solves. First, different grids can be used if necessary, as we spelled out the constraints that one needs to enforce to get proper discretizations. Second, one can modify our Hodge stars by deriving them from the inner product instead: this is the approach spelled out in [55, 28]. While the Hodge star operators are no longer converting primal to dual forms (and vice-versa), the advantage of this approach is that the resulting operators inherit the symmetry of the inner product.

Logically rectangular grids Finally, modifying our approach for arbitrary logically rectangular computational grids is simple. First, both $\mathbf{D}$ and $\mathbf{W}$ are unchanged since the integral of forms are unchanged by the pullback φ^* and our numerical treatment of these operators is metric independent. The Hodge star operator needs, however, to be modified. In fact, only the matrices Ω_N require modification: one simply has to rescale the diagonal elements of Ω_N by how much the orthogonal Chebyshev grid is deformed into the computational grid. In this sense, our spectral framework is reminiscent of the regular Dec framework, as the diagonal Hodge stars are also subject to be changed based on the volume forms of the computational domains [21].



(a) Periodic Domain **(b)** Bounded Domain

Figure 3.7: Convergence graphs for a 1D Poisson equation: (a) we solve $\Delta f = e^{\sin x}(\cos^2 x - \sin x)$ on a periodic domain for either a primal, or dual 0-form f; (b) we solve $\Delta f = e^x$ on a Chebyshev grid, for either a primal 0-form with Dirichlet boundary conditions $f(-1) = e^{-1}$ and $f(1) = e$, or for a dual 0-form with Neumann boundary conditions $f'(-1) = e^{-1}$ and $f'(+1) = e$. All of our results exhibit spectral convergence (measured through the L^∞ error $\|f - \Delta^{-1}q\|_\infty$), with the conventional plateau when we reach the limit of accuracy of the representation of floating point numbers.

Convergence Tests

In order to provide numerical evidence of spectral behavior, we tested our operators on a variety of examples. We show below a series of Poisson equations with Dirichlet and Neumann conditions in 1D and 2D.

1D Poisson The traditional Poisson equation for a scalar function f reads:

$$\nabla^2 f = q,$$

which can be rewritten in exterior calculus as an equation on a 0-form f:

$$\star \mathbf{d} \star \mathbf{d} f = q.$$

Using a periodic domain (where a regular grid is placed at $[0, 2\pi]$) or a bounded domain (chosen to be $[-1, +1]$, discretized by a Chebyshev grid), we can directly discretize this last equation. The only choice we are left with is to decide whether to think of f as a primal ($\bar{f}$) or a dual ($\tilde{f}$) 0-form. Thus, the 1D Poisson equation is implemented as either

$$\widetilde{\mathbf{H}}^1 \widetilde{\mathbf{D}}^1 \mathbf{H}^1 \mathbf{D}^0 \bar{f}^0 = \bar{q}^0$$

or

$$\mathbf{H}^1 \mathbf{D}^1 \widetilde{\mathbf{H}}^1 \widetilde{\mathbf{D}}^0 \tilde{f}^0 = \tilde{q}^0.$$

Note that the boundary conditions (Dirichlet or Neumann) are simply incorporated into the **D** operators: prescribed primal values or dual values at the boundary will be used as is instead of being considered unknowns. This is substantially simpler than the traditional use of, e.g., generalized Lagrange polynomials for the treatment of boundary conditions. The convergence plots are shown in Fig 3.7, proving spectral convergence.

2D Poisson We also tested our approach on 2D problems. This time, we computed Poisson equations not only for scalar functions, but also for vector fields—or in exterior calculus jargon, we can compute the Laplacian $\Delta = \star\mathbf{d}\star\mathbf{d} + \mathbf{d}\star\mathbf{d}\star$ for 0-, 1-, and 2-forms. Given that these forms can also be performed on the primal or the dual, we now have six discrete versions of the Laplace operator:

$$\mathcal{P}\left(\star\mathbf{d}\star\mathbf{d}+\mathbf{d}\star\mathbf{d}\star\right)\mathcal{R} = \begin{cases} \widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^1\mathbf{H}^1\mathbf{D}^0 & \text{for primal 0-forms} \\ \mathbf{H}^1\mathbf{D}^1\widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^0 & \text{for dual 0-forms} \\ \widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^0\mathbf{H}^2\mathbf{D}^1 + \mathbf{D}^0\widetilde{\mathbf{H}}^2\widetilde{\mathbf{D}}^1\mathbf{H}^1 & \text{for primal 1-forms} \\ \mathbf{H}^1\mathbf{D}^0\widetilde{\mathbf{H}}^2\widetilde{\mathbf{D}}^1 + \widetilde{\mathbf{D}}^0\mathbf{H}^2\mathbf{D}^1\widetilde{\mathbf{H}}^1 & \text{for dual 1-forms} \\ \mathbf{D}^1\widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^0\mathbf{H}^2 & \text{for primal 2-forms} \\ \widetilde{\mathbf{D}}^1\mathbf{H}^1\mathbf{D}^0\widetilde{\mathbf{H}}^2 & \text{for dual 2-forms.} \end{cases}$$

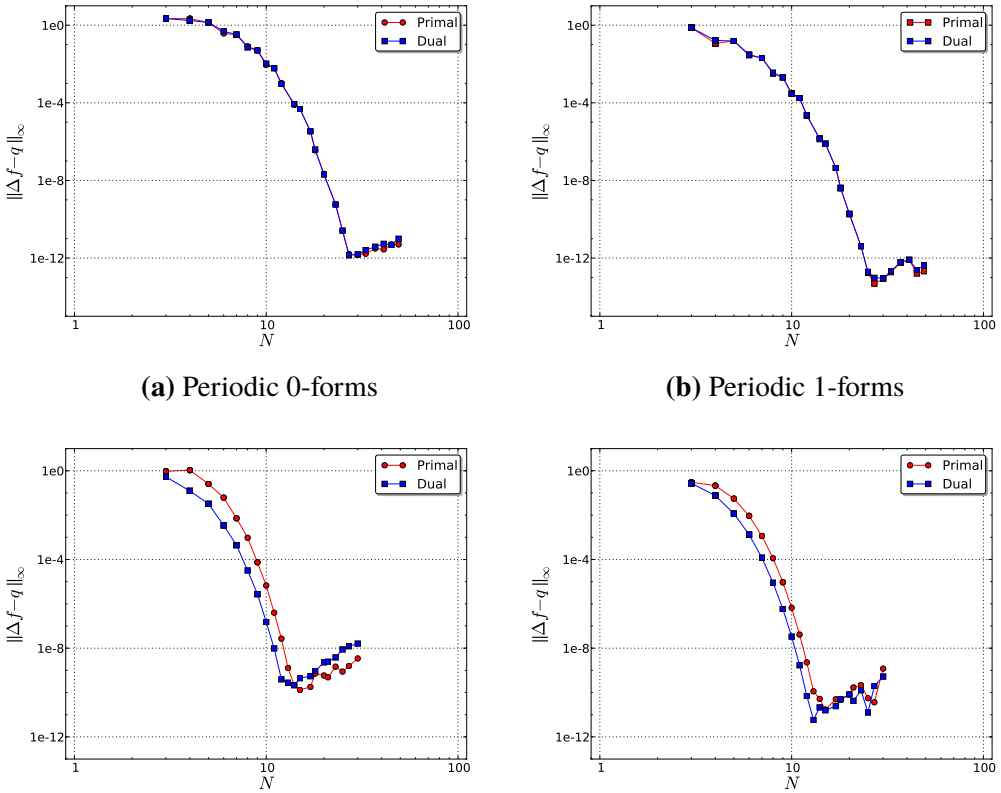
The convergence plots, shown in Fig 3.8, demonstrate spectral convergence again.

Finally, to distinguish our approach from other conventional discretization techniques and reinforce the point that our spectral method is not solely limited to scalar functions, we depict a solution of the Poisson equation for primal *1-forms* (i.e., the equivalent of a vectorial Poisson equation) on Fig 3.9.

3.7 Conclusions and Future Work

In this chapter, we laid out the foundations for a spectral instance of *discrete exterior calculus*. Using chains and cochains as discretization for differential forms, we provide a discrete exterior derivative, a discrete Hodge star, and a discrete wedge product that are spectrally accurate. One can use the resulting calculus as a chain collocation method to solve differential equations.

While the three exterior operators we treated cover already a large range of differential operators (allowing a number of computational tasks including Hodge decomposition), contractions (exterior product) and Lie derivatives still need to be derived in a similar geometric, coordinate-free manner—currently, they can only



(a) Periodic 0-forms

(b) Periodic 1-forms

(c) Chebyshev 0-forms

(d) Chebyshev 1-forms

Figure 3.8: Convergence graphs for a 2D Poisson equation; (a) we solve $\Delta f = e^{\sin x}(\cos^2 x - \sin x) + e^{\sin y}(\cos^2 y - \sin y)$ on a periodic domain for either a primal, or dual 0-form f; (b) now for $\Delta f = e^{\sin x}(\cos^2 x - \sin x)\mathbf{d}x + e^{\sin y}(\cos^2 y - \sin y)\mathbf{d}y$; (c) we solve $\Delta f = e^x + e^y$ on a Chebyshev grid, for either a primal 0-form with Dirichlet boundary conditions $f(x, y) = e^x + e^y$, or for a dual 0-form with Neumann boundary conditions $\nabla f(x) \cdot \mathbf{n} = (e^x \; e^y)^t \; \mathbf{n}$; (d) now for $\Delta f = e^x \mathbf{d}x + e^y \mathbf{d}y$. All of our results exhibit spectral convergence (measured through the L^∞ error $\|\Delta f - q\|_\infty$), with the conventional plateau in accuracy for fine meshes.

be handled using coordinates. Note that defining the contraction operator would be enough, since one can then define the Lie derivative algebraically with Cartan's "magic formula" [22]. But a direct dynamic ("flowed-out") definition as presented in [48] may instead prove beneficial.

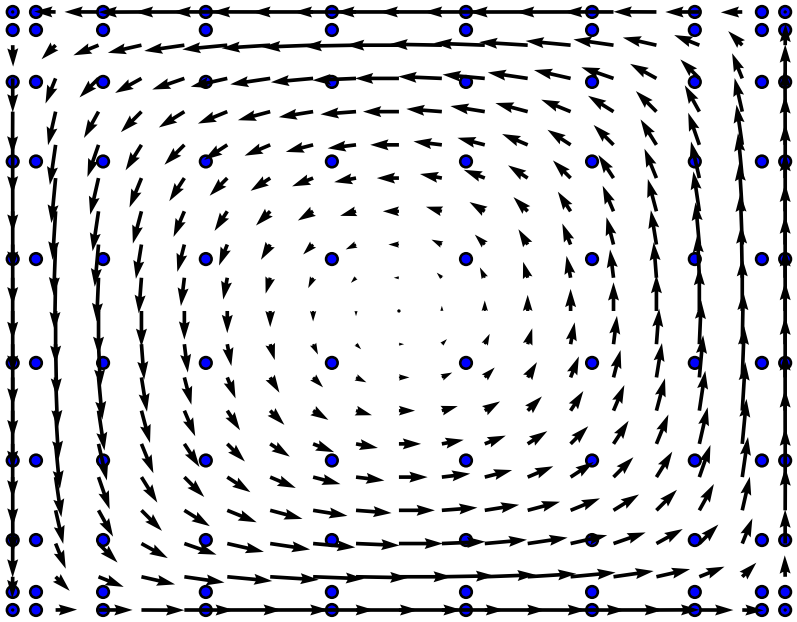


Figure 3.9: Plot for the solution of $(\star\mathbf{d}\star\mathbf{d}+\mathbf{d}\star\mathbf{d}\star)f=0$ with the boundary conditions of $\star f|_{\mathcal{B}}=0$ (vector field is tangent to the boundary) and $\star\mathbf{d}\star f|_{\mathcal{B}}=1$ (the curl at the boundary is equal to 1). The domain is $[-1,1]^2$, discretized by a 10×10 Chebyshev grid.

Chapter 4

SPECTRAL EXTERIOR CALCULUS

4.1 Introduction

Continuum mechanics formulates the governing equations of deformable bodies and fluids using operators from integral and differential calculus on Riemannian manifolds. While the mathematics of Riemannian geometry has been long established, computational methods require the discretization of domains, fields, and operators since numerical simulation necessitates finite-dimensional approximations—and many questions remain open about what are the best approaches to discretize the aforementioned continuum theories. Discretization of the domain geometry is often done through the design of a simplicial or cell complex, while fields and operators are approximated using finite element basis functions and weak formulations. The locality of the basis functions used in computations directly influences both the computational complexity (i.e., simulation times and memory usage) and the accuracy (i.e., order of approximation) of the resulting computations.

Spectral methods have gained popularity in the computation of continuum mechanics problems whose solutions are sufficiently smooth: their use leads to exponential rates of convergence (referred to as *spectral convergence* [17, 69]). Among the various spectral methods, spectral collocation that uses global smooth trial functions and Dirac test functions at collocation points is particularly attractive due to its simplicity and its computational efficiency.

Recent years have also seen the development of novel discretizations aimed at preserving in the discrete realm the underlying geometric, topological, and algebraic structures at stake in differential equations. This has proven to be a fruitful guiding principle for discretization [7, 2, 21, 3]. This geometric approach has led to new *structure-preserving* numerical methods, analyzed in, e.g., [3, 36], that inherit a variety of properties from the continuous world.

Yet, combining structure preservation with spectral convergence is far from a simple matter. Only a handful of works have been proposed [28, 56, 57, 62, 29], in which the foundations of exterior calculus [1] are adapted to spectral methods to offer spectrally accurate treatments of various operators. To this day, there is no unifying framework

covering all the exterior calculus operators needed for computational modeling that offers an efficient implementation of generic computations with structure-preserving and spectral properties.

In this paper, we present SPEX, an exterior calculus based spectral approach to computations on simply-connected domains of arbitrary dimension, equipped with an arbitrary Riemannian metric. We leverage two complementary discrete representations of differential forms to offer an efficient evaluation of both metric-dependent and metric-independent operators: namely, we represent k-forms via their components at every mesh node and via their integral over mesh k-dimensional elements. The resulting spectral calculus extends previous approaches [57, 62] and provides efficient evaluations of all the exterior operators ($\mathbf{d}, \star, \wedge, \lrcorner, \mathcal{L}$) and the musical isomorphisms $\flat$ and $\sharp$ (to convert vector fields to one-forms and vice-versa), while preserving many of the topological properties and features of the continuum theory.

4.2 Context and Motivations

Computational methods that inherently preserve geometric structures have become increasingly popular over the past few years. They have gained acceptance among engineers and mathematicians [4] by deriving numerical evaluations based on well-established geometric foundations of calculus on smooth manifolds as we briefly review next.

Exterior Calculus Based Computations

Exterior calculus [1] is a concise mathematical formalism that expresses differential and integral equations on smooth and curved spaces while revealing important geometric and topological structures behind integration and derivation. At the heart of exterior calculus is the notion of differential forms (pioneered by Cartan), denoting antisymmetric tensors of arbitrary rank. Formally, a k-form on a manifold $\mathcal{M}$ is a multi-linear map that takes, at every point, k tangent vectors as input and returns a real number; in this paper, we will denote the space of k forms as Λ^k and the space of vector fields as $\mathfrak{X}$. Note that consequently, Λ^0 is simply the space of scalar functions on $\mathcal{M}$, and that Λ^1 and $\mathfrak{X}$ are dual in the algebraic sense, i.e., one-forms can be understood as covectors since a one-form applied to a vector field returns a scalar function. As integration of differential forms is an abstraction of the process of measurement, this form-based calculus provides an intrinsic, coordinate-free approach to describing a multitude of physical models that make heavy use of line, surface and volume integrals [15, 25, 26]. Moreover, physical

measurements (such as fluxes or currents) typically represent local integrations over a small line, surface or volume element of the measuring instrument: pointwise evaluation of such quantities does not have physical meaning. Thus, one should manipulate these quantities as geometrically-meaningful entities integrated over appropriate submanifolds. With differential forms as its building blocks, exterior calculus consists of seven operators as listed in Table 4.1, and compositions thereof such as the inner product of forms and the Laplace-Beltrami operator. (Note that the contraction of a one-form α by a vector field X, $X \lrcorner \alpha$, is sometimes written as $\langle \alpha, X \rangle$, or as $\mathbf{i}_X \alpha$, or as $\alpha(X)$ depending on the authors.)

Name	**Symbol**	**Domain → Range**	**Metric**
Derivative	$\mathbf{d}$	$\Lambda^k \to \Lambda^{k+1}$	no
Wedge product	$\wedge$	$\Lambda^k \times \Lambda^l \to \Lambda^{k+l}$	no
Contraction	$\lrcorner$	$\mathfrak{X} \times \Lambda^k \to \Lambda^{k-1}$	no
Hodge star	$\star$	$\Lambda^k \to \Lambda^{n-k}$	yes
Flat	$\flat$	$\mathfrak{X} \to \Lambda^1$	yes
Sharp	$\sharp$	$\Lambda^1 \to \mathfrak{X}$	yes
Inner product	$\_ \cdot \_ \equiv \star(\_ \wedge \star \_)$	$\Lambda^k \times \Lambda^k \to \Lambda^0$	yes
Laplace-Beltrami	$\Delta \equiv \mathbf{d} \star \mathbf{d} \star + \star \mathbf{d} \star \mathbf{d}$	$\Lambda^k \to \Lambda^k$	yes
Lie derivative	$\mathcal{L} = \_ \lrcorner \mathbf{d} \_ + \mathbf{d} \_ \lrcorner \_$	$\mathfrak{X} \times \Lambda^k \to \Lambda^k$	no

Table 4.1: List of basic and a few compound operators of exterior calculus, with their dependence on the metric being indicated in the last column.

(Note that the contraction of a one-form α by a vector field X, $X \lrcorner \alpha$, is sometimes written as $\langle \alpha, X \rangle$, or as $\mathbf{i}_X \alpha$, or as $\alpha(X)$ depending on the authors.) This exterior calculus notation is very convenient and universal, as it allows us to write a number of key theorems and physical models on arbitrary (flat or curved) domains in a very compact manner that is independent of any particular choice of coordinate system. The Laplace operator Δ for instance encapsulates both the scalar Laplacian (used in diffusion) when applied to zero-forms, and the vector Laplacian (used in the wave equation for the electric field in E&M) when applied to one-forms. Stokes' theorem (which supersedes Green's, Kevin's, and Gauss's theorems) states that the integral of the derivative $\mathbf{d}$ of an arbitrary form α over a domain $\mathcal{D}$ is equal to the integral of the said form over the boundary $\partial\mathcal{D}$ of $\mathcal{D}$, i.e., $\int_{\mathcal{D}} \mathbf{d}\alpha = \int_{\partial\mathcal{D}} \alpha$; that is, $\mathbf{d}$ and the boundary operator ∂ are adjoint with respect to integration. In fluid dynamics, the incompressible Navier-Stokes equations on arbitrary domains in the case of a fluid

density being constant in time and uniform in space are simply

$$\begin{cases} \mathbf{v}_t + \mathcal{L}_{\mathbf{v}^\sharp}\mathbf{v} = -\mathbf{d}p + \dfrac{1}{\mathrm{Re}}\Delta\mathbf{v}, \\ \mathbf{d} \star \mathbf{v} = 0, \end{cases}$$

where $\mathbf{v}$ is the velocity one-form, p is the pressure zero-form, and Re is the dimensionless Reynolds number measuring the importance of viscous forces relative to inertial forces. Maxwell's equations of electromagnetism can also be formulated very compactly using the operators of exterior calculus:

$$\begin{aligned} \mathbf{dF} &= 0, \\ \mathbf{d} \star \mathbf{F} &= \mathbf{J}, \end{aligned}$$

where $\mathbf{F}$ is the Faraday 2-form (incorporating the electric and magnetic fields) and $\mathbf{J}$ is the electric current 3-form (incorporating the current and electric charge densities) [64]. Typical boundary conditions (Dirichlet, Neumann) are also trivially expressed with exterior operators and inclusions maps.

Finite Dimensional Exterior Calculus

Discrete exterior calculus endeavors to extend exterior calculus to discrete spaces, be they graphs or manifold meshes. Behind the various efforts in this direction [8, 22, 2, 31, 60, 30] is the desire to preserve as many properties as possible of the continuous forms and exterior calculus operators in the finite-dimensional case for computational purposes. For instance, enforcing Stokes' theorem at the discrete level guarantees various conservation properties, while a proper notion of a discrete Hodge decomposition prevents spurious modes in solutions. Most of these efforts use the notion of chains and cochains from algebraic topology to discretize manifolds and differential forms, respectively. Integration is thus seen as the pairing between chains (linear combination of mesh elements) and cochains, and Stokes' theorem is enforced by construction. This approach, creating an entire discrete de Rham complex, extends traditional finite element methods with now (low- or high-order) Whitney basis functions associated to each element of a (simplicial or cell) complex that represents the computational domain: k-forms can thus be histopolated (i.e., reconstructed from their integrated values on chains) over the whole domain.

Spectral Chain Based Methods

While finite element methods use locally supported basis functions, spectral methods favor globally supported basis functions instead, combined with the use of the

fast Fourier transform for efficiency of computation. They are often a preferred approach for problems where the solution is smooth because their global nature offers exponential convergence under refinement. However, structure preservation was initially not a topic of interest for spectral methods. In fact, the point collocation method, one of the most common spectral approaches, computes derivatives nodewise in Fourier space, thus failing to satisfy Stokes' theorem.

A decade ago, Mai-Duy et al. [44] proposed to modify the spectral collocation method by using *integrated* Chebyshev polynomials to numerically solve biharmonic differential equations with easier enforcement of boundary conditions. Shortly after, Gerritsma [28] described a method for constructing higher order *edge basis functions*, a staple of discrete counterparts to exterior calculus, to interpolate 1-forms. From there on, a few approaches were recently formulated that marry structure preservation properties and spectral accuracy [56, 57, 62, 29]. The ingredients at the core of these methods are quite similar: all rely on the use of cochains as a finite-dimensional representation of forms and they introduce spectral approximations of Hodge stars. Most related to our contribution, the work of [62] presents a spectrally accurate calculus of discrete differential forms described as a "chain collocation method" as it extends the traditional point collocation method to collocations over any type of mesh elements.

Limitations of Previous Work

While discrete, structure-preserving versions of exterior calculus offering spectral accuracy are readily available, the current state of the art is limited in various ways.

- While exterior derivative, Hodge star, and wedge product have been formulated in these frameworks, there are currently no associated contraction and Lie derivative that fit the same computational framework: previous approaches rely on high-order accurate extrusion of chains [47] or invoke pointwise notions of forms [29]) instead of cochains—both incompatible with existing spectral cochain-based methods.

- In existing works, vector fields and 1-forms are usually not differentiated: vector fields are simply treated through their corresponding one-forms. Yet these two entities are clearly different from a geometric perspective (Lie derivatives of forms vs. vector fields, for instance, are clearly distinct operations), as they do not even transform similarly under a change of basis.

Symbol	Meaning
d	Dimension of the manifold
$\mathcal{D}$	Reference domain of dimension d
$\mathcal{M}$	Manifold of dimension d
φ	Diffeomorphism $\varphi : \mathcal{D} \to \mathcal{M}$
$\bar{\Sigma}^k$	Primal k-dim cells of $\mathcal{M}$
$\widetilde{\Sigma}^k$	Dual k-dim cells of $\mathcal{M}$
$\mathring{\Sigma}$	Component 0-dim cells (vetices) of $\mathcal{M}$
$\bar{\sigma}_i^k$	Primal k-dim cell of grid $\bar{\Sigma}^k$ with index i
$\widetilde{\sigma}_i^k$	Dual k-dim cell of grid $\widetilde{\Sigma}^k$ with index i
$\bar{\phi}_i^k$	Primal basis function for k-forms associated with $\bar{\sigma}_i^k$
$\widetilde{\phi}_i^k$	Dual basis function for k-forms associated with $\widetilde{\sigma}_i^k$
Λ^k	Space of differential k-forms ω^k on $\mathcal{M}$
$\bar{\Lambda}^k$	Space of primal discrete k-forms $\bar{\omega}^k$ on $\bar{C}\mathcal{M}$
$\widetilde{\Lambda}^k$	Space of dual discrete k-forms $\widetilde{\omega}^k$ on $\widetilde{C}\mathcal{M}$
$\mathring{\Lambda}^k$	Space of component k-forms $\widetilde{\omega}^k$ on $\bar{C}^0\mathcal{M} \cup \widetilde{C}^0\mathcal{M}$
$\mathfrak{X}$	Space of vector fields on $\mathcal{M}$ (dual of Λ^1)
$\mathring{\mathfrak{X}}$	Space of discrete component vector fields on $\mathcal{M}$
$*$	Poincaré duality operator ($*\bar{\sigma}_i^k = \widetilde{\sigma}_i^{d-k}$)
$\boldsymbol{\partial}$	Boundary operator on grid elements

Table 4.2: Meaning of the basic notations used throughout this document.

- The notion of discrete metric is also rarely discussed in existing works. In particular, the derivation of Hodge stars from arbitrary metrics is not available, and neither are the musical isomorphisms $\flat$ and $\sharp$ which lower and raise indices for one-forms and vector fields.

Overcoming these limitations will require enriching the cochain-based representation that is the basis of all spectral calculus approaches thus far to offer a fast, and spectrally accurate formulation of *all* the exterior calculus operators.

Spex Overview

We now formulate the basic design principles that we will use to derive our *spectral exterior calculus*.

Pointwise vs. Integral Representations The operators derived in the context of existing spectral calculus frameworks have been mostly unary. However, binary

operators such as the wedge product $\wedge$ and the contraction $\lrcorner$, which have two inputs, require combining components through multiplication. Therefore any discrete form that is not associated with a vertex (i.e., with a degree higher than zero) must be converted to a component representation at vertices that are collocated to allow for direct pointwise multiplication. Additionally, since multiplication can double the spectral band, the multiplication has to be performed on a refined grid with twice as many vertices along each dimension in order to properly resolve the result. This necessitates the introduction of a component-based representation of differential forms, which we will call Component Forms, for which the space of all component forms will be denoted $\mathring{\Lambda}$. Additionally, a space of Component Vector Fields (denoted $\mathring{\mathfrak{X}}$) will also be similarly defined, so that the operators $\flat$ and $\sharp$ providing the mapping to and from Component One Forms $\mathring{\Lambda}^1$ can be easily implemented. Importantly, we will show that being able to easily convert between these twin (chain-based and component-based) representations will be crucial to the efficient implementation of our operators. The various maps between the spaces of continuous and discrete forms are shown in Figure 4.10.

Chain Based Exterior Derivative Discrete and finite element exterior calculus strive to mimetically reproduce exterior calculus on finite degrees of freedom. With this goal in mind, the use of chains and cochains was shown to be the most natural approach for constructing a discrete counterpart of the continuous **d** operator: its topological nature and adjointness to the boundary operators make it best described in the discrete sense purely based the incidence of elements of the computational grid, as it enforces a simple discrete version of Stokes' theorem for all chains of the grid [35]. Spex will proceed similarly, with the discrete exterior derivative acting naturally on $\bar{\Lambda}$ and $\widetilde{\Lambda}$.

Hodge Operator and Poincaré Duality If $\mathcal{M}$ is a d-dimensional manifold, then there is a natural isomorphism between k-forms Λ^k and $(d-k)$-forms Λ^{d-k}, called *Poincaré duality*. The duality map between the two spaces is through the Hodge star operators $\star^k$ and $\star^{d-k}$, acting on k- and $(d-k)$-forms respectively:

$$\star^k : \quad \Lambda^k \to \Lambda^{d-k}, \qquad\qquad \star^{d-k} : \quad \Lambda^{d-k} \to \Lambda^k.$$

In the discrete settings, this duality calls for the construction of a dual grid: the discrete star operators will then provide an isomorphism between primal ($\bar{\Lambda}$) and dual cochains ($\widetilde{\Lambda}$) since the dimension of the space of primal (resp., dual) k-forms matches the dimension of the space of dual (resp., primal) $(d-k)$-form [35].

High-order Operators Previously, the Hodge star has been evaluated using zeroth or first order basis functions, which while sparse and efficient, have poor convergence properties that do not permit its use in expressions involving combinations of multiple operators such as the composite Poisson operator $\star\mathbf{d}\star\mathbf{d} + \mathbf{d}\star\mathbf{d}\star$. For lower order $\star$, the result of $\star\mathbf{d}\star f$ is not meaningful since the error of $\star f \sim \mathcal{O}(h)$, where h is the resolution of the discrete mesh, would become $\mathbf{d}\star f \sim \mathcal{O}(1)$ after differentiation. Solving this issue can be achieved through a weak formulation; but with higher order methods, where $\star f \sim \mathcal{O}(h^k)$, taking the derivative returns directly a meaningful result $\mathbf{d}\star f \sim \mathcal{O}(h^{k-1})$ for sufficiently high grid resolution, and therefore the Poisson operator, or even higher-order operators, can be computed directly.

Additionally, since multiplication doubles the spectral band, in order to properly resolve the result, the multiplication has to be performed on a refined grid with twice as many points along each dimension.

Orthonormal Frames The metric will be stored in this representation per node and will be used to locally construct an orthonormal frame, in which the $\star$, $\flat$ and $\sharp$ operators will have will be easy to compute.

In Riemannian geometry and manifold theory, orthonormal frames are useful for studying the structure of a differentiable manifold equipped with a metric g. If $\mathcal{M}$ is a manifold and g is its metric, then an orthonormal frame at a point m of $\mathcal{M}$ is an ordered basis of elements of the tangent bundle $T_m\mathcal{M}$ consisting of vectors which are orthonormal with respect to the bilinear form g_m. The metric provides a mapping to a basis set of the dual tangent space $T_m^*\mathcal{M}$ given by the flat operator. To make that clear, consider a vector $\mathbf{v} \in T_m\mathcal{M}$; then its corresponding one-form is $\mathbf{v}^\flat(\cdot) = g_m(\mathbf{v}, \cdot)$.

The components of forms or vectors are always expressed relative to a basis frame. A vector frame $\mathcal{B}$ is an ordered set of vectors $\{\mathbf{e}_i\}$ for each point of the manifold, whereas a co-vector frame $\mathcal{B}^*$ is an ordered set of co-vectors $\{\mathbf{e}^i\}$. Formally $\mathbf{e}^i$ is an assignment of a vector to each point, i.e. $\mathbf{e}^i : \mathcal{M} \to T\mathcal{M}$, whereas $\mathbf{e}_i$ is the same on the cotangent bundle $\mathbf{e}_i : \mathcal{M} \to T^*\mathcal{M}$.

The reference coordinate basis for a rectilinear d-dimensional domain is

$$\mathcal{B} = \left\{\frac{\partial}{\partial x^1}, \frac{\partial}{\partial x^2}, \dots \frac{\partial}{\partial x^d}\right\}, \qquad \mathcal{B}^* = \left\{dx^1, dx^2, \dots dx^d\right\},$$

where the coordinates $\{x^1, x^2, \ldots, x^d\}$ span the manifold $\mathcal{M}$. This basis set is **holonomic** because for any differentiable manifold the set of basis vectors $\{\mathbf{e}_i\}$ are integrable and they are such that there exists a coordinate system $\{x^i\}$ for which

$$\mathbf{e}_i = \frac{\partial}{\partial x^i}, \qquad \mathbf{e}^i = dx^i, \tag{4.1}$$

The tangent and cotangent basis sets satisfy the contraction rule

$$\mathbf{e}_i \lrcorner \mathbf{e}^j = \delta_i^j.$$

The contraction pairs forms with vectors. If we want to pair vectors with vectors or forms with forms, we need the metric. The coefficients of the metric are expressible in terms of the basis elements as follows:

$$g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j, \qquad g^{ij} = \mathbf{e}^i \cdot \mathbf{e}^j,$$

and the metric is related to its inverse by

$$g_{ik}\, g^{kj} = \delta_i^j, \qquad g^{ik}\, g_{kj} = \delta_j^i.$$

Unless the metric is flat everywhere, this basis set will not be orthonormal in the general case, i.e. $\mathbf{e}_i$ will not be perpendicular to $\mathbf{e}_j$.

The metric establishes a duality between $\mathbf{e}_i$ and $\mathbf{e}^i$ by lowering and raising their indices:

$$\mathbf{e}_i = g_{ij}\mathbf{e}^j, \qquad \mathbf{e}^i = g^{ij}\mathbf{e}_j.$$

The reason orthonormal frames are useful is because it is easy to express the Hodge star and the inner product relative to an orthonormal frame of reference. In such a frame, the Hodge star operator acts on a basis form of degree r as follows:

$$\star\left(\mathbf{e}^{i_1} \wedge \mathbf{e}^{i_2} \wedge \cdots \wedge \mathbf{e}^{i_r}\right) = \epsilon^{i_1 i_2 \ldots i_r}{}_{j_{r+1}\cdots j_m}\, \mathbf{e}^{j_{r+1}} \wedge \mathbf{e}^{j_{r+2}} \wedge \cdots \wedge \mathbf{e}^{j_d}, \tag{4.2}$$

and the inner product of basis forms of degree r is given by

$$\left(\mathbf{e}^{i_1} \wedge \mathbf{e}^{i_2} \wedge \cdots \wedge \mathbf{e}^{i_r}\right) \cdot \left(\mathbf{e}^{j_1} \wedge \mathbf{e}^{j_2} \wedge \cdots \wedge \mathbf{e}^{j_r}\right) = \delta^{i_1 j_1}\delta^{i_2 j_2} \ldots \delta^{i_r j_r} \tag{4.3}$$

assuming the basis indices are ordered $i_1 < i_2 < \cdots < i_r$ and $j_1 < j_2 < \cdots < j_r$. For example, in two dimensions, for a flat metric, the Hodge star and the inner product are completely defined by the equalities

$$\star\, 1 = \mathbf{e}^1 \wedge \mathbf{e}^2, \quad \star\mathbf{e}^1 = \mathbf{e}^2, \quad \star\mathbf{e}^2 = -\mathbf{e}^1, \quad \star\left(\mathbf{e}^1 \wedge \mathbf{e}^2\right) = 1, \tag{4.4}$$

$$1 \cdot 1 = 1, \quad \mathbf{e}^1 \cdot \mathbf{e}^1 = 1, \quad \mathbf{e}^1 \cdot \mathbf{e}^2 = 0, \quad \mathbf{e}^2 \cdot \mathbf{e}^2 = 1, \quad (\mathbf{e}^1 \wedge \mathbf{e}^2) \cdot (\mathbf{e}^1 \wedge \mathbf{e}^2) = 1. \quad (4.5)$$

In order to get to an orthonormal frame, vectors and forms must undergo a basis transformation. To avoid ambiguity, let us denote the orthonormal frame with a hat:

$$\hat{\mathcal{B}} = \{\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_d\}, \qquad \hat{\mathcal{B}}^* = \{\hat{\mathbf{e}}^1, \hat{\mathbf{e}}^2, \ldots, \hat{\mathbf{e}}^d\},$$

where $\hat{\mathbf{e}}_i \perp \hat{\mathbf{e}}_j$ for any pair of basis elements with respect to the relevant metric. The orthonormal frame is defined on each tangent space of the manifold and it may be **non-holonomic** as there does not necessarily exist an integrable global coordinate system that is linked to the basis vectors by Eq. (4.1).

The transformation between the two basis sets $\hat{\mathbf{e}}_i$ and $\mathbf{e}_i$ will be given by

$$\hat{\mathbf{e}}^i = J^i_j \mathbf{e}^j \qquad \hat{\mathbf{e}}_i = \mathbf{e}_j (J^{-1})^j_i.$$

Notice that vectors and forms transform in such a way that their contraction remains invariant:

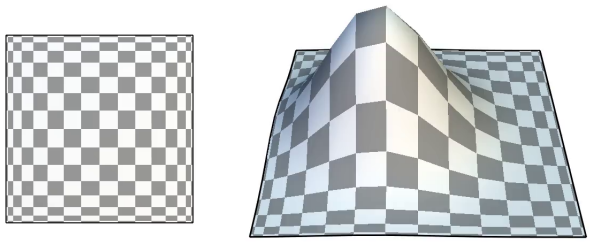
$$\hat{\mathbf{e}}_i \lrcorner \hat{\mathbf{e}}^j = \mathbf{e}_i \lrcorner \mathbf{e}^j = \delta_{ij}.$$

Namely, if vectors transform as $\mathbf{v} \mapsto \mathbf{J}\mathbf{v}$, then one-forms must transform as $\alpha \mapsto \alpha \mathbf{J}^{-1}$, where the contraction corresponds to the matrix product of a row with a column vector $\mathbf{v} \lrcorner \alpha = \alpha \mathbf{v}$. Section 4.A provides a detailed overview of how different objects transform under a coordinate change. The results relevant to the two-dimensional case are summarized in Table 4.3. In matrix notation, vectors are treated as a column matrix with one single column and differential forms are treated as a row matrix with a single row.

Name	**Element**	**Transformation**
vector	$X \in \mathfrak{X}$	$X \mapsto \mathbf{J}X$
0-form	$f \in \Lambda^0$	$f \mapsto f$
1-form	$\alpha \in \Lambda^1$	$\alpha \mapsto \alpha \mathbf{J}^{-1}$
2-form	$\omega \in \Lambda^2$	$\omega \mapsto \omega \det(\mathbf{J})^{-1}$

Table 4.3: List of transformation for vectors and forms under a basis change in 2D.

Induced Metric The reference computational grid will be denoted by $\mathcal{D}$, which for the periodic case will be $[0, 2\pi)^2$ and for the bounded cases will be $[-1, 1]^2$. The reference domain $\mathcal{D}$ will be mapped to the physical domain $\mathcal{M}$ by the map $\varphi : \mathcal{D} \to \mathcal{M}$.



$\mathcal{D}$ $\rightarrow$ $\mathcal{M}$

Typically, the manifold $\mathcal{M}$ will be embedded in either two- or three-dimensional Eucledian space, i.e. $\mathcal{M} \subset \mathbb{R}^{\{2,3\}}$, and it will have an associated metric $g_{\mathcal{M}}$ that will be the standard Eucledian metric for the embedding space. The map φ induces a metric $g_{\mathcal{D}}$ on the reference domain $\mathcal{D}$ given by the pullback of the metric from the physical manifold $\mathcal{M}$:

$$g_{\mathcal{D}} = \varphi^*(g_{\mathcal{M}}).$$

The components are given by its application to the basis elements

$$g_{ij} = g_{\mathcal{D}}(\mathbf{e}_i, \mathbf{e}_j)$$

Given this induced metric, we want to compute a transformation J^i_j such that $\hat{\mathbf{e}}_i = \mathbf{e}_j (J^{-1})^j_i$ is an orthonormal frame with respect to $g_{\mathcal{D}}$:

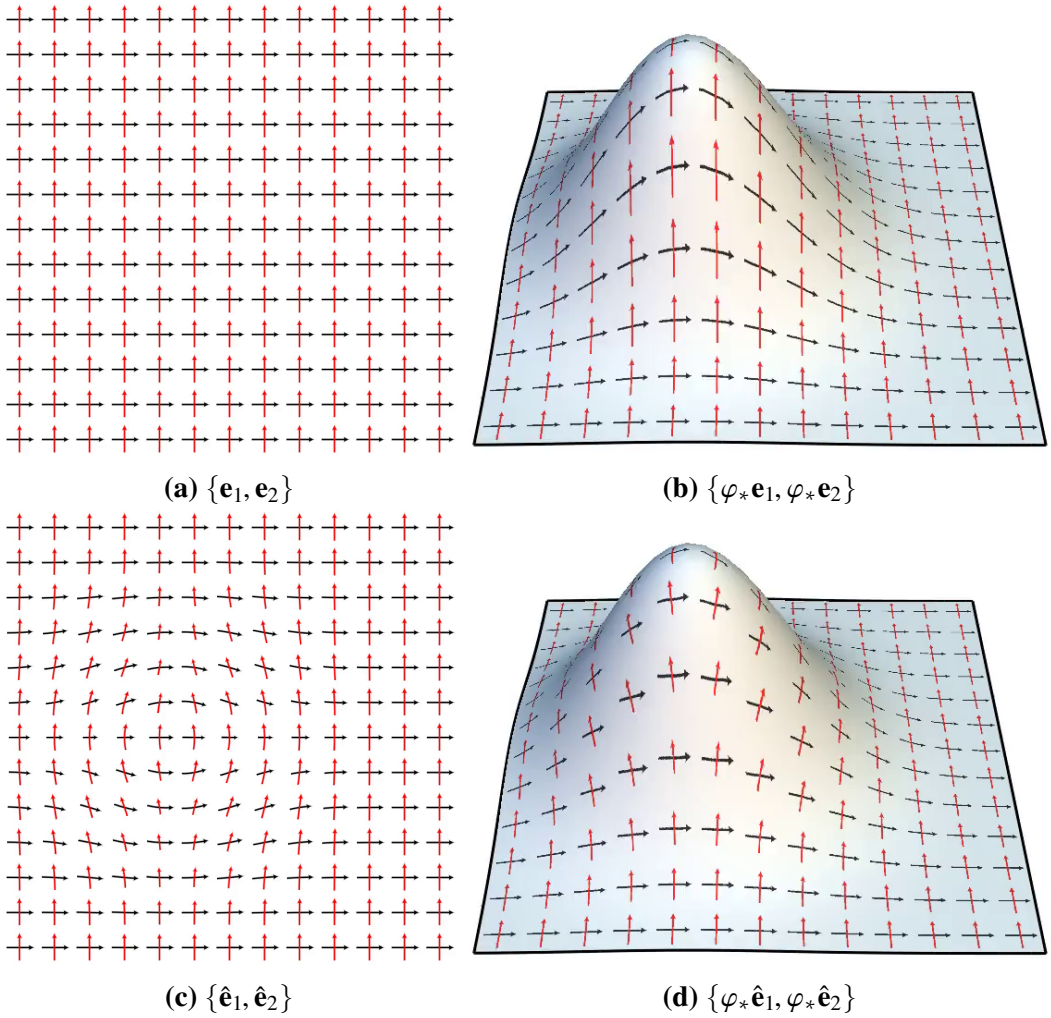
$$\hat{g}_{ij} = g_{\mathcal{D}}(\hat{\mathbf{e}}_i, \hat{\mathbf{e}}_j) = \delta_{ij}.$$

Because the expression above is invariant under rotations,

$$g_{\mathcal{D}}(\hat{\mathbf{e}}_i, \hat{\mathbf{e}}_j) = g_{\mathcal{D}}(\mathbf{R}\hat{\mathbf{e}}_i, \mathbf{R}\hat{\mathbf{e}}_j),$$

there are many choices for the gauge. The matrix square root is one such choice. Let us denote the elements of the transformation J^i_j using the matrix $\mathbf{J}$, and the elements of the tensor g_{ij} using the matrix $\mathbf{g}$. For example, in two dimensions ,

$$\mathbf{J} = \begin{bmatrix} J^1_1 & J^1_2 \\ J^2_1 & J^2_2 \end{bmatrix}, \qquad \mathbf{g} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}.$$



(a) $\{\mathbf{e}_1, \mathbf{e}_2\}$ **(b)** $\{\varphi_* \mathbf{e}_1, \varphi_* \mathbf{e}_2\}$

(c) $\{\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2\}$ **(d)** $\{\varphi_* \hat{\mathbf{e}}_1, \varphi_* \hat{\mathbf{e}}_2\}$

Figure 4.1: Computing an orthonormal frame for a surface embedded in 3D Euclidean space. The black arrows represent $\mathbf{e}_1$, and the red arrows represent $\mathbf{e}_2$. (a) and (b) illustrate the coordinate basis $\mathcal{B}$ in $\mathcal{D}$ and $\mathcal{M}$ respectively, and (c) and (d) illustrate the orthonormal basis $\hat{\mathcal{B}}$.

For $\mathbf{J}$ to map to an orthonormal frame, it must be related to $\mathbf{g}$ by

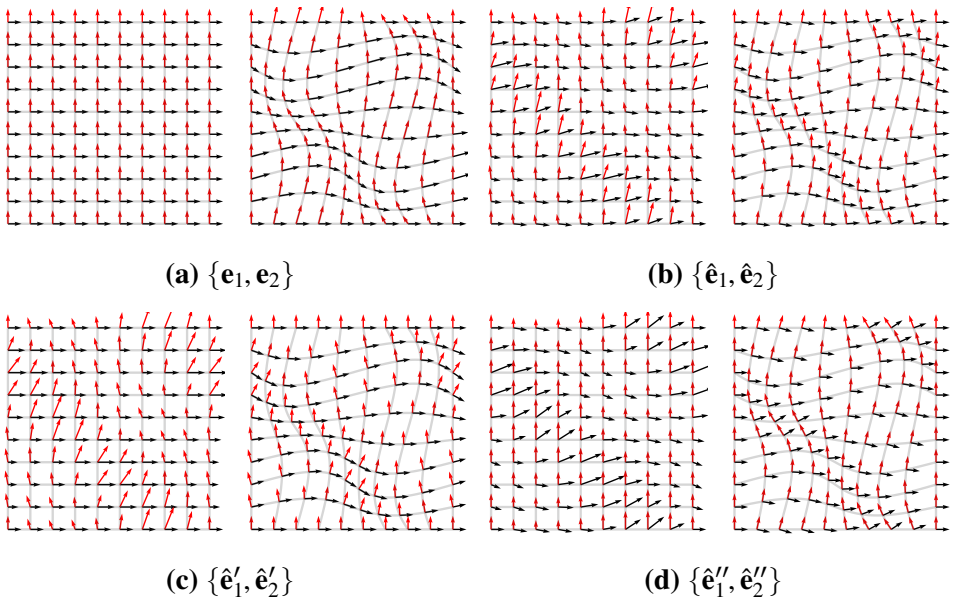
$$\mathbf{g} = \mathbf{J}^T \mathbf{J}.$$

To see that this does indeed result in an orthonormal frame:

$$\begin{aligned}
\hat{g}_{ij} &= g_{\mathcal{D}}(\hat{\mathbf{e}}_i, \hat{\mathbf{e}}_j) \\
&= g_{\mathcal{D}}(\mathbf{e}_m, \mathbf{e}_n)(J^{-1})^m_i (J^{-1})^n_j \\
&= g_{mn}(J^{-1})^m_i (J^{-1})^n_j \\
&= (\mathbf{J}^{-T} \mathbf{g} \mathbf{J}^{-1})_{ij} \\
&= \delta_{ij}
\end{aligned}$$

as required. One possibility for a $\mathbf{J}$ that satisfies the above requirement is given by the matrix square root, which results in a symmetric $\mathbf{J}$ since $\mathbf{g}$ is symmetric:

$$\mathbf{J} = \sqrt{\mathbf{g}}.$$



(a) $\{\mathbf{e}_1, \mathbf{e}_2\}$ **(b)** $\{\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2\}$

(c) $\{\hat{\mathbf{e}}'_1, \hat{\mathbf{e}}'_2\}$ **(d)** $\{\hat{\mathbf{e}}''_1, \hat{\mathbf{e}}''_2\}$

Figure 4.2: Computing an orthonormal frame for a surface embedded in 2D Euclidean space. (a) illustrates the standard reference basis and its push forward, (b) - an orthonormal basis by taking the square root of the metric, (c) - an orthonormal basis using the Gran-Schmidt process starting with $\mathbf{e}_1$, and (d) - starting with $\mathbf{e}_2$.

Figure 4.1 illustrates the results of constructing of an orthonormal frame using the above square root method for a manifold embedded in $\mathbb{R}^3$. Note that the orthonormal frame is by no means unique and there are infinitely many equally valid choices for constructing $\mathbf{J}$ from $\mathbf{g}$. For example, given any set of linearly independent vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$, instead of taking the square root as above, one could use the Gram-Schmidt process to compute an orthonormal set $\{\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_d\}$. Figure 4.2 compares the matrix square root and the Gram-Schmidt orthogonalization.

4.3 Spatial Discretization and Associated Basis Functions

We begin our exposition by discussing how the computational domain $\mathcal{D}$ can be discretized into a cell complex, and how the basis functions (each associated with a cell of the computational grid) are expressed accordingly. We will consider two types of domains - periodic and bounded.

One-dimensional Cell Complexes

For a one-dimensional domain, its discretization into a computational cell complex consists of only vertices (also called nodes) and edges. We will use N to refer to the grid size, defined by the number of primal edges. As noted earlier, our discrete calculus setup exploits Poincaré duality by using both a primal and dual grid between which the Hodge stars can act as a duality operator that converts 0-forms to 1-forms and vice versa. Consequently, a one-dimensional grid has four different cells: primal vertices ($\bar{V}$), primal edges ($\bar{E}$), dual vertices ($\widetilde{V}$), and dual edges ($\widetilde{E}$).

Periodic Domain When the 1D domain is periodic, we consider $\mathcal{D} = [0, 2\pi)$ without loss of generality. The positions of primal and dual vertices are set to

$$\bar{V}_N = \left\{ \bar{\theta}_n = \frac{2\pi}{N} n, \quad n = 0, \ldots, N-1 \right\},$$
$$\widetilde{V}_N = \left\{ \widetilde{\theta}_n = \frac{2\pi}{N} \left(n + \tfrac{1}{2}\right), \quad n = 0, \ldots, N-1 \right\},$$

offering a regular sampling of the domain. Primal (resp., dual) edges are between consecutive primal (resp., dual) vertices. The resulting cell complex will be referred to as the periodic cell complex. Its cells will be given by

$$\bar{\sigma}^0_n = [\bar{\theta}_n], \qquad \widetilde{\sigma}^0_n = [\widetilde{\theta}_n], \qquad \bar{\sigma}^1_n = [\bar{\theta}_n, \bar{\theta}_{n+1}], \qquad \widetilde{\sigma}^1_n = [\widetilde{\theta}_{n-1}, \widetilde{\theta}_n].$$

Bounded Domain Without loss of generality, we consider $\mathcal{D} = [-1, 1]$ in the bounded case. Positions of the primal and dual vertices are set to

$$\bar{V}_N = \left\{ \bar{x}_n = -\cos\left(\frac{n+1}{N}\pi\right), \quad n = 0, \ldots, N-2 \right\},$$
$$\widetilde{V}_N = \left\{ \widetilde{x}_n = -\cos\left(\frac{n+1/2}{N}\pi\right), \quad n = 0, \ldots, N-1 \right\}.$$

Just like in the periodic case, primal (resp., dual) edges are between consecutive primal (resp., dual) vertices; but in this bounded domain case, two *boundary* vertices are also added at each end of the domain $\mathcal{D}$ in order to ensure the closure of the boundary operator ∂. The resulting cell complex will be called the **bounded cell complex**. The cells, as in the periodic case, are given by

$$\bar{\sigma}^0_n = [\bar{x}_n], \qquad \widetilde{\sigma}^0_n = [\widetilde{x}_n], \qquad \bar{\sigma}^1_n = [\bar{x}_n, \bar{x}_{n+1}], \qquad \widetilde{\sigma}^1_n = [\widetilde{x}_{n-1}, \widetilde{x}_n].$$

Figure 4.3 illustrates the relative topological arrangement of its elements.

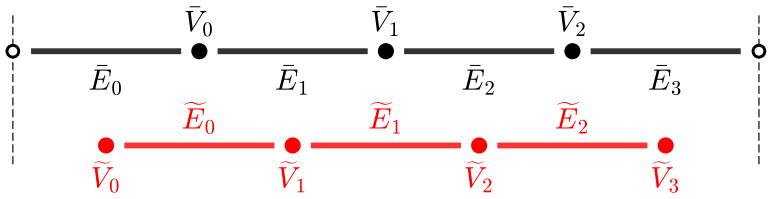


Figure 4.3: Schematic representation of the computational grid in the one-dimensional bounded case. The simplices $\bar{V}$, $\bar{E}$, $\widetilde{V}$, and $\widetilde{E}$ are used to store the discrete forms in $\bar{\Lambda}^0$, $\bar{\Lambda}^1$, $\widetilde{\Lambda}^0$, and $\widetilde{\Lambda}^1$ respectively. Vertices are drawn equally spaced for simplicity.

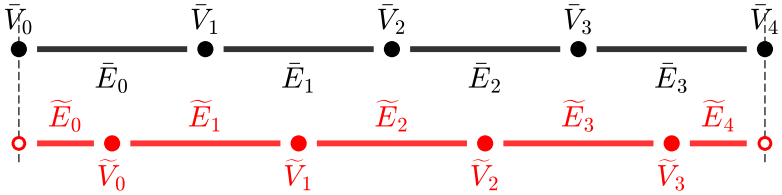


Figure 4.4: Cell complex with clipped edges at the boundary (old setup).

Comparison Note that the cell complex shown in Figure 4.3 is different from the one used in the previous chapter [62]. In the new grid there are no longer any *clipped* edges. All the edges that form the cell complex are *whole*, whereas before they were clipped at the boundary (see Figure 4.4). While this difference is rather insignificant from a geometric point of view, the use of half-edges complicates the analytical expressions for histopolation as well as the implementation of exterior calculus operators. We thus default to using the simpler arrangement of edges in Figure 4.3: the new arrangement for the dual edges no longer fills the entire domain, and only the primal edges fill up the domain completely. Boundary vertices are considered separately, as they will allow us to prescribe boundary conditions.

1D Grid	$\lvert\bar{V}_N\rvert$	$\lvert\bar{E}_N\rvert$	$\lvert\widetilde{V}_N\rvert$	$\lvert\widetilde{E}_N\rvert$	$\lvert\mathring{V}_N\rvert$
Periodic	N	N	N	N	$2N$
Bounded	$N-1$	N	N	$N-1$	$2N-1$
Bounded (old)	$N+1$	N	N	$N+1$	$2N+1$

Table 4.4: Number of primal and dual elements in 1D grids as a function of N.

Component vertices For any cell complex, the component vertices ($\mathring{V}$) are then defined as the union of the primal and dual vertices:

$$\mathring{V} \equiv \bar{V} \cup \widetilde{V}.$$

The total number of vertices and edges for the one-dimensional grid for a given discretization level N is shown in Table 4.4. Notice that our construction implies

$$\mathring{V}_N = \bar{V}_{2N},$$

that is, the component vertices are identified to the primal vertices of the twice-finer grid $\bar{V}_{2N}$, a property that we will leverage to seamlessly upsample or downsample a field later on.

One-dimensional Basis Functions

Having defined the cell complexes, we now proceed to describe a general approach for deriving the basis functions associated with them. We need to define the following set of basis functions for the cell complexes described in the preceding section

$$\bar{\phi}^0_n \qquad \widetilde{\phi}^0_n \qquad \bar{\phi}^1_n \qquad \widetilde{\phi}^1_n$$

The topic of constructing cardinal basis functions associated with the vertices of a computational grid has been well studied in the literature [11, 42] and can be used to interpolate vertex-based functions (0-forms). Additionally, we can construct histopolation basis functions that can be used to do the same for edge-based functions (1-forms).

	periodic	bounded
position variable	θ	x
vertex positions	θ_n	x_n
grid function	$\Phi_N(\theta)$	$P_N(x)$
vertex functions	$\mathcal{V}[\Phi_N, n](\theta)$	$\mathcal{V}[P_N, n](x)$
edge functions	$\mathcal{E}[\Phi_N, n](\theta)$	$\mathcal{E}[P_N, n](x)$

Table 4.5: Correspondence between notation for periodic and bounded domains.

Table 4.5 illustrates the correspondence between the notation for periodic and bounded domains. The grid function is a function whose zeroes correspond to the vertices of the grid, hence its name. Its shape defines the grid. For the periodic case, the grid function will be denoted by $\Phi_N(\theta)$ and in the bounded case it will be $P_N(x)$. From the grid functions we can generate the cardinal vertex and edge functions, which we will denote using $\mathcal{V}$ and $\mathcal{E}$ respectively.

Vertex Interpolation

Bounded For bounded domains we use polynomial interpolation. The Lagrange polynomials, which correspond to the vertex basis functions, are given by

$$p_n(x) = \prod_{m \neq n} \frac{x - x_m}{x_n - x_m}.$$

The polynomial $p_n(x)$ assumes the value one only at the vertex x_n, and is zero at every other vertex x_m ($m \neq n$). If x_n are the roots of the polynomial P_N, i.e. $P_N(x_n) = 0$ for $\forall n = 0, \ldots, N-1$, then the Lagrange polynomials can be written in the equivalent form

$$p_n(x) = \frac{1}{\mathcal{N}} \frac{P_N(x)}{x - x_n},$$

where $\mathcal{N}$ is a normalization factor chosen to enforce the condition

$$p_n(x_n) = 1.$$

By L'Hospital's rule, we can compute the limit as

$$\lim_{x \to x_n} p_n(x) = \frac{1}{\mathcal{N}} P'_N(x_n),$$

where the prime denotes differentiation. Thus the appropriate value for the normalization constant is $\mathcal{N} = P'_N(x_n)$ and the formula for the cardinal functions becomes

$$\mathcal{V}[P_N, n](x) \equiv p_n(x) = \frac{1}{P'_N(x_n)} \frac{P_N(x)}{x - x_n}.$$

We will say that these are the vertex functions associated with the roots of P_N, and we will write them as $\mathcal{V}[P_N, n](x)$ in order to underline the fact that the vertex functions can be computed solely from P_N and n. Note that $\mathcal{V}[P_N, n]$ as a function of P_N is homogeneous in P_N, i.e. for any nonzero constant a it holds true that $\mathcal{V}[aP_N, n] = \mathcal{V}[P_N, n]$.

Periodic A similar formula holds true for trigonometric interpolation over periodic domains.

$$\phi_n(\theta) = c_N(\theta) \prod_{m \neq n} \frac{\sin \frac{1}{2}(\theta - \theta_m)}{\sin \frac{1}{2}(\theta_n - \theta_m)},$$

where c_N is a correction factor for when N is odd/even.[1] If θ_n are the roots of the trigonometric function Φ_N, i.e. $\Phi_N(\theta_n) = 0$, then the periodic cardinal functions

[1] For equidistant points c_N is given by $c_N(\theta) = \begin{cases} 1 & \text{if N odd} \\ \cos\frac{\theta}{2} & \text{if N even} \end{cases} = \frac{1-(-1)^N}{2} + \frac{1+(-1)^N}{2}\cos\frac{\theta}{2} = \cos^2\frac{\theta}{4} - (-1)^N \sin^2\frac{\theta}{4}$.

can be written as

$$\mathcal{V}[\Phi_N, n](\theta) \equiv \phi_n(\theta) = \frac{1}{\Phi'_N(\theta_n)} \frac{\Phi_N(\theta)}{2 \sin \frac{1}{2}(\theta - \theta_n)},$$

which emphasizes the fact that the above cardinal functions can be solely derived from Φ_N and n. It can be easily checked by applying L'Hospital's rule that the above expression does indeed constitute a cardinal function by verifying that $\phi_n(\theta_m) = \delta_{nm}$.

Edge Histopolation

Bounded In order to compute the edge functions, which are needed for histopolation, we follow the approach taken by Gerritsma [28]. Namely,

$$\mathcal{E}[P_N, n](x) = -\sum_{m=0}^{n} \mathcal{V}[P_N, m]'(x).$$

The above expression corresponds to taking the derivative of a "step function". Only the edge across which the step shifts from 0 to 1 will have a non-zero value for the integral, whereas for the rest of the edges that value will be equal to zero.

The derivative is explicitly given by

$$\mathcal{V}[P_N, n]'(x) = \frac{(x - x_n)P'_N(x) - P_N(x)}{(x - x_n)^2 P'_N(x_n)}.$$

Sometimes we will need to evaluate the above expression at x_n, and its limit there is

$$\mathcal{V}[P_N, n]'(x_n) = \frac{P''_N(x_n)}{2P'_N(x_n)}.$$

Explicit expressions for the limits at the boundaries for Chebyshev polynomials are given in 4.B.

Periodic To go from vertex functions to edge functions, we first note that the convolution of the edge function with a box function that spans the corresponding edge must be equal to the vertex function [62]. Given a function $f(\theta)$ we define the following functional transformation on it:

$$\mathcal{T}[f](\theta) = \int_{\theta - \frac{h}{2}}^{\theta + \frac{h}{2}} f(\theta')\, d\theta' \quad \text{with } h = \frac{2\pi}{N}.$$

The cardinal edge functions then can be expressed in terms of the inverse of the above transformation:

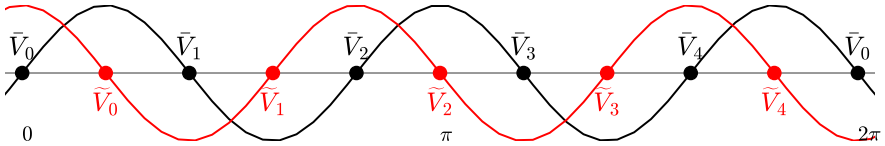
$$\mathcal{E}[\Phi_N, n](\theta) = \mathcal{T}^{-1}\big[\mathcal{V}[\Phi_N, n]\big](\theta).$$

Basis Functions Explicitly

Periodic The primal and dual vertices are the roots of the trigonometric functions below:

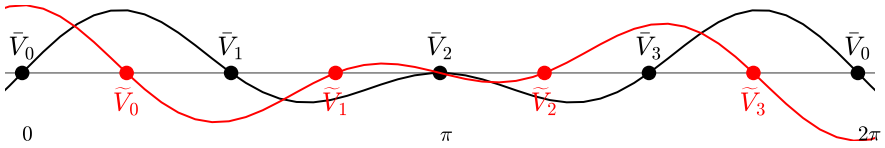
N odd

$$\bar{\Phi}_N(\theta) = \sin\left(\frac{1}{2}N\theta\right) \qquad \widetilde{\Phi}_N(\theta) = \cos\left(\frac{1}{2}N\theta\right) \qquad \text{if N odd.}$$



N even

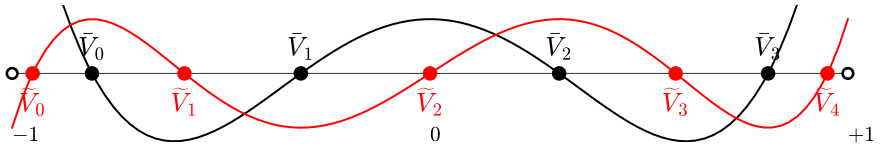
$$\bar{\Phi}_N(\theta) = \sin\left(\frac{1}{2}N\theta\right)\cos\frac{\theta}{2} \qquad \widetilde{\Phi}_N(\theta) = \cos\left(\frac{1}{2}N\theta\right)\cos\frac{\theta}{2} \qquad \text{if N even.}$$



Then the basis functions in the periodic case become

$$\bar{\phi}^0_n(\theta) = \mathcal{V}[\bar{\Phi}_N, n](\theta), \qquad \widetilde{\phi}^0_n(\theta) = \mathcal{V}[\widetilde{\Phi}_N, n](\theta),$$
$$\bar{\phi}^1_n(\theta) = \mathcal{E}[\widetilde{\Phi}_N, n](\theta), \qquad \widetilde{\phi}^1_n(\theta) = \mathcal{E}[\bar{\Phi}_N, n](\theta).$$

Bounded One distinguishes between Chebyshev polynomials of the first kind which are denoted by T_n and Chebyshev polynomials of the second kind which are denoted U_n. The primal vertices for the Chebyshev grid are the roots of the polynomials of second kind, whereas the dual vertices are the roots of the first kind. Hence, the primal and dual grid functions can be written as



For the primal edges, we need to clamp the polynomial at the endpoints by multiplying by $1 - x^2$:

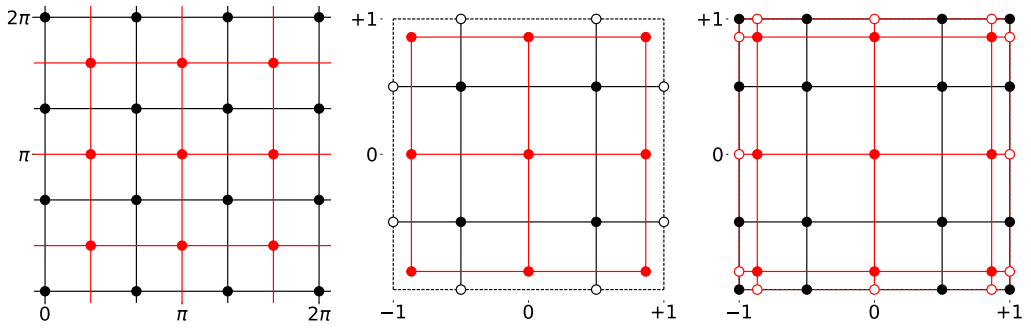
$$\text{clamp}[P_N](x) \equiv (1 - x^2)P_N(x).$$

Then basis functions are

$$\bar{\phi}_n^0(x) = \mathcal{V}[U_{N-1}, n](x), \qquad \tilde{\phi}_n^0(x) = \mathcal{V}[T_N, n](x),$$
$$\bar{\phi}_n^1(x) = \mathcal{E}[\text{clamp}[U_{N+1}], n](x), \qquad \tilde{\phi}_n^1(x) = \mathcal{E}[T_N, n](x).$$

Higher-dimension Grids and Associated Basis Functions

The case of grids in higher dimensions are simply handled via Cartesian product of the 1D grids we just described: Figure 4.5) shows the case of 2D grids for the periodic domain $[0, 2\pi)^2$ and the bounded domain $[-1, 1]^2$. Note that the component vertices are also obtained by Cartesian product of the 1D component vertices: they are no longer just the union of primal and dual vertices, but also contain intersections of primal and dual elements (see example in Figure 4.9). Finally, the associated basis functions to each of the cells of these grids in nD are simply Cartesian products of the one-dimensional ones.



(a) Periodic **(b)** Bounded **(c)** Bounded (old)

Figure 4.5: Computational grids of domain $\mathcal{D}$ for the periodic and non-periodic cases. Hollow nodes denote boundary vertices, whereas dashed edges denote boundary edges.

Note that while the use of vertices (V) and edges (E) were particularly convenient for the 1D case, we will now refer to arbitrary d-dimensional cells as $\bar{\sigma}$ for primal

ones and $\tilde{\sigma}$ for dual ones, with an upper index indicating its dimension, and a lower index indicating its label; i.e., in 1D, we will now use

$$\bar{V}_k \equiv \bar{\sigma}^0_k, \qquad \tilde{V}_k \equiv \tilde{\sigma}^0_k, \qquad \bar{E}_k \equiv \bar{\sigma}^1_k, \qquad \tilde{E}_k \equiv \tilde{\sigma}^1_k,$$

and higher-dimensional cells will be similarly denoted with higher upper indices. We will denote by $\bar{\phi}^p_i$ (resp., $\tilde{\phi}^p_i$) the p-form primal (resp., dual) basis function of the i-th p-dimensional element $\bar{\sigma}^p_i$ (resp., $\tilde{\sigma}^p_i$); by construction, they satisfy:

$$\int_{\bar{\sigma}^p_j} \bar{\phi}^p_i = \delta_{ij} \quad \text{and} \quad \int_{\tilde{\sigma}^p_j} \tilde{\phi}^p_i = \delta_{ij}.$$

We also use a systematic enumeration of cells in our code, an example of which is presented in Figure 4.6. In the remainder of this paper, we will use $\Sigma^0 \equiv \{\sigma^0_k\}_k$ to refer to the set of all nodes, Σ^1 for all the edges, etc.

4.4 Discretization of Fields

Equipped with the periodic and bounded computational grids, we can now discuss how the main fields (vector fields and k-forms) on a smooth manifold $\mathcal{M}$ are stored on this grid. In particular, we will use the property that integration of differential forms is preserved by the pullback of a smooth mapping function between two manifolds to motivate a cochain representation of arbitrary forms. A pointwise coordinate representation of forms, vector fields, and metric will also be leveraged to offer flexibility in the type of computations that one can achieve with Spex.

We introduce spaces of discrete forms and we will be denote them by adding a superscript on top of the continuous symbol. Primal and dual cochain forms will be denoted by $\bar{\Lambda}$ and $\widetilde{\Lambda}$, respectively.

The space of pointwise-component forms will be denoted by $\mathring{\Lambda}$.

Discrete Forms As Cochains

As discussed in Section 4.2, cochains are a very powerful discretization of differential forms. For an arbitrary p-form ω^p on the original manifold $\mathcal{M}$, its integrations over the submanifolds formed by the grid elements of order p of the domain $\mathcal{D}$ via the map ϕ provide a simple finite-dimensional approximation of the original form: these integrated form values of (primal or dual) cells of $\mathcal{D}$ can be histopolated using the (primal or dual) basis functions we described in the section above, in order to reconstruct a smooth approximation of the pull-back of ω on $\mathcal{D}$. For illustration purposes, we show the cells associated with our cochain representation of forms of

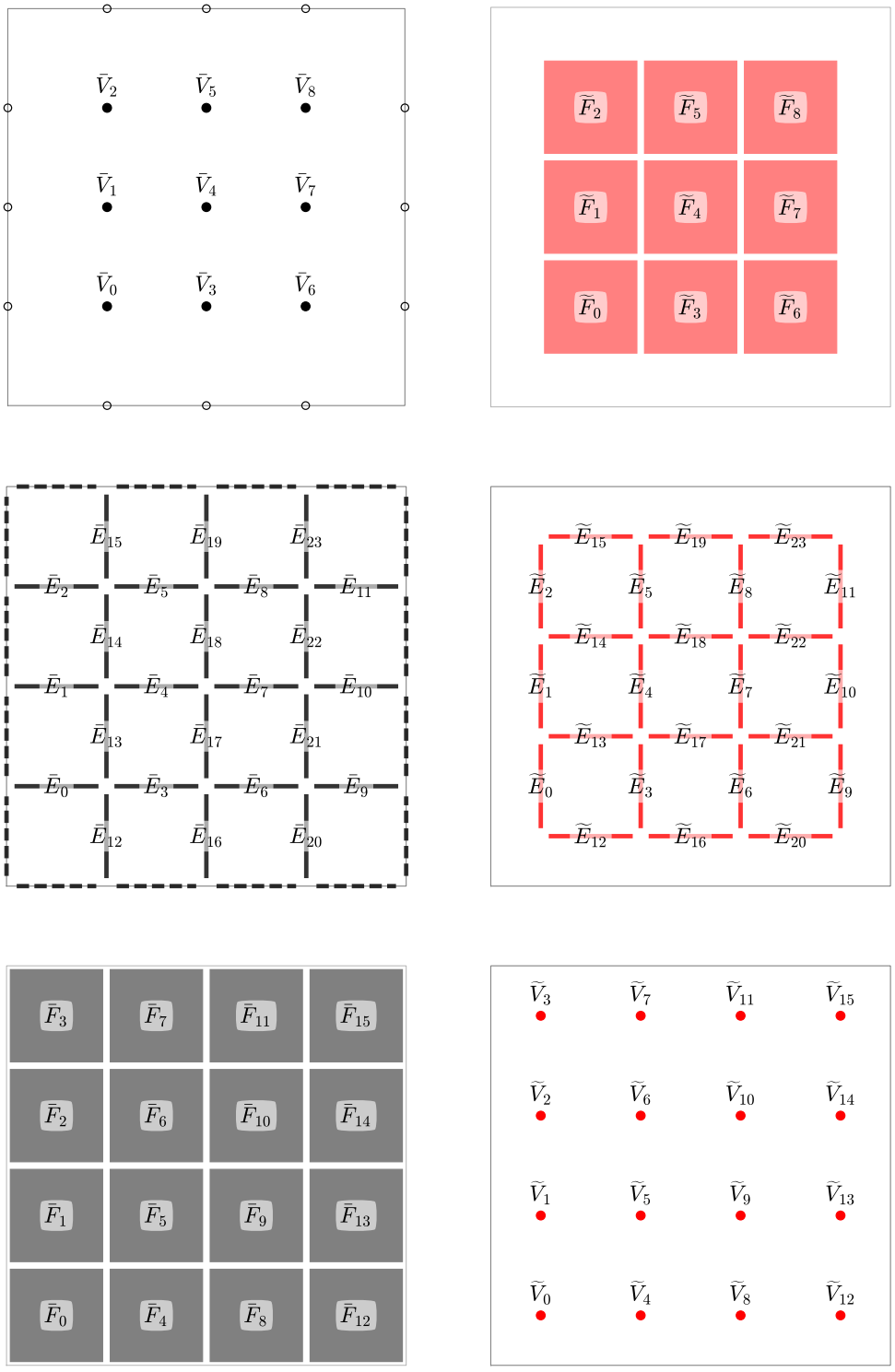


Figure 4.6: Enumeration of the cells of a 4×4 grid. Boundary vertices and edges are excluded from the figure.

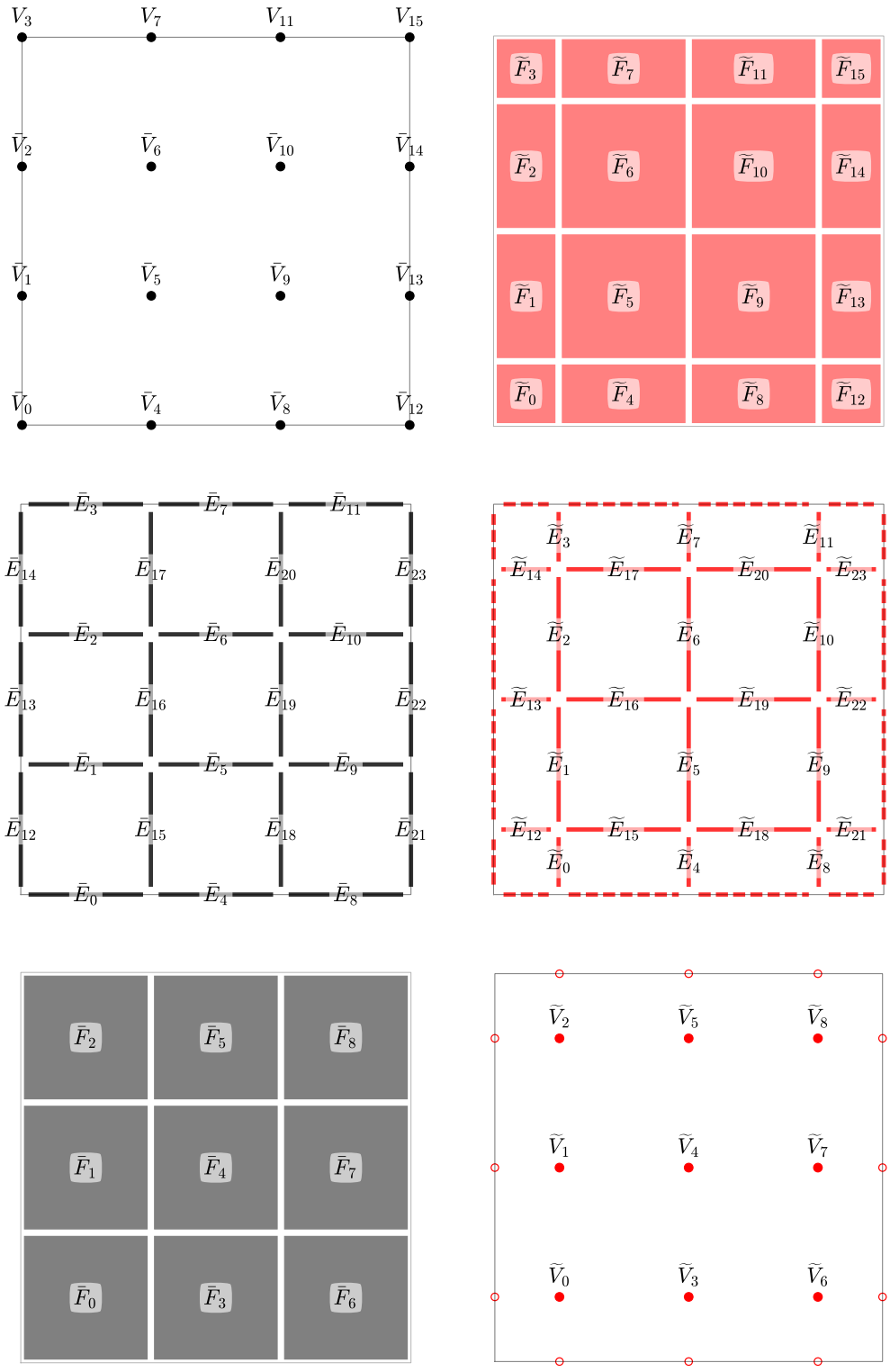
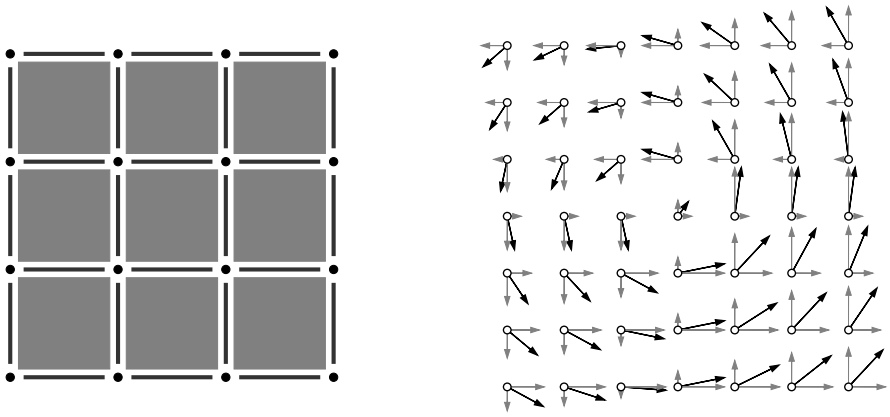


Figure 4.7: Enumeration of the cells of a 3×3 grid. This corresponds to the old setup with clipped edges at the boundary (e.g., the half-edges $\widetilde{E}_{12}$) visible in the dual grid.



(a) Cochain Representation.

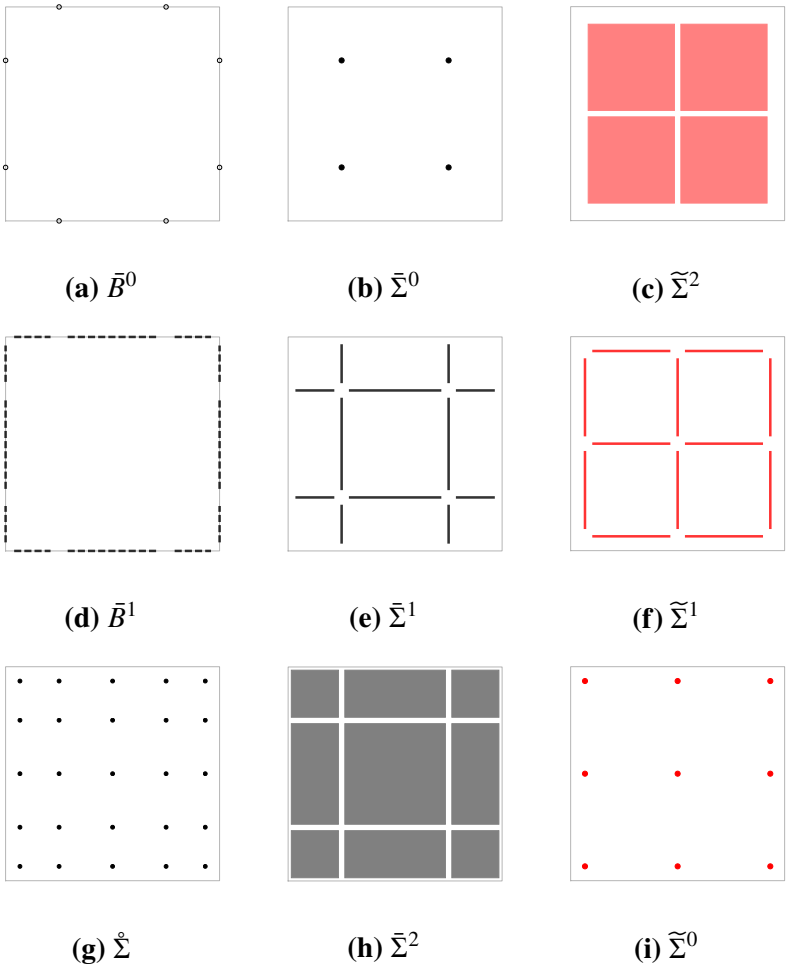
(b) Pointwise Component Representation.

Figure 4.8: Two discrete representations of forms.

various orders for a 2D grid on the first two rows of Figure 4.9. We now make these concepts of discretization and histopolation more formal by introducing cochain spaces, along with a reduction operator and a reconstruction operator.

Spaces of Primal and Dual Cochains In exterior calculus, differential forms of order p form a vector space Λ^p. As exploited in various efforts to construct a discrete equivalent of exterior calculus [35, 7, 3], the algebraic topology notion of cochain is a natural discretization of differential forms of degree p: just as differential forms are objects to be integrated over p-dimensional domains to return a scalar, cochains are fundamental objects that assign a number to each cell of dimension p of the grid. Formally, the space of cochains is the dual linear space to the space of chains, i.e., linear mappings from chains to real numbers. For instance, a primal p-cochain is a linear function defined on the primal p-chain space, which consists of linear combinations of p-dimensional primal mesh elements. As a linear function, a primal p-cochain can be determined by the value that it assigns to each p-dimensional primal mesh element.

In this work, we also use cochains as *one* discrete representation of differential forms. We will denote the space of primal p-cochains as $\bar{\Lambda}^p$, and the space of dual p-cochains as $\widetilde{\Lambda}^p$. Note that we may sometimes omit the superscript (p) for brevity, when the order of the cochain is obvious from context.



(a) $\bar{B}^0$ **(b)** $\bar{\Sigma}^0$ **(c)** $\widetilde{\Sigma}^2$

(d) $\bar{B}^1$ **(e)** $\bar{\Sigma}^1$ **(f)** $\widetilde{\Sigma}^1$

(g) $\mathring{\Sigma}$ **(h)** $\bar{\Sigma}^2$ **(i)** $\widetilde{\Sigma}^0$

Figure 4.9: Boundary elements are shown in the first column. Boundary vertices (a) are marked as hollow points, and boundary edges (d) as dashed line segments. The component vertices (g) $\mathring{\Sigma}$ are the points at which component forms are collocated regardless of their degree. The primal cell complex $\bar{\Sigma}^k$ of degree k is shown in the middle column. The dual cell complex $\widetilde{\Sigma}^k$ is shown in the last column. There is a correspondence between $\bar{\Sigma}^k$ and $\widetilde{\Sigma}^{n-k}$ given by the Poincaré duality operator $*$.

Reduction Operators We denote the linear map from a continuous form to its corresponding, finite-dimensional primal (resp., dual) cochain representation as the reduction operator $\mathcal{P}$ (resp., $\widetilde{\mathcal{P}}$). This reduction is achieved through integration over each of the grid elements σ^k (or equivalently, on their mapped submanifolds $\phi(\sigma^k)$ onto $\mathcal{M}$) of the same order k as the differential form; i.e.,

$$\bar{\mathcal{P}}: \quad \Lambda^k \to \bar{\Lambda}^k$$
$$\omega^k \to \bar{\omega}^k, \quad \text{with} \quad \bar{\omega}_i^k = \int_{\sigma_i^k} \omega^k,$$

and its dual version

$$\begin{aligned} \widetilde{\mathcal{P}}: \quad & \Lambda^k \to \widetilde{\Lambda}^k \\ & \omega^k \to \widetilde{\omega}^k, \quad \text{with} \quad \widetilde{\omega}_i^k = \int_{\widetilde{\sigma}_i^k} \omega^k. \end{aligned}$$

This reduction operator commutes with the exterior derivative (defined as the adjoint of the boundary operator on primal or dual chains) and creates an isomorphism between de Rham cohomology and singular cohomology, two particularly useful properties for computations [7, 21, 13]. Note that an obvious alternative is through L^2 projection onto the basis functions of forms.

Reconstruction Operators In order to go from primal or dual cochains to continuous forms, we simply apply interpolation/histopolation based on the basis functions defined in Section 4.3: with the value of each cell as the coefficient for the basis function associated to the cell, the resulting linear combination reconstructs a continuous form, i.e.,

$$\begin{aligned} \bar{\mathcal{R}}: \quad & \bar{\Lambda}^k \to \Lambda^k \\ & \bar{\omega}^k \to \omega^k, \quad \text{with } \omega^k \equiv \sum_i \bar{\omega}_i^k \bar{\phi}_i^k \end{aligned}$$

as well as

$$\begin{aligned} \tilde{\mathcal{R}}: \quad & \tilde{\Lambda}^k \to \Lambda^k \\ & \tilde{\omega}^k \to \omega^k, \quad \text{with } \omega^k \equiv \sum_i \tilde{\omega}_i^k \tilde{\phi}_i^k. \end{aligned}$$

For simplicity, we will omit the bar or the tilde when it can be trivially determined by the operand that the operator acts upon.

Mimetic Projection Maps The primal (resp., dual) mimetic projection maps $\pi_{\bar{h}}$ (resp., $\pi_{\tilde{h}}$) combine reduction and reconstruction [54] through

$$\pi_{\bar{h}} := \bar{\mathcal{R}}\bar{\mathcal{P}}, \qquad \pi_{\tilde{h}} := \tilde{\mathcal{R}}\tilde{\mathcal{P}}.$$

It can be readily verified that the above maps are projections since they are idempotent:

$$\pi_h \pi_h = \pi_h.$$

Any differential form originally inside the subspace spanned by the form basis functions can be discretized and reconstructed without any loss. Moreover, as the resolution of the meshes increases, the form subspaces become dense, and the projection onto these finite-dimensional subspaces converges to the identity in the continuous limit, i.e.,

$$\lim_{h \to 0} \|\omega - \pi_h \omega\| = 0.$$

Discrete Forms as Pointwise Components

While cochains are a convenient representation of integrals of forms (which will allow us, for instance, to enforce Stokes' theorem on the computational grid), we will also make use of a pointwise, component-based representation of forms. Our "pointwise component" representation of forms is defined as a set of discrete zero forms stored on the component (i.e., refined) grid $\mathring{\Sigma}$ (see Figure 4.9). Each zero form represents one of the coordinates of the form on a chosen basis; for instance, a 1-form on a 2-manifold parmaterized through two coordinates u and v can be represented as $f(u,v)du + g(u,v)dv$, so its point component representation will be given as the values of f and g sampled at the vertices of the component grid. More formally, the space of discrete "pointwise component" k-forms will be denoted by $\mathring{\Lambda}^k$, with a number of components in d dimensions given by $\binom{d}{k} = \frac{d!}{k!(d-k)!}$:

$$\mathring{\Lambda}^k_N \cong \underbrace{\bar{\Lambda}^0_{2N} \times \cdots \times \bar{\Lambda}^0_{2N}}_{\binom{d}{k}}.$$

For example, on a two-dimensional manifold, a 0-form has a single component, a 1-form has two components (e.g., du and dv), and a 2-form has a single component (e.g., $du \wedge dv$).

Conversion Between Discrete Forms

To enforce consistency between our twin representations of forms in our finite-dimensional setting, we need to be able to easily convert from one representation to another. For this purpose, we define two operators: a (primal or dual) upsampling operator, and a (primal or dual) downsampling operator.

From Cochains to Pointwise Components The upsampling operator $\bar{\mathbf{P}}_\uparrow$ (resp., $\widetilde{\mathbf{P}}_\uparrow$) converts primal (resp., dual) k-cochains to $\binom{d}{k}$ components on the component grid $\mathring{\Sigma}$:

$$\bar{\mathbf{P}}_\uparrow : \quad \bar{\Lambda}^k \to \mathring{\Lambda}^k \qquad \widetilde{\mathbf{P}}_\uparrow : \quad \widetilde{\Lambda}^k \to \mathring{\Lambda}^k.$$

Notice that their implentations are simple: one only has to apply the reconstruction operator $\mathcal{R}_N$, followed by a pointwise sampling of the resulting k-form over the vertices of the component grid.

From Pointwise Components to Cochains Conversely, the downsampling operator $\bar{\mathbf{P}}_\downarrow$ (resp., $\widetilde{\mathbf{P}}_\downarrow$) converts a k-form given via its components on the grid $\mathring{\Sigma}$ into primal (resp., dual) k-cochains on $\bar{\Sigma}$ (resp., $\widetilde{\Sigma}$):

$$\bar{\mathbf{P}}_\downarrow : \quad \mathring{\Lambda}^k \to \bar{\Lambda}^k \qquad \widetilde{\mathbf{P}}_\downarrow : \quad \mathring{\Lambda}^k \to \widetilde{\Lambda}^k$$

Note that their implementation is rather simple as well: from the reconstruction of each component using $\mathcal{R}_{2N}$, we simply have to apply the projection operator $\mathcal{P}_n$ to integrate the resulting continuous form onto each of the k-dimensional elements of $\bar{\Sigma}_N$ or $\widetilde{\Sigma}_N$.

Mimetic Conversion Finally, we point out that our downsampling is the left inverse of upsampling, i.e., one has

$$\bar{\mathbf{P}}_\downarrow \bar{\mathbf{P}}_\uparrow = \mathbf{id} \qquad \text{and} \qquad \widetilde{\mathbf{P}}_\downarrow \widetilde{\mathbf{P}}_\uparrow = \mathbf{id}.$$

Thus, as long as we apply first downsampling then upsampling, any differential form originally inside the subspace spanned by the form basis functions of a grid stays unaltered.

Choice of Pointwise Basis for Components Finally, note that a local form basis needs to be defined for the components to be meaningful. To denote in which basis the components are expressed, we will add a subscript to the pointwise component spaces. For example, $\mathring{\Lambda}^k_{\mathfrak{B}}$ will denote k-forms with components given in a basis $\mathcal{B}$ defined at each vertex of $\mathring{\Sigma}$. Furthermore, a given basis $\mathfrak{B}$ can be transformed into another basis $\hat{\mathcal{B}}$ by a non-singular basis change $\mathbf{J}_{\mathcal{B}\hat{\mathcal{B}}}$, thus offering a map between $\mathring{\Lambda}_{\hat{\mathcal{B}}}$ and $\mathring{\Lambda}_{\mathcal{B}}$. This will be particularly convenient later on as we will see that choosing an orthonormal basis with respect to the local metric simplifies a number of metric-dependent operators like the Hodge stars.

Figure 4.10 summarizes all the spaces of forms, both continuous and discrete, and the maps between them. Dashed lines signify maps that are surjections (meaning information may be lost), whereas solid lines indicate injections (where no information is lost).

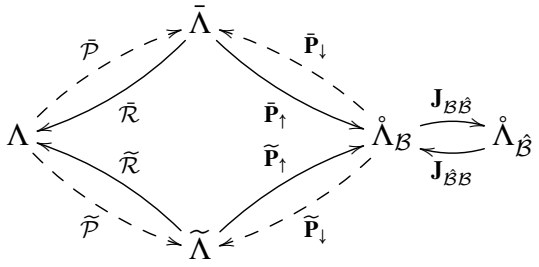


Figure 4.10: Schematic description of the spaces of forms and their various operators.

Discrete Metric

The metric from the original manifold $\mathcal{M}$ can be pulled back onto the computational grid $\mathcal{D}$, and sampled at the component vertices $\mathring{\Sigma}$. For a given basis $\mathcal{B}$, the resulting *discrete metric* can be represented as three components per node, since the metric is always symmetric. Note that a continuous metric field over $\mathcal{D}$ can be easily reconstructed through $\mathcal{R}_{2N}$.

Discrete Vector Fields

Finally, discrete vector fields are also represented as d components on $\mathring{\Sigma}$ for a given vector basis per node. The resulting space of discrete vector fields will be denoted by $\mathring{\mathfrak{X}}$.

4.5 Operators on Discrete Forms and Vector Fields

On Cochains

Since the exterior derivative is a fully topological operator, its most natural implementation is done directly using the cochain representation so as to enforce, by definition, Stokes' theorem,

$$\int_{\sigma} \mathbf{d}\omega = \int_{\partial\sigma} \omega$$

on each mesh element σ. Consequently, the discrete exterior derivative is the signed incidence matrix between $(k+1)$ elements and k elements of the (primal or dual) grid, whose non zero elements are only $+1$ and -1 values to indicate incidence between mesh cells with the sign determined by the relative orientation of the elements:

$$\boxed{\bar{\mathbf{D}} = \bar{\boldsymbol{\partial}}^T}, \qquad \boxed{\widetilde{\mathbf{D}} = \widetilde{\boldsymbol{\partial}}^T},$$

where $\bar{\boldsymbol{\partial}}$ (resp., $\widetilde{\boldsymbol{\partial}}$) refers to the primal (resp., dual) *boundary operator* acting on primal (resp., dual) chains (see [49, 21]). Note that these operators are thus implemented via sparse matrix multiplication. They also satisfy $\bar{\mathbf{D}}^2 = \widetilde{\mathbf{D}}^2 = 0$ like

their continuous equivalent (since the boundary of a boundary is always the empty set). These discrete realizations of the exterior derivative are *exact*, in the sense that the operators commute with their reduction operators:

$$\bar{\mathbf{D}}\bar{\mathcal{P}} = \bar{\mathcal{P}}\mathbf{d} \qquad \text{and} \qquad \widetilde{\mathbf{D}}\widetilde{\mathcal{P}} = \widetilde{\mathcal{P}}\mathbf{d}. \tag{4.6}$$

Therefore, these operators need no special treatment to ensure spectral accuracy. Note that their use for spectral computations represents a clear departure from the conventional spectral methods [17, 69], which evaluate derivatives pointwise and in Fourier space. Based on the enumerations in Figure 4.11, the derivative matrices

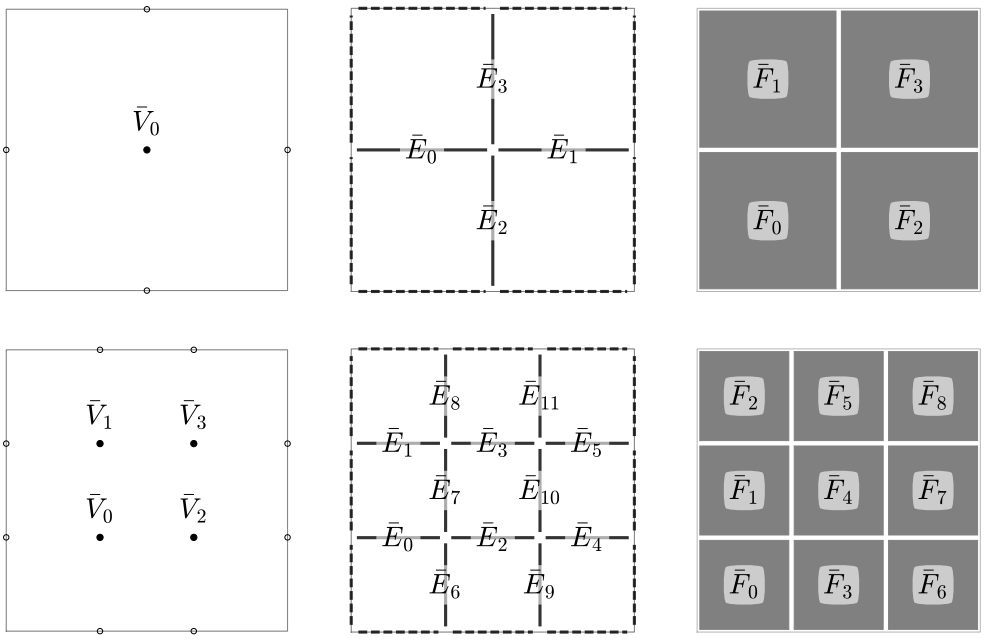


Figure 4.11: 2×2 and 3×3 grids

will be

2×2 **Grid**

$$\bar{\mathbf{D}}^0 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad \bar{\mathbf{D}}^1 = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix},$$

3×3 **Grid**

$$\bar{\mathbf{D}}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \bar{\mathbf{D}}^1 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

and the corresponding matrices on the dual grid will be

$$\widetilde{\mathbf{D}}^0 = \bar{\mathbf{D}}^{1\,T} \qquad \widetilde{\mathbf{D}}^1 = -\bar{\mathbf{D}}^{0\,T}.$$

Boundary Conditions

With our choice of primal and dual Chebyshev grids as defined in Section 4.3, the Poincaré duality operator $*$ is closed under the primal and dual cells

$$\begin{aligned} * : \quad & \bar{\Sigma}^k \to \widetilde{\Sigma}^{d-k}, \\ & \widetilde{\Sigma}^k \to \bar{\Sigma}^{d-k}. \end{aligned}$$

However, the boundary operator ∂ is not closed under the Σs. We need to introduce boundary cells to complete its range. Note that for a boundary cell $\sigma \in B^k$ we have no notion of a dual $*\sigma$.

$$\partial : \quad \Sigma^{k+1} \to \Sigma^k \oplus B^k,$$

Because $\mathbf{D}$ is the adjoint of ∂, this leads to a boundary term in its definition:

$$\mathbf{D} : \quad \Lambda^k \oplus \mathbb{B}^k \to \Lambda^{k+1}.$$

Thus, if $\lambda \in \Lambda^k$ and $\beta \in \mathbb{B}^k$, then because $\mathbf{D}$ is a linear operator, it can be decomposed $\mathbf{D}(\lambda + \beta) = \mathbf{D}(\lambda) + \mathbf{D}(\beta)$ into its restrictions to the internal $\bar{\Lambda}^k$ and boundary $\mathbb{B}^k$ cochain subspaces. In our present setup for the bounded domain the boundary cells are associated with the primal complex, and therefore only the primal $\mathbf{D}$ will have a boundary term.

Let us denote the restriction of $\mathbf{D}$ to $\bar{\Lambda}^k$ with the symbol $\bar{\mathbf{D}}$, and to $\mathbb{B}^k$ with the symbol $\mathbb{D}$:

$$\bar{\mathbf{D}} : \bar{\Lambda}^k \to \bar{\Lambda}^{k+1}, \qquad \mathbb{D} : \mathbb{B}^k \to \bar{\Lambda}^{k+1}.$$

Given a k-cochain λ in $\bar{\Lambda}^k$, and given a boundary condition β in $\mathbb{B}^k$, we will say that $\bar{\mathbf{D}}_\beta \lambda$ is the discrete derivative of λ with boundary condition β:

$$\bar{\mathbf{D}}_\beta \lambda = \bar{\mathbf{D}} \lambda + \mathbb{D} \beta.$$

The dual $\widetilde{\mathbf{D}}$ will have no associated boundary conditions because there are no boundary vertices or edges associated with the dual grid.

On Components

All the operators that involve multiplication need to be performed in the pointwise representation. The pointwise representation is useful for implementing the discrete Hodge star, inner product, sharp, and flat ($\star \cdot \sharp \flat$), which depend on the metric, and the contraction and wedge product ($\wedge \lrcorner$) which do not depend on the metric. The discrete metric is also stored in this representation as a matrix per point. There is no distinction between primal and dual forms in this representation.

Let us denote the components of the discrete component forms ($\mathring{\Lambda}$) using square brackets $[\_]$. To distinguish vectors from forms, we denote the vector components ($\mathring{\mathfrak{X}}$) using $[\![\_]\!]$. The components will simply be numbers expressing the coordinates of the discrete vector or forms relative to a predefined frame of reference. When the frame is orthonormal, the metric dependent operators are very easy to express. For the 2D case, which we implement in this work, the operators are given by:

Hodge Star

$$\mathring{\mathbf{H}}^0 \left(\left[\mathbf{a} \right] \right) = \left[\mathbf{a} \right] \tag{4.7}$$

$$\mathring{\mathbf{H}}^1 \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \end{bmatrix} \right) = \begin{bmatrix} -\mathbf{a}_y \\ \mathbf{a}_x \end{bmatrix} \tag{4.8}$$

$$\mathring{\mathbf{H}}^2 \left(\left[\mathbf{a}_{xy} \right] \right) = \left[\mathbf{a}_{xy} \right] \tag{4.9}$$

Inner Product

$$\mathring{\mathbf{I}}^{00} \left(\left[\mathbf{a} \right], \left[\mathbf{b} \right] \right) = \left[\mathbf{a} \odot \mathbf{b} \right] \tag{4.10}$$

$$\mathring{\mathbf{I}}^{11} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \end{bmatrix}, \begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_y \end{bmatrix} \right) = \left[\mathbf{a}_x \odot \mathbf{b}_x + \mathbf{a}_y \odot \mathbf{b}_y \right] \tag{4.11}$$

$$\mathring{\mathbf{I}}^{22} \left(\left[\mathbf{a}_{xy} \right], \left[\mathbf{b}_{xy} \right] \right) = \left[\mathbf{a}_{xy} \odot \mathbf{b}_{xy} \right] \tag{4.12}$$

The general expressions for dimensions higher than two can be found in Eq. (4.2) and Eq. (4.3).

Flat & Sharp The expressions for the discrete flat and sharp acting on component forms in the orthonormal frame will be given by

$$\mathring{\flat}\left(\left[\!\left[\begin{matrix}\mathbf{a}_x\\ \mathbf{a}_y\end{matrix}\right]\!\right]\right) = \begin{bmatrix}\mathbf{a}_x\\ \mathbf{a}_y\end{bmatrix} \quad \mathring{\sharp}\left(\begin{bmatrix}\mathbf{a}_x\\ \mathbf{a}_y\end{bmatrix}\right) = \left[\!\left[\begin{matrix}\mathbf{a}_x\\ \mathbf{a}_y\end{matrix}\right]\!\right]. \tag{4.13}$$

For the wedge product and contraction one does not need an orthornormal frame, because they are metric independent and invariant under a basis change. Therefore, these operators can be evaluated using the expressions below for any basis frame that is composed of a linearly independent set of vectors.

Wedge Product

$$\mathring{\mathbf{W}}^{00}\left(\left[\mathbf{a}\right],\left[\mathbf{b}\right]\right) = \left[\mathbf{a}\odot\mathbf{b}\right] \tag{4.14}$$

$$\mathring{\mathbf{W}}^{01}\left(\left[\mathbf{a}\right],\begin{bmatrix}\mathbf{b}_x\\ \mathbf{b}_y\end{bmatrix}\right) = \begin{bmatrix}\mathbf{a}\odot\mathbf{b}_x\\ \mathbf{a}\odot\mathbf{b}_y\end{bmatrix} \tag{4.15}$$

$$\mathring{\mathbf{W}}^{02}\left(\left[\mathbf{a}\right],\left[\mathbf{b}_{xy}\right]\right) = \left[\mathbf{a}\odot\mathbf{b}_{xy}\right] \tag{4.16}$$

$$\mathring{\mathbf{W}}^{11}\left(\begin{bmatrix}\mathbf{a}_x\\ \mathbf{a}_y\end{bmatrix},\begin{bmatrix}\mathbf{b}_x\\ \mathbf{b}_y\end{bmatrix}\right) = \left[\mathbf{a}_x\odot\mathbf{b}_y - \mathbf{a}_y\odot\mathbf{b}_x\right] \tag{4.17}$$

Contraction

$$\mathring{\mathbf{C}}^{1}\left(\left[\!\left[\begin{matrix}\mathbf{a}_x\\ \mathbf{a}_y\end{matrix}\right]\!\right],\begin{bmatrix}\mathbf{b}_x\\ \mathbf{b}_y\end{bmatrix}\right) = \left[\mathbf{a}_x\odot\mathbf{b}_x + \mathbf{a}_y\odot\mathbf{b}_y\right] \tag{4.18}$$

$$\mathring{\mathbf{C}}^{2}\left(\left[\!\left[\begin{matrix}\mathbf{a}_x\\ \mathbf{a}_y\end{matrix}\right]\!\right],\left[\mathbf{b}_{xy}\right]\right) = \begin{bmatrix}-\mathbf{a}_y\odot\mathbf{b}_{xy}\\ \mathbf{a}_x\odot\mathbf{b}_{xy}\end{bmatrix} \tag{4.19}$$

On Integrated Forms

The corresponding operators on Dec forms can be easily constructed by mapping from and to the component space via upsampling and downsampling. For example, for the wedge product,

$$\bar{\mathbf{W}}(\mathbf{a},\mathbf{b}) = \bar{\mathbf{P}}_{\downarrow}\,\mathring{\mathbf{W}}(\bar{\mathbf{P}}_{\uparrow}\mathbf{a},\,\bar{\mathbf{P}}_{\uparrow}\mathbf{b}). \tag{4.20}$$

For the Hodge star one needs to transform to an orthonormal frame:

$$\begin{array}{ccccc}
\bar{\Lambda}^k & \xrightarrow{\bar{\mathbf{P}}^k_{\uparrow}} & \mathring{\Lambda}^k_{\mathcal{B}} & \xrightarrow{\mathbf{J}^k_{\mathcal{B}\hat{\mathcal{B}}}} & \mathring{\Lambda}^k_{\hat{\mathcal{B}}} \\
\Big\downarrow{\scriptstyle \bar{\mathbf{H}}^k} & & & & \Big\downarrow{\scriptstyle \mathring{\mathbf{H}}^k} \\
\widetilde{\Lambda}^{n-k} & \xleftarrow[\widetilde{\mathbf{P}}^{n-k}_{\downarrow}]{} & \mathring{\Lambda}^{n-k}_{\mathcal{B}} & \xleftarrow[\mathbf{J}^{n-k}_{\hat{\mathcal{B}}\mathcal{B}}]{} & \mathring{\Lambda}^{n-k}_{\hat{\mathcal{B}}}
\end{array}$$

$$\bar{\mathbf{H}}(\mathbf{a}) = \bar{\mathbf{P}}_{\downarrow}\mathbf{J}_{\hat{\mathcal{B}}\mathcal{B}}\mathring{\mathbf{H}}\mathbf{J}_{\mathcal{B}\hat{\mathcal{B}}}\bar{\mathbf{P}}_{\uparrow}\mathbf{a}. \tag{4.21}$$

Similarly, for the inner product,

$$\bar{\mathbf{I}}(\mathbf{a},\mathbf{b}) = \bar{\mathbf{P}}_{\downarrow}\mathbf{J}_{\hat{\mathcal{B}}\mathcal{B}}\mathring{\mathbf{I}}(\mathbf{J}_{\mathcal{B}\hat{\mathcal{B}}}\bar{\mathbf{P}}_{\uparrow}\mathbf{a},\ \mathbf{J}_{\mathcal{B}\hat{\mathcal{B}}}\bar{\mathbf{P}}_{\uparrow}\mathbf{b}). \tag{4.22}$$

Analogous relationships hold for the dual operators with the bar replaced by a tilde.

4.A Transformations under Coordinate Change

Let us consider how different objects transform under a coordinate change. Forms and vectors transform as follows

$$\hat{\mathbf{e}}^i = J^i_j\mathbf{e}^j, \qquad \hat{\mathbf{e}}_i = \mathbf{e}_j(J^{-1})^j_i,$$

where

$$\mathbf{J} = \begin{bmatrix} J^1_1 & J^1_2 & \cdots \\ J^2_1 & J^2_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

Vectors will be represented by column arrays; forms will be represented by row arrays.

$$X = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \end{bmatrix}, \qquad \alpha = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots \end{bmatrix}.$$

Vectors $\mathfrak{X}$

Vector components transform as

$$\hat{X}^i = J^i_j X^j$$

so that

$$\mathbf{X} = \hat{X}^i\hat{\mathbf{e}}_i = X^i\mathbf{e}_i$$

remains invariant under a transformation.

One-Forms (Covectors) Λ^1

One form components transform as

$$\hat{\alpha}_i = \alpha_j(J^{-1})^j_i$$

so that

$$\alpha = \hat{\alpha}_i \, \hat{\mathbf{e}}^i = \alpha_i \, \mathbf{e}^i$$

remains invariant under such transformation. Notice also the contraction between a vector and a form is independent of the choice of basis:

$$\mathbf{X} \lrcorner \alpha = (\hat{X}^i \hat{\mathbf{e}}_i) \lrcorner (\hat{\alpha}_j \hat{\mathbf{e}}^j) = \hat{X}^i \hat{\alpha}_i = X^i \alpha_i.$$

Zero-Forms Λ^0

$$\hat{f} = f$$

Two-Forms Λ^2

Given a two-form

$$\omega = \omega_{ij} \, \mathbf{e}^i \wedge \mathbf{e}^j,$$

its components transform as

$$\hat{\omega}_{ij} = J_i{}^m J_j{}^n \omega_{mn}.$$

Because of anti-symmetry $\omega_{ij} = -\omega_{ji}$, the only component in two dimensions will be ω_{12}, and the transformation simply becomes

$$\hat{\omega}_{12} = \det(J)^{-1} \, \omega_{12}$$

since

$$\begin{aligned} \hat{\omega}_{12} &= \omega_{12} (J^{-1})^1_1 (J^{-1})^2_2 + \omega_{21} (J^{-1})^2_1 (J^{-1})^1_2 \\ &= \omega_{12} \left[(J^{-1})^1_1 (J^{-1})^2_2 - (J^{-1})^2_1 (J^{-1})^1_2 \right] \\ &= \omega_{12} \det(J)^{-1}. \end{aligned}$$

Again under a coordinate change the contraction will remain invariant:

$$\hat{\omega}(\hat{X}, \hat{Y}) = \omega(X, Y),$$

where

$$\omega(X, Y) = Y \lrcorner X \lrcorner \omega = X^i Y^j \, \omega_{ij}.$$

Metric $\mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$

$$g = g_{ij} \, \mathbf{e}^i \otimes \mathbf{e}^j,$$

$$\hat{g}_{ij} = g_{mn} (J^{-1})^m_i (J^{-1})^n_j,$$

or using matrix notation

$$\hat{\mathbf{g}} = \mathbf{J}^{-T} \mathbf{g} \mathbf{J}^{-1}.$$

Contraction $\lrcorner : \Lambda^1 \times \mathfrak{X} \to \mathbb{R}$

$$X \lrcorner \alpha = \alpha X = \alpha_i X^i$$

Flat $\flat : \mathfrak{X} \to \Lambda^1$

$$\alpha_i = g_{ij} X^j$$

$$\alpha = X^\flat = X^T \mathbf{g}$$

Sharp $\sharp : \Lambda^1 \to \mathfrak{X}$

$$X^i = g^{ij} \alpha_j$$

$$X = \alpha^\sharp = \mathbf{g}^{-1} \alpha^T$$

Inner Products

$\mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$

$$X \cdot Y = X^i \, g_{ij} \, Y^j$$

$$X \cdot Y = X^T \, \mathbf{g} \, Y$$

$$X \cdot Y = X^\flat \cdot Y^\flat = X \lrcorner Y^\flat$$

$\Lambda^1 \times \Lambda^1 \to \mathbb{R}$

$$\alpha \cdot \beta = \alpha_i g^{ij} \beta_j$$

$$\alpha \cdot \beta = \alpha \mathbf{g}^{-1} \beta^T$$

$$\alpha \cdot \beta = \alpha^\sharp \cdot \beta^\sharp = \alpha^\sharp \lrcorner \beta$$

4.B Chebyshev Identities and Limits

The higher order derivatives of Chebyshev polynomials can be expressed in terms of their lower order counterparts. The basis functions used in this work require the

derivatives up to order two.

$$
\begin{aligned}
T_0(x) &= 1 & U_0(x) &= 1 \\
T_1(x) &= x & U_1(x) &= 2x \\
T_N(x) &= 2xT_N(x) - T_{N-1}(x) & U_N(x) &= 2xU_N(x) - U_{N-1}(x) \\
T'_N(x) &= NU_{N-1}(x) & U'_N(x) &= \frac{xU_N(x) - (N+1)T_{N+1}(x)}{1-x^2} \\
T''_N(x) &= NU'_{N-1}(x) & U''_N(x) &= \frac{3xU'_N(x) - N(N+2)U_N(x)}{1-x^2}
\end{aligned}
$$

For some of the expressions, it is necessary to evaluate T_N and U_N at the endpoints $x = \pm 1$. While formulas exist for T_N in the literature, no similar expressions were found for U_N.

$$
\begin{aligned}
T_N(\pm 1) &= (\pm 1)^N \\
T'_N(\pm 1) &= N^2(\pm 1)^N \\
T''_N(\pm 1) &= \frac{1}{3}N^2(N-1)(N+1)(\pm 1)^N
\end{aligned}
$$

The expressions below were derived empirically, and they have been checked to hold true for up to a very large N using a modern computer.

$$
\begin{aligned}
U_N(\pm 1) &= (N+1)(\pm 1)^N \\
U'_N(\pm 1) &= \frac{1}{3}N(N+1)(N+2)(\pm 1)^N \\
U''_N(\pm 1) &= \frac{1}{15}(N-1)N(N+1)(N+2)(N+3)(\pm 1)^N
\end{aligned}
$$

Chapter 5

IMPLEMENTATION, VALIDATION, AND RESULTS

5.1 Implementation

The Spex operators are constructed as compositions of the one-dimensional elementary operators that we consider here. For the periodic and bounded domains, we need to define the following Spex operators for $k = \{0, 1\}$:

$$\begin{array}{cccc} \bar{\mathbf{D}}^k & \bar{\mathbf{H}}^k & \bar{\mathbf{P}}^k_\uparrow & \bar{\mathbf{P}}^k_\downarrow \\ \widetilde{\mathbf{D}}^k & \widetilde{\mathbf{H}}^k & \widetilde{\mathbf{P}}^k_\uparrow & \widetilde{\mathbf{P}}^k_\downarrow. \end{array}$$

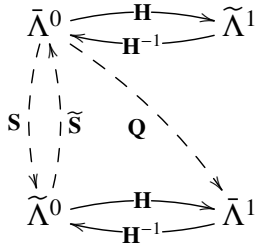
Because of the two possible values for k there are sixteen operators to implement in total. Only one-dimensional operators are provided, because higher-dimensional ones can be constructed trivially as compositions of the one-dimensional ones along each dimension of the computational domain. For example, the wedge product and the contraction are composite operators that can be expressed in terms of $\mathbf{P}_\uparrow$ and $\mathbf{P}_\downarrow$ by Eq. (4.20) and Eq. (4.22), respectively.

Building Blocks

The Spex operators are defined in terms of the following basic operators which are used as elementary building blocks:

$$\mathbf{H} = \widetilde{\mathcal{P}} \star \bar{\mathcal{R}}, \qquad \mathbf{H}^{-1} = \bar{\mathcal{P}} \star^{-1} \widetilde{\mathcal{R}}, \qquad \mathbf{S} = \widetilde{\mathcal{P}}\bar{\mathcal{R}}, \qquad \widetilde{\mathbf{S}} = \bar{\mathcal{P}}\widetilde{\mathcal{R}}, \qquad \mathbf{Q} = \bar{\mathcal{P}} \star \bar{\mathcal{R}}.$$

These operators correspond to maps between the discrete cochain spaces on the periodic domain.



(5.1)

In the above formulation, it is clear that the basis functions and the geometry of the cells alone completely determine **H**, **S**, and **Q** through the reconstruction map $\mathcal{R}$ and the reduction map $\mathcal{P}$. However, any implementation that relies on the above definitions is slow as it must construct dense matrices by evaluating the functions ϕ

on the corresponding cells σ. In this section we provide an implementation via the discrete *fast Fourier transforms* (FFT) denoted $\mathcal{F}$ and its inverse $\mathcal{F}^*$, as well as in terms of other fast array operations, and use the basis formulation to validate that the provided implementation is indeed correct.

We can efficiently express our building blocks using Fourier space:

$$\mathbf{H} = \mathcal{F}^* \widehat{\mathbf{H}} \mathcal{F}, \qquad \mathbf{S} = \mathcal{F}^* \widehat{\mathbf{S}} \mathcal{F}, \qquad \mathbf{Q} = \mathcal{F}^* \widehat{\mathbf{Q}} \mathcal{F},$$

the Fourier space operators are diagonal matrices whose entries are given by

$$\begin{aligned}
\widehat{\mathbf{H}}_{qq} &= \frac{2}{k_q} \sin\left(\frac{\pi k_q}{N}\right), \\
\widehat{\mathbf{S}}_{qq} &= \exp\left(\frac{i\pi k_q}{N}\right), \\
\widehat{\mathbf{Q}}_{qq} &= \frac{1}{ik_q}\left(\exp\left(\frac{2i\pi k_q}{N}\right) - 1\right),
\end{aligned}$$

for $q = 0, 1, \ldots, N-1$, and where k_q refers to the FFT sampling frequencies, i.e.,

$$k_q = \begin{cases} q & q < (N+1)/2 \\ q - N & \text{otherwise} \end{cases}.$$

The time complexity of the above operators is thus the same as that of the FFT which is $\mathcal{O}(N \log N)$. Having defined the FFT based operators $\mathbf{H}$, $\mathbf{Q}$, and $\mathbf{S}$, we need to define a few more fast array operations (which we call *auxiliary operators*). The auxiliary operators provide the infrastructure upon which the rest of the higher level Spex operators are most efficiently built. Usually they act on arrays of size N and have an output that is also an array. The auxiliary operators have time complexity linear in the size of their input $\mathcal{O}(N)$. Table 5.1 lists all the operators that will be used for defining the full hierarchy of Spex operators. A more detailed description of each individual operator is provided in 5.A.

Periodic Domain

Derivatives:

$$\begin{aligned}
\bar{\mathbf{D}}^0(f) &= \text{roll}^{-1}(f) - f & \bar{\mathbf{D}}^1(f) &= 0 \\
\widetilde{\mathbf{D}}^0(f) &= f - \text{roll}^{+1}(f) & \widetilde{\mathbf{D}}^1(f) &= 0
\end{aligned}$$

Hodge stars:

$$\begin{aligned}
\bar{\mathbf{H}}^0(f) &= \mathbf{H}(f) & \bar{\mathbf{H}}^1(f) &= \mathbf{H}^{-1}(f) \\
\widetilde{\mathbf{H}}^0(f) &= \mathbf{H}(f) & \widetilde{\mathbf{H}}^1(f) &= \mathbf{H}^{-1}(f)
\end{aligned}$$

Auxilary Operator	Description
diff	discrete difference of an array
roll	cyclically roll an array
slice	extract elements from an array
even	extract even elements from an array
odd	extract odd elements from an array
weave	merge two arrays by alternating their elements
concat	concatenate two arrays sequentially
mid	extract the middle elements of an array
rev	reverse an array in order
mirror	concatenate an array with its mirror image

Table 5.1: List of auxilary operators

Upsampling:

$$\bar{\mathbf{P}}^0_\uparrow(f) = \text{weave}(f, \mathbf{S}(f)) \qquad \bar{\mathbf{P}}^1_\uparrow(f) = \widetilde{\mathbf{P}}^0_\uparrow(\bar{\mathbf{H}}^1(f))$$
$$\widetilde{\mathbf{P}}^0_\uparrow(f) = \text{weave}(\widetilde{\mathbf{S}}(f), f) \qquad \widetilde{\mathbf{P}}^1_\uparrow(f) = \bar{\mathbf{P}}^0_\uparrow(\widetilde{\mathbf{H}}^1(f))$$

Downsampling:

$$\bar{\mathbf{P}}^0_\downarrow = \text{even}(f) \qquad \bar{\mathbf{P}}^1_\downarrow = \text{evenodd}(\mathbf{Q}(f))$$
$$\widetilde{\mathbf{P}}^0_\downarrow = \text{odd}(f) \qquad \widetilde{\mathbf{P}}^1_\downarrow = \text{evenodd}(\text{roll}^{+1}(\mathbf{Q}(f)))$$

where

$$\text{evenodd}(f) = \text{even}(f) + \text{odd}(f)$$

Bounded Domain

In the non-periodic Chebyshev domain, integration requires multiplication by non-uniform weights. Depending on the sampling points (primal or dual) we need to multiply by one of the following two diagonal weight matrices:

$$\mathbf{W}_{qq} = \sin\frac{q\pi}{N-1} \qquad q = \{0, 1, \ldots, N-1\},$$
$$\widetilde{\mathbf{W}}_{qq} = \sin\frac{(q+\frac{1}{2})\pi}{N} \qquad q = \{0, 1, \ldots, N-2\}.$$

$\mathbf{A}^{00}$ pads the array with zeros on both sides. $\mathbf{A}^{bb}$ pads the array by extrapolating the boundary values and adding them to the array.

$$\begin{aligned}\mathbf{A}^{00}(f) &= \text{concat}\left([0],\, f,\, [0]\right),\\ \mathbf{A}^{bb}(f) &= \text{concat}\left([b(f,-1)],\, f,\, [b(f,+1)]\right),\\ \mathbf{A}^{\dagger}(f) &= \text{mid}(f),\end{aligned}$$

with the extrapolated boundary values given by

$$b(f,x) = \sum_{n=0}^{N} f_n\, \phi_n^0(x).$$

$\mathbf{A}^{\dagger}$ is the left inverse of the padding operators, i.e. $\mathbf{A}^{\dagger}\mathbf{A}^{00} = \mathbf{Id}$ and $\mathbf{A}^{\dagger}\mathbf{A}^{bb} = \mathbf{Id}$.

The mirroring matrices are

$$\begin{aligned}\mathbf{M}_0^{\pm}(f) &\coloneqq \text{concat}(f,\, \pm\text{mid}(\text{rev}(f))),\\ \mathbf{M}_1^{\pm}(f) &\coloneqq \text{concat}(f,\, \pm\text{rev}(f)),\\ \mathbf{M}_0^{\dagger}(f) &\coloneqq \text{slice}^{0:\lfloor \frac{N}{2} \rfloor + 1}(f),\\ \mathbf{M}_1^{\dagger}(f) &\coloneqq \text{slice}^{0:\lfloor \frac{N}{2} \rfloor}(f).\end{aligned}$$

$\mathbf{M}_0^{\dagger}$ and $\mathbf{M}_1^{\dagger}$ are the left inverses of the above operators and they satisfy $\mathbf{M}_0^{\dagger}\mathbf{M}_0^{\pm} = \mathbf{Id}$ and $\mathbf{M}_1^{\dagger}\mathbf{M}_1^{\pm} = \mathbf{Id}$. The mirroring operations are explicitly given in section 5.A.

Derivatives:

$$\begin{aligned}\bar{\mathbf{D}}^0(f) &= \text{diff}(\mathbf{A}^{00}(f)) & \bar{\mathbf{D}}^1(f) &= 0\\ \widetilde{\mathbf{D}}^0(f) &= \text{diff}(f) & \widetilde{\mathbf{D}}^1(f) &= 0\end{aligned}$$

Hodge stars:

$$\begin{aligned}\bar{\mathbf{H}}^0(f) &= \mathbf{A}^{\dagger}(\mathbf{M}_0^{\dagger}(\mathbf{H}(\mathbf{M}_0^{-}(\mathbf{W}(\mathbf{A}^{00}(f)))))) & \bar{\mathbf{H}}^1(f) &= \widetilde{\mathbf{W}}^{-1}(\mathbf{M}_1^{\dagger}(\mathbf{H}^{-1}(\mathbf{M}_1^{-}(f))))\\ \widetilde{\mathbf{H}}^0(f) &= \mathbf{M}_1^{\dagger}(\mathbf{H}(\mathbf{M}_1^{-}(\widetilde{\mathbf{W}}(f)))) & \widetilde{\mathbf{H}}^1(f) &= \mathbf{A}^{\dagger}(\mathbf{W}^{-1}(\mathbf{M}_0^{\dagger}(\mathbf{H}^{-1}(\mathbf{M}_0^{-}(\mathbf{A}^{00}(f))))))\end{aligned}$$

$$\begin{aligned}\mathbb{S}(f) &= \mathbf{M}_1^{\dagger}(\mathbf{S}(\mathbf{M}_0^{+}(\mathbf{A}^{bb}(f)))) & \widetilde{\mathbb{S}}(f) &= \mathbf{A}^{\dagger}(\mathbf{M}_0^{\dagger}(\widetilde{\mathbf{S}}(\mathbf{M}_1^{+}(f))))\\ \mathbb{Q}(f) &= \mathbf{M}_1^{\dagger}(\mathbf{Q}(\mathbf{M}_0^{-}(\mathbf{W}(\mathbf{A}^{00}(f))))) & \widetilde{\mathbb{Q}}(f) &= \mathbf{A}^{\dagger}(\mathbf{W}^{-1}(\mathbf{M}_0^{\dagger}(\widetilde{\mathbf{Q}}(\mathbf{M}_1^{-}(f)))))\end{aligned}$$

Upsampling:

$$\bar{\mathbf{P}}^0_\uparrow(f) = \text{weave}(f, \mathbb{S}(f)) \qquad \bar{\mathbf{P}}^1_\uparrow(f) = \widetilde{\mathbf{P}}^0_\uparrow(\bar{\mathbf{H}}^1(f))$$
$$\widetilde{\mathbf{P}}^0_\uparrow(f) = \text{weave}(\widetilde{\mathbb{S}}(f), f) \qquad \widetilde{\mathbf{P}}^1_\uparrow(f) = \bar{\mathbf{P}}^0_\uparrow(\widetilde{\mathbf{H}}^1(f))$$

Downsampling:

$$\bar{\mathbf{P}}^0_\downarrow(f) = \text{odd}(f) \qquad \bar{\mathbf{P}}^1_\downarrow(f) = \text{evenodd}(\mathbb{Q}(f))$$
$$\widetilde{\mathbf{P}}^0_\downarrow(f) = \text{even}(f) \qquad \widetilde{\mathbf{P}}^1_\downarrow(f) = \text{evenodd}(\mathbf{A}^\dagger(\mathbb{Q}(f)))$$

5.2 Convergence and Validation

The Laplace-Beltrami operator is a composition of almost all the operators that we have implemented in the previous section. On two-dimensional domains, it acts on 0-forms and 1-forms (2-forms are omitted because we do not use them for our convergence and validation tests) as follows:

$$\Delta = \begin{cases} \widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^1\mathbf{H}^1\mathbf{D}^0 & \bar{\Lambda}^0 \to \bar{\Lambda}^0 \\ \mathbf{H}^1\mathbf{D}^1\widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^0 & \widetilde{\Lambda}^0 \to \widetilde{\Lambda}^0 \\ \widetilde{\mathbf{H}}^1\widetilde{\mathbf{D}}^0\mathbf{H}^2\mathbf{D}^1 + \mathbf{D}^0\widetilde{\mathbf{H}}^2\widetilde{\mathbf{D}}^1\mathbf{H}^1 & \bar{\Lambda}^1 \to \bar{\Lambda}^1 \\ \mathbf{H}^1\mathbf{D}^0\widetilde{\mathbf{H}}^2\widetilde{\mathbf{D}}^1 + \widetilde{\mathbf{D}}^0\mathbf{H}^2\mathbf{D}^1\widetilde{\mathbf{H}}^1 & \widetilde{\Lambda}^1 \to \widetilde{\Lambda}^1 \end{cases}.$$

If we can show the correctness of the Laplace-Beltrami implementation, it is proof of correctness for implementation of the operators it is composed of as well. For that purpose, we define the 0-form f^0 and the 1-form f^1, which will be used as test functions that will be fed to the Δ operator:

$$f^0(x, y) = e^x + e^y, \qquad f^1(x, y) = e^x\,\mathbf{d}x + e^y\,\mathbf{d}y,$$
$$\Delta f^0(x, y) = e^x + e^y, \qquad \Delta f^1(x, y) = e^x\,\mathbf{d}x + e^y\,\mathbf{d}y.$$

Each one will be tested on the grids generated by the diffeomorphisms $\varphi : \mathcal{D} \to \mathcal{M}$ that will map the reference computational grid to physical space:

$$\begin{aligned} \varphi_0 :&\quad (x, y) \to (x,\ y), \\ \varphi_1 :&\quad (x, y) \to (x + \epsilon \sin(\pi x)\sin(\pi y),\ y + \epsilon \sin(\pi x)\sin(\pi y)) && \epsilon = {}^{13}/_{100}, \\ \varphi_2 :&\quad (x, y) \to (x,\ y - \epsilon \sin(\pi x)) && \epsilon = {}^{1}/_{3}, \\ \varphi_3 :&\quad (x, y) \to (x^2 - y^2,\ 2xy). \end{aligned}$$

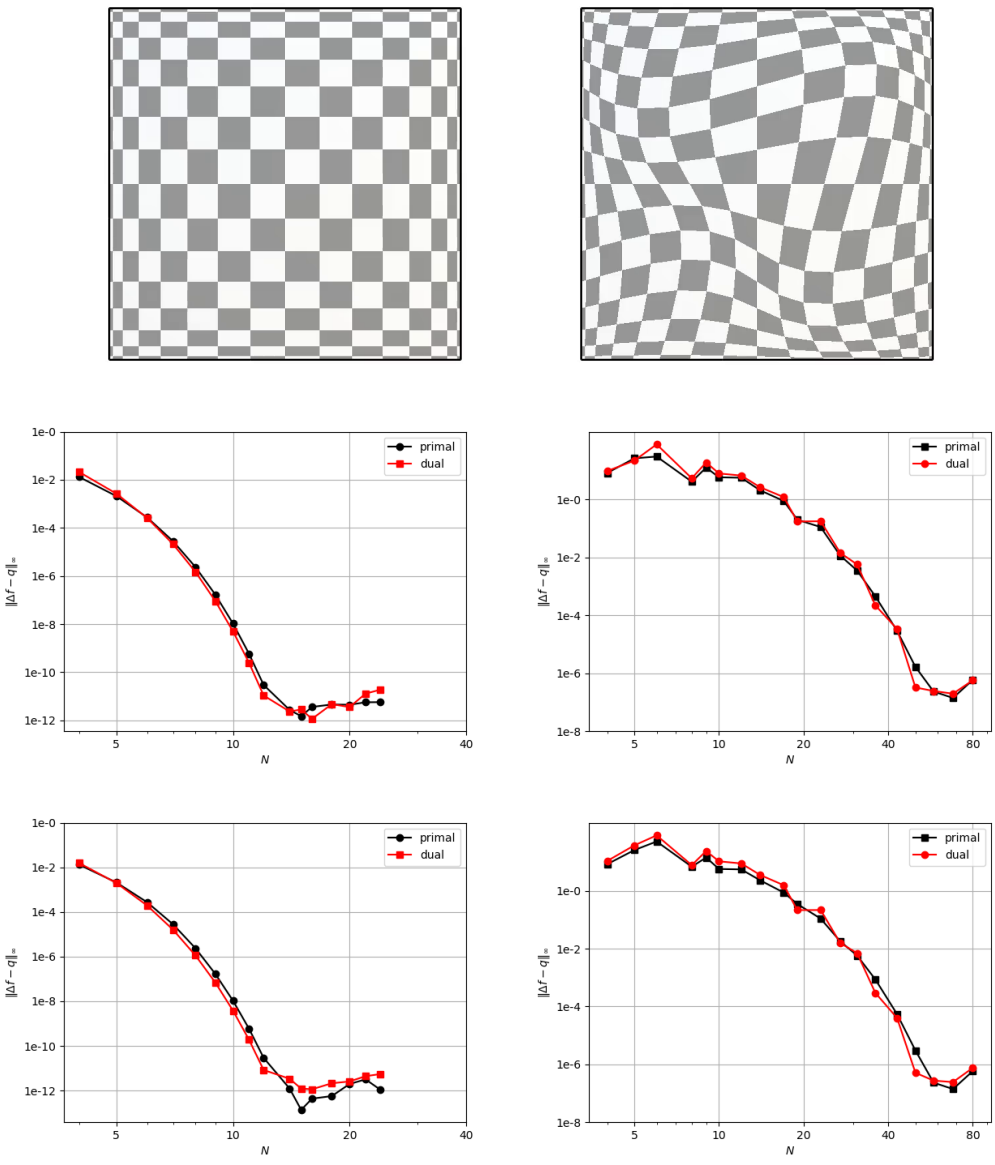


Figure 5.1: Convergence for the Laplace operator in a square domain with rectilinear and curvilinear Chebyshev grids for 0-forms (top) and 1-forms (bottom).

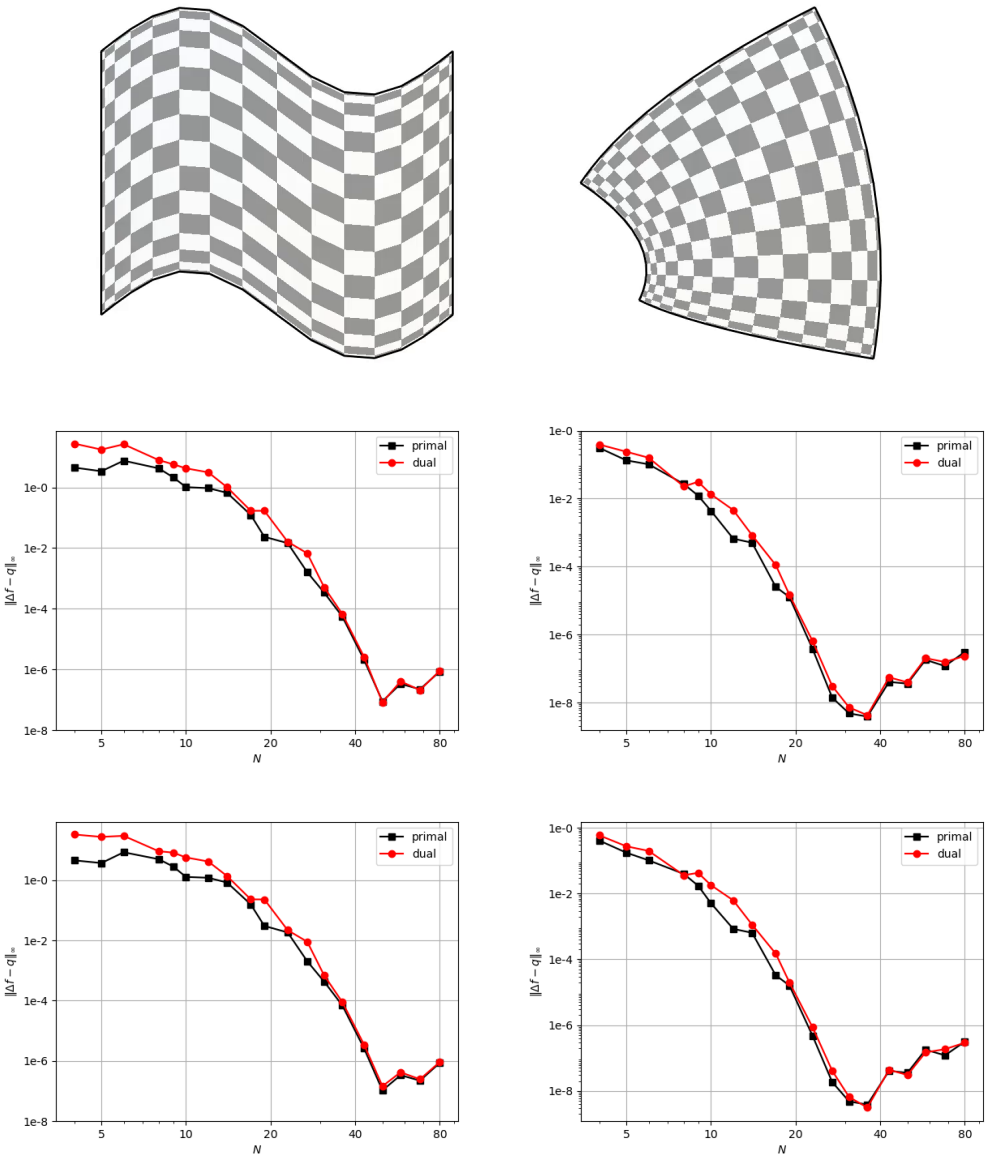


Figure 5.2: Convergence of the Laplace operator in a non-square domain.

In order to show that our method converges, we are going to apply the numerical Laplacian to the functions above in each one of the grids, and compare the output to the expected exact solution. As we increase the grid resolution we expect to see a faster than exponential (supergeometric) decrease in the error. Refer to Figure 5.1 for convergence of the Laplace-Beltrami operator acting on f^0 and f^1 for the grids generated by φ_0 and φ_1, which have rectilinear boundaries. For curvilinear boundaries, corresponding to the diffeomorphisms φ_2 and φ_3, refer to Figure 5.2.

The figures display logarithmic plots of the L_∞ error between the expected and the actual output of the discrete operator. On these plots, spectral convergence looks like curves that develop an increasingly negative slope relative to a straight line. Eventually the round-off plateau is reached at which point the error flattens out and begins to slowly increase due to the accumulation errors from the rising number of floating point operations. Our method converges much faster on flat domains ($N \sim 15$), where no metric related manipulations are performed, and is somewhat slower ($N \sim 60$), when done on curved domains because of the extra computations related to the transformation to and from an orthonormal frame, which contributes to the computational error.

5.3 Applications

Having provided the detail of the implementation, and having proved its correctness, we now proceed to show various applications to problems in physics.

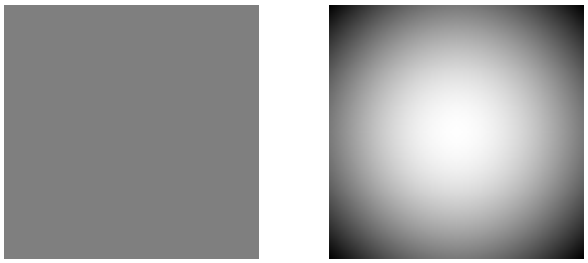
For the purposes of this section, plots for 1-forms are generated using the line integral convolution (LIC) method [16], which is a popular method to visualize vector fields. In order to visualize 1-forms, we must first convert them to vector fields by taking their sharp, and then use the result as input to the LIC algorithm which then outputs a pixel map of the visualization. Taking the sharp is necessary because 1-forms are not appropriate for generating curves along which to take convolutions, vector fields are.

Laplace's Equation

Laplace's equation is ubiquitous in physics and mathematics. It is an elliptic partial differential equation, that can be readily generalized to k-forms denoted as f using the Laplace-Beltrami operator as

$$\Delta f = 0.$$

Forms that satisfy Laplace's equation are known as **harmonic** forms.



(a) $f|_{\mathcal{B}} = 1$ **(b)** $\star\mathbf{d}f|_{\mathcal{B}} = 1$

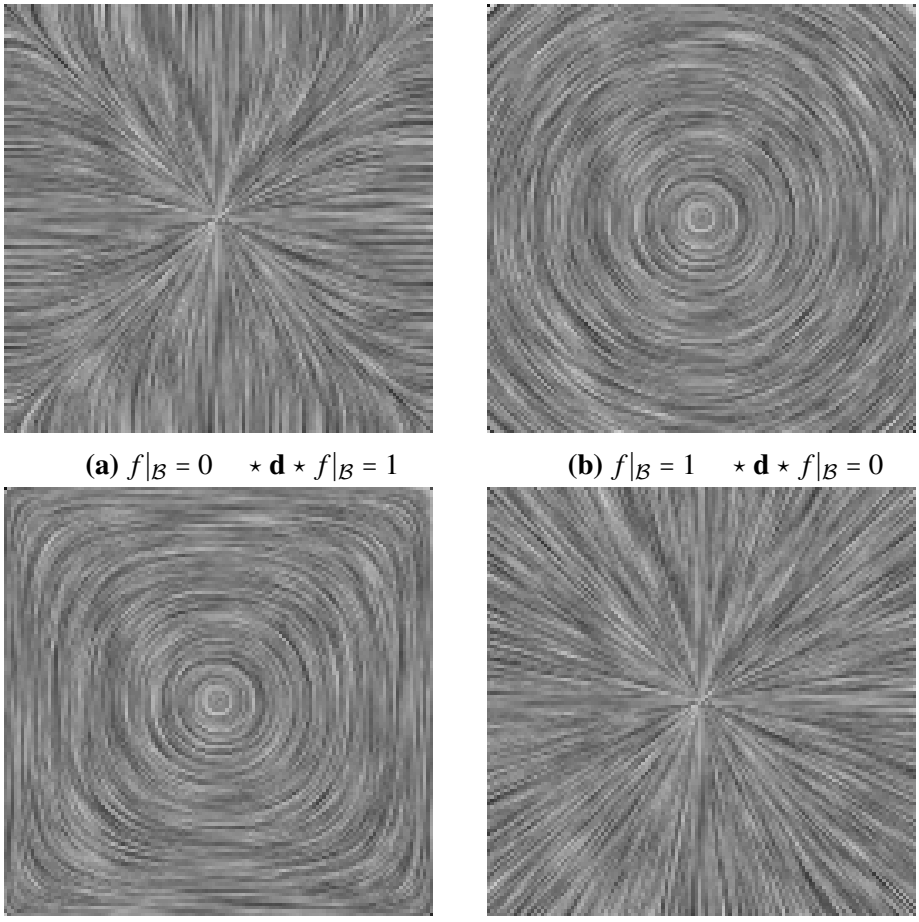
Figure 5.3: Laplace's equation for a 0-form f. The 0-form is represented using a greyscale colorplot. The Dirichlet boundary condition (a) results in a uniform solution, whereas the Neumann boundary condition (b) results in a quadratic polynomial.

As is well known from the classical theory of partial differential equations [24], for scalar fields (i.e., 0-forms in our present framework) we distinguish between two kinds of boundary conditions - Dirichlet and Neumann. The Dirichlet boundary conditions fix the function at the boundary $f|_{\mathcal{B}}$ while Neumann boundary conditions fix the normal component of its gradient $(\star\mathbf{d}f)|_{\mathcal{B}}$. We can easily reproduce these boundary conditions in the present framework (see Figure 5.3).

Because of the generality of the Laplace-Beltrami operator and the possibility to apply it to higher-dimensional forms, we can solve Laplace's equation for 1-forms in addition to scalar fields. In the case of 1-forms the treatment for boundary conditions is richer than in the case of forms of degree 0. We can either set $f|_{\mathcal{B}}$ and $(\star\mathbf{d}\star f)|_{\mathcal{B}}$ along the boundary, or we can set $(\star f)|_{\mathcal{B}}$ and $(\star\mathbf{d}f)|_{\mathcal{B}}$. Note that in the first case we are prescribing the tangential component and the divergence, whereas in the second case we are setting the normal component and the curl. Figure 5.4 illustrates the different possibilities for combining such boundary conditions. Note that the dual solutions at the bottom row are 90° rotations of the primal ones at the top. This is because the two boundary conditions are related by the transformation $f \mapsto \star f$, and therefore the solutions must also be.

For the exact details of what matrix equation is solved for Laplace's equation with boundary conditions and how that corresponds to the different cases described above see Appendix 5.B.

Further illustration of the generality of our approach is that it is not limited to flat domains. The Laplace-Beltrami operator has been implemented for non-flat domains which induce a metric on the computational grid that may be different from the identity. We demonstrate such domains in Figure 5.5. On the first row we show the solution on a flat grid that is identical to the reference grid $\mathcal{D}$. This solution



(a) $f|_{\mathcal{B}} = 0 \quad \star\, \mathbf{d} \star f|_{\mathcal{B}} = 1$ **(b)** $f|_{\mathcal{B}} = 1 \quad \star\, \mathbf{d} \star f|_{\mathcal{B}} = 0$

(c) $\star f|_{\mathcal{B}} = 0 \quad \star\, \mathbf{d}\, f|_{\mathcal{B}} = 1$ **(d)** $\star f|_{\mathcal{B}} = 1 \quad \star\, \mathbf{d}\, f|_{\mathcal{B}} = 0$

Figure 5.4: Plots for the solutions of Laplace's Equation on a 1-form f with different combinations of boundary conditions. The domain is $[-1, 1]^2$, discretized by a 16×16 Chebyshev grid. On the top row we solve for f on the primal grid, whereas on the bottom on the dual grid.

will be used for comparison with the second row, where we preserve the boundaries exactly but internally deform the grid. Because the boundary remains the same, the solution on the physical grid must be identical to the solution of the undeformed domain, which, as is evident from the figure, is indeed the case. This is further evidence for the validity of the computations and transformations that are performed in our implementation. While the solutions on the computational reference grids are different, their push-forwards to the physical domain match exactly. The last two rows show the solutions for grids that have curvilinear boundaries.

So far we have only considered cases where the physical grid is embedded in 2D, but there is no difficulty if one wishes to embed in 3D. On Figure 5.6 we illustrate solutions with boundary conditions, which in the flat case result in a trivial uniform

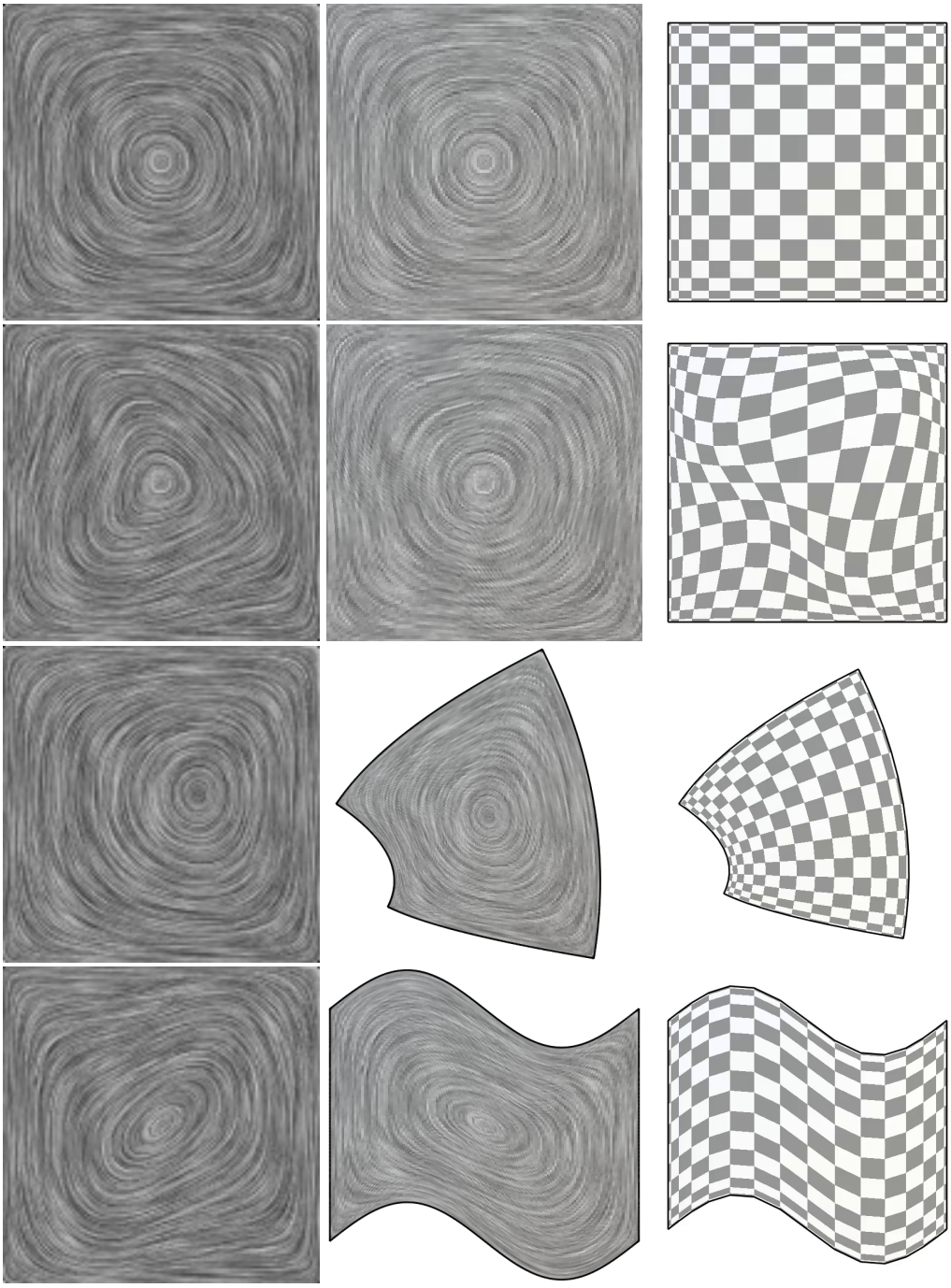


Figure 5.5: Solution to Laplace's equation for 1-forms on curved domains with boundary conditions $\star f|_{\mathcal{B}} = 0$ and $\star \mathbf{d} f|_{\mathcal{B}} = 1$. The rightmost column shows the grid, the middle column shows the solution, and the leftmost column shows the pull-back of the solution to the reference grid.

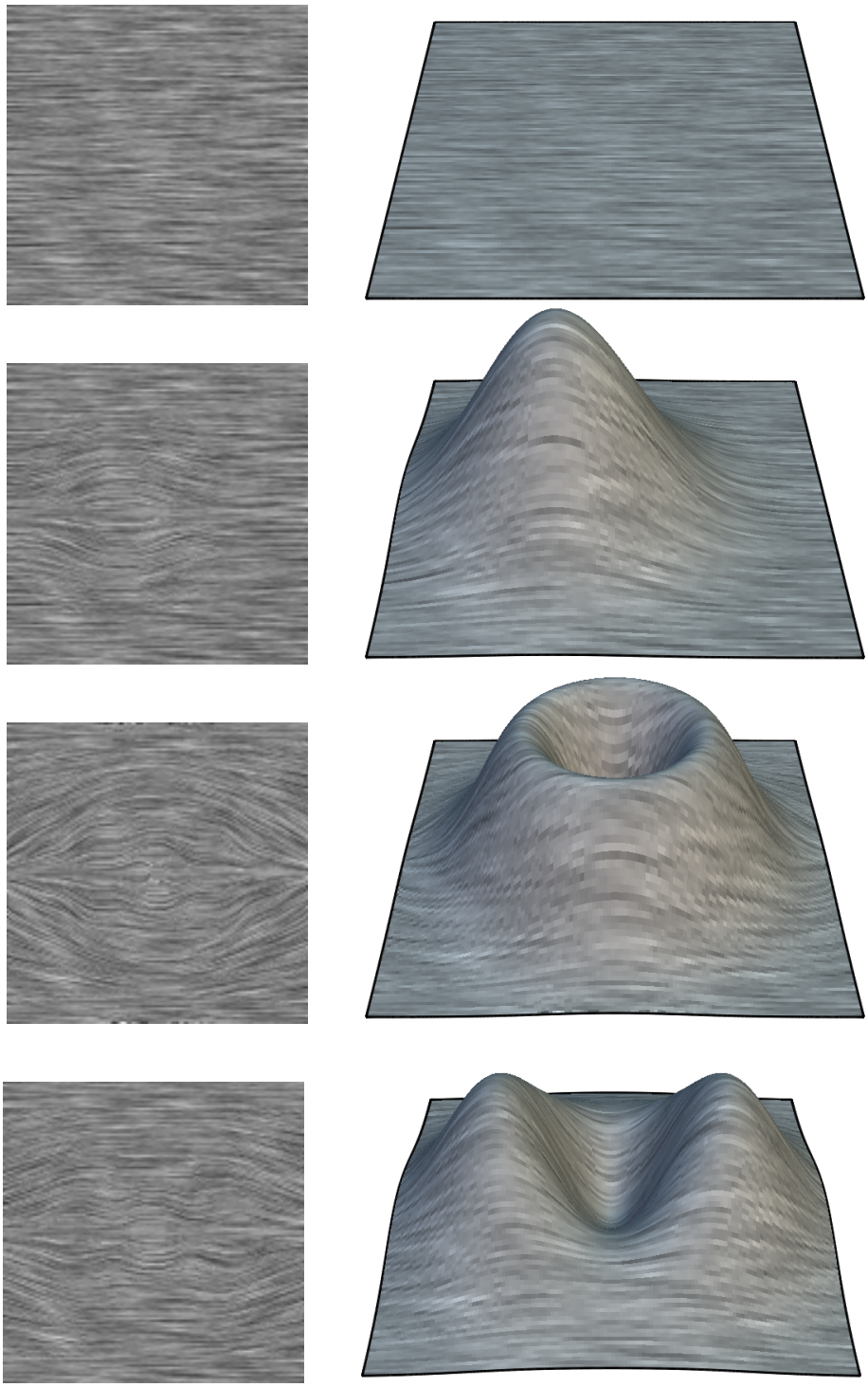


Figure 5.6: Laplace's equation for 1-forms on non-flat domains with boundary conditions $\star f|_{\mathcal{B}_{\text{left}}} = 1$, $\star f|_{\mathcal{B}_{\text{right}}} = -1$, $\star f|_{\mathcal{B}_{\text{top,bottom}}} = 0$ and $\star \mathbf{d} f|_{\mathcal{B}} = 0$.

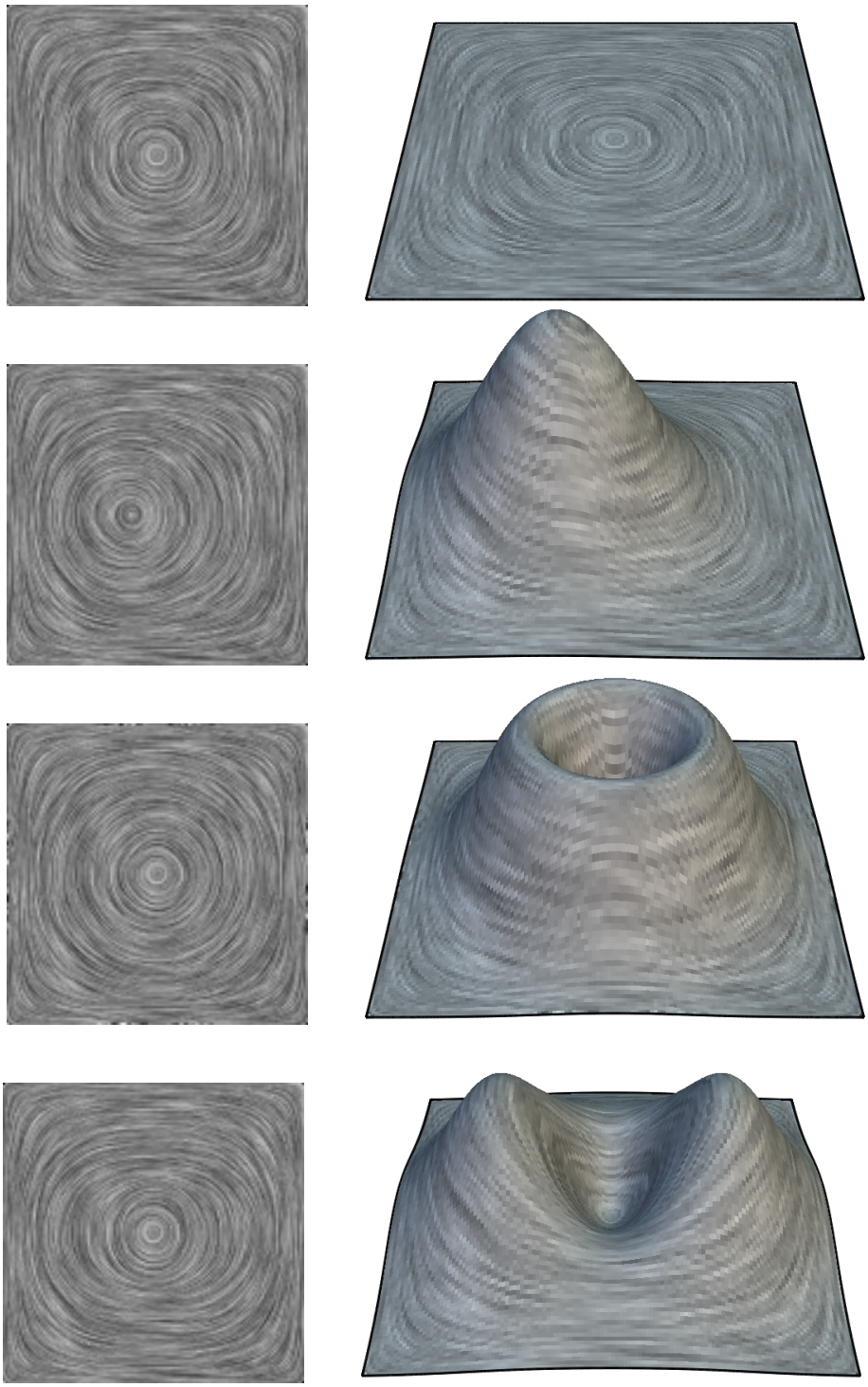


Figure 5.7: Laplace's equation for 1-forms on non-flat domains with boundary conditions $\star f|_{\mathcal{B}} = 0$ and $\star \mathbf{d} f|_{\mathcal{B}} = 1$.

field. Because the Laplace-Beltrami operator is metric dependent, the deformation modifies the solution in different ways for each surface. Figure 5.7 illustrates solutions with boundary conditions that result in a rotational field. Again depending on the surface, the solutions on the reference grid are modified in comparison to the flat domain.

Poisson's Equation

Poisson's equation is a non-homogenuous differential equation that adds a forcing term q to Laplace's equation:

$$\Delta f = q.$$

In this section we illustrate the usefulness of Poisson's equation in computing a divergence-free velocity field that matches a prescribed vorticity field. While computing the vorticity from a velocity field is straightforward, doing the reverse is not. Being able to compute the velocity from vorticity is important in numerical fluid mechanics, where the vorticity field is being advected by the divergence-free velocity field. Computing the velocity field at each time step would significantly reduce the memory footprint of the simulation as one would need to store only the vorticity, rather than both the vorticity and velocity.

The vorticity 2-form of a velocity field 1-form is defined as

$$\omega = \mathbf{d}v.$$

Given a vorticity 2-form, we will compute a divergence-free velocity 1-form from it. By the Hodge decomposition theorem, the velocity 1-form can be expressed as

$$v = \mathbf{d}\alpha + \star\mathbf{d}\star\beta + \gamma$$

where $\alpha \in \Lambda^0$, $\beta \in \Lambda^2$, and $\gamma \in \Lambda^1$ is harmonic. Because we require the velocity field to be divergence-free, $\mathbf{d} \star v = 0$, we must set $\alpha = 0$. For simplicity we will ignore the harmonic term and set $\gamma = 0$. Then,

$$v = \star\mathbf{d}\star\beta.$$

Applying $\mathbf{d}$ to both sides, we obtain

$$\mathbf{d}\star\mathbf{d}\star\beta = \omega.$$

This is precisely Poisson's equation for 2-forms. Generally the right hand side of the above equation is thought of as an input of some kind that drives the solution

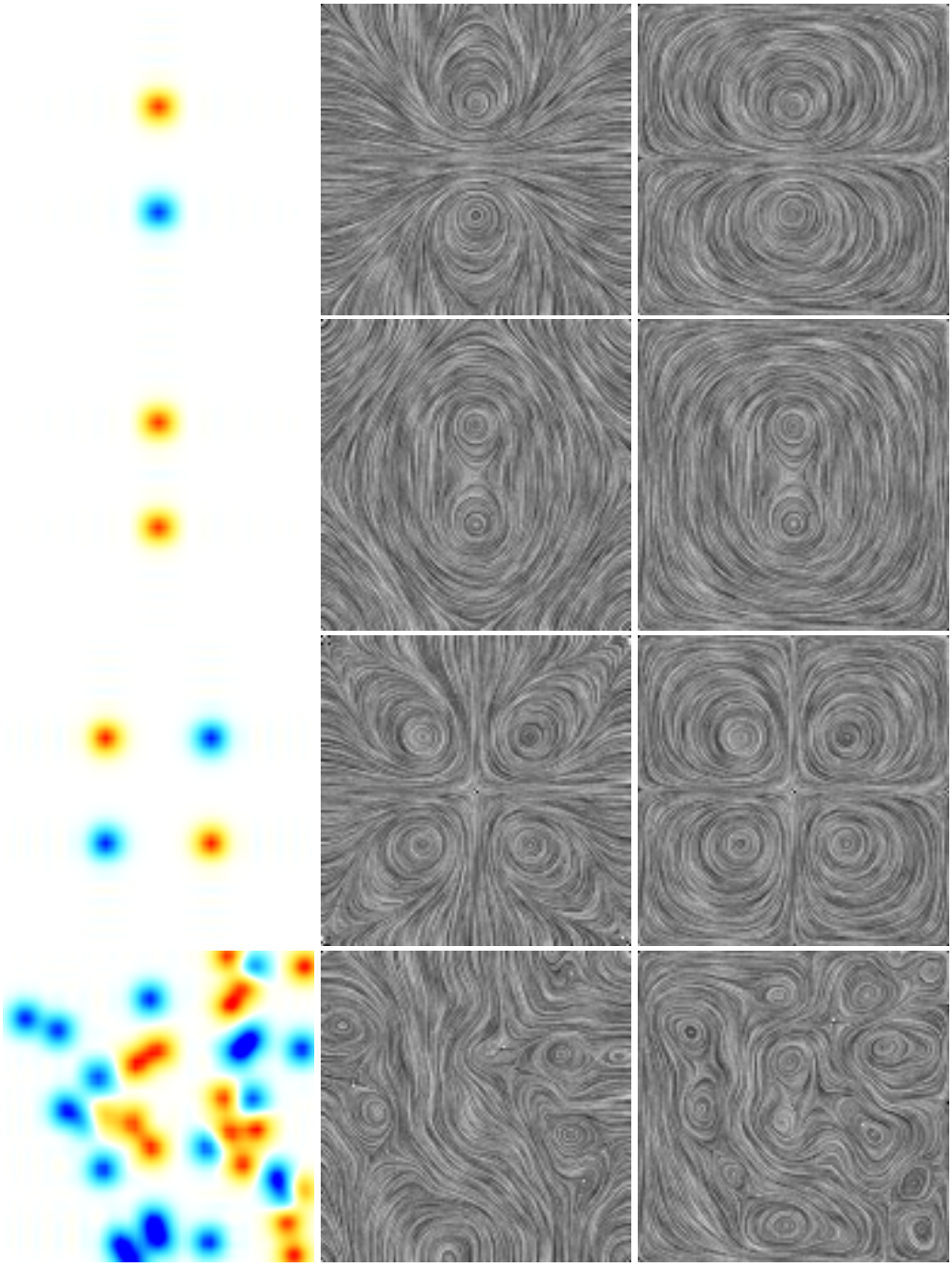


Figure 5.8: Velocity reconstruction from vorticity. The velocity field is expressed as $v = \star\mathbf{d} \star \beta$. The leftmost column displays the vorticity 2-form. The middle column shows the solution for the velocity 1-form on the primal grid with a boundary condition $\star\mathbf{d} \star \beta|_{\mathcal{B}} = 0$. The rightmost column shows the solution on the dual grid with a boundary condition $\star\beta|_{\mathcal{B}} = 0$.

of the equation. Thus, given ω we can solve for β from which we can compute the velocity directly, or formally

$$v = \star\mathbf{d}\star(\mathbf{d}\star\mathbf{d}\star)^{-1}\omega,$$

Figure 5.8 illustrates how the method described here can be used to reconstruct velocity fields from vortices placed at a variety of positions and with a variety of strengths. The vortices are modeled as Gaussian functions with widths that are small relative to the domain size. On the last row, we place many vortices at random locations in order to show that the method can be used with complicated inputs rather than just nicely behaved and symmetric ones.

Diffusion Equation

The diffusion equation (also known as the heat equation)

$$f_t = \Delta f$$

is a parabolic partial differential equation which describes how the density f of a diffusing material changes with time as matter is transported from regions of high concentration to regions of low concentration.

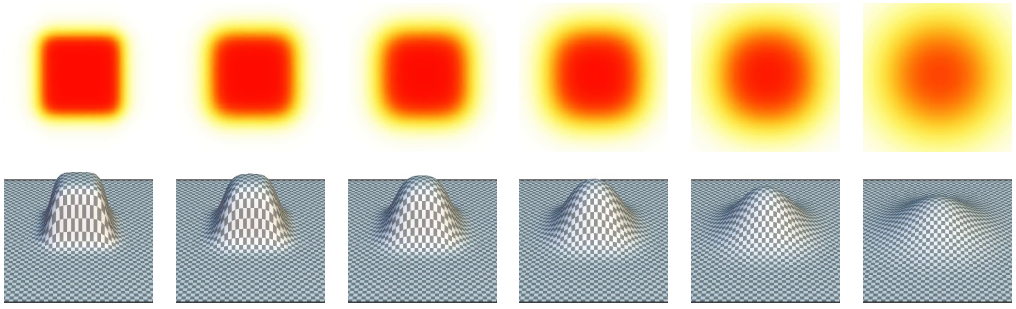


Figure 5.9: Diffusion equation at $t = 0.0, 0.02, 0.04, 0.08, 0.16, 0.32$ with periodic boundary conditions.

We can easily simulate it using our Spex implementation. Figure 5.9 illustrates the diffusion from a region of high concentration under periodic boundary conditions with explicit integration in time. The concentration is shown both as a colormap and as a heightmap. At first the colored part is separated from the clear part by a sharp well-defined boundary, but as time passes the boundary becomes less and less clear with the outside region becoming more intensely colored and the inside region becoming correspondingly less intensely colored until eventually as $t \to \infty$ (not shown on the figure) the whole domain becomes uniformly colored.

Wave Equation

The wave equation

$$f_{tt} = \Delta f$$

is a hyperbolic partial differential equation that governs the evolution of a variety of phenomena in physics such as sound waves, light waves, or water waves.

In its one-dimensional form, the wave equation describes the vibrations of a string. When the string is fastened at each end, this corresponds to Dirichlet boundary conditions, whereas when it is allowed to move freely we have Neumann boundary conditions. To simulate such a string we set up a grid with periodic boundary conditions in the y-direction and periodic, Neumann, and Dirichlet boundary condtions in the x-direction. We place a Guassian function with an initial condition set to its derivative along the x-direction and allow it to evolve forward in time. Because of the symmetry in the y-direction, the problem is essentially a one-dimensional one. The results are illustrated in Figure 5.10. These solutions can be easily derived using classical wave theory, but in this case they have been obtained by running simulations on our Spex framework which provides further validation for its correctness.

We also provide solutions that cannot be derived analytically. Figure 5.11 illustrates a radially symmetric Gaussian placed at the center of the computational domain with stationary initial conditions. Figure 5.12 illustrates the same Gaussian with initial conditions set to the spatial derivative in the x-direction.

Because we have implemented all our operators as tensor products of their one-dimensional counterparts it is trivial to set up separate boundary conditions along each dimension of the computational grid. We show such mixed boundary conditions (Dirichlet along x-direction, Neumann along y-direction) in the last column of Figure 5.13.

When simulating the wave equation, in order to ensure numerical stability, special care must be taken that the Courant-Friedrichs-Lewy (CFL) condition

$$C = \frac{u\Delta t}{\Delta x} \leq C_{\max}$$

is met. The meaning of the variable is that u is the wave velocity, Δt is the size of the time step, and Δx is the size of the grid spacing. For explicit time integrators (as we use here) $C_{\max}$ is usually equal to 1. The condition dictates that the largest stable time step Δt is proportional to the size of the grid cell. As the computational grid is refined to increase its spatial resolution, the impact of the CFL condition is

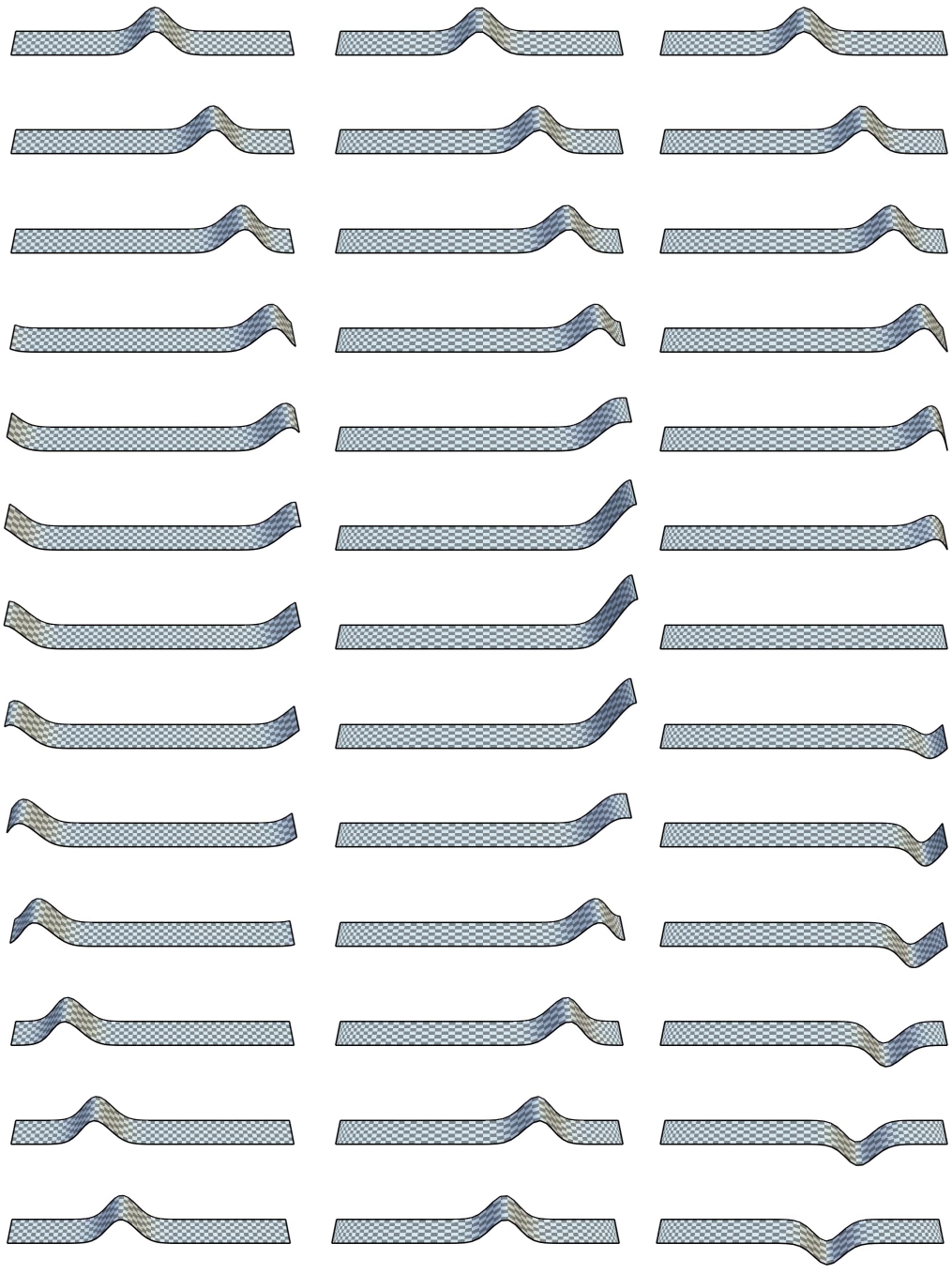


Figure 5.10: One-dimensional wave propagation with periodic, Neumann and Dirichlet boundary conditions on a 64×8 grid at times t = 0.1, 0.2, 0.3, 0.4, 0.45, 0.48, 0.5, 0.52,0.55, 0.6, 0.7, 0.8, 0.9.

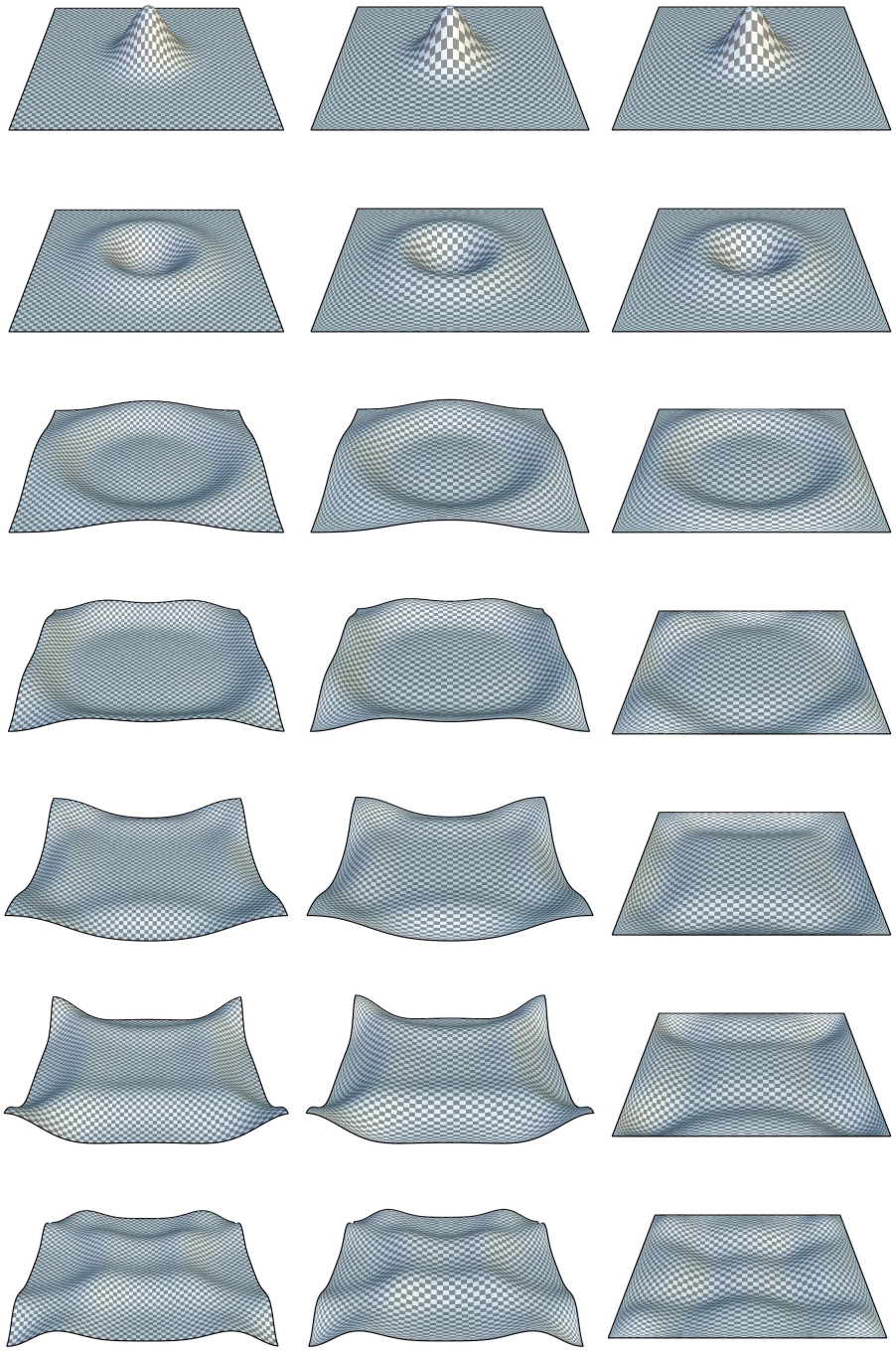


Figure 5.11: Wave propagation with periodic, Neumann, and Dirichlet boundary conditions on a 64×64 grid shown at times $t = 0.0, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8$. The initial condition is set to be a Gaussian function with a time derivative set to zero: $f_t(x, y, t)|_{t=0} = 0$
.

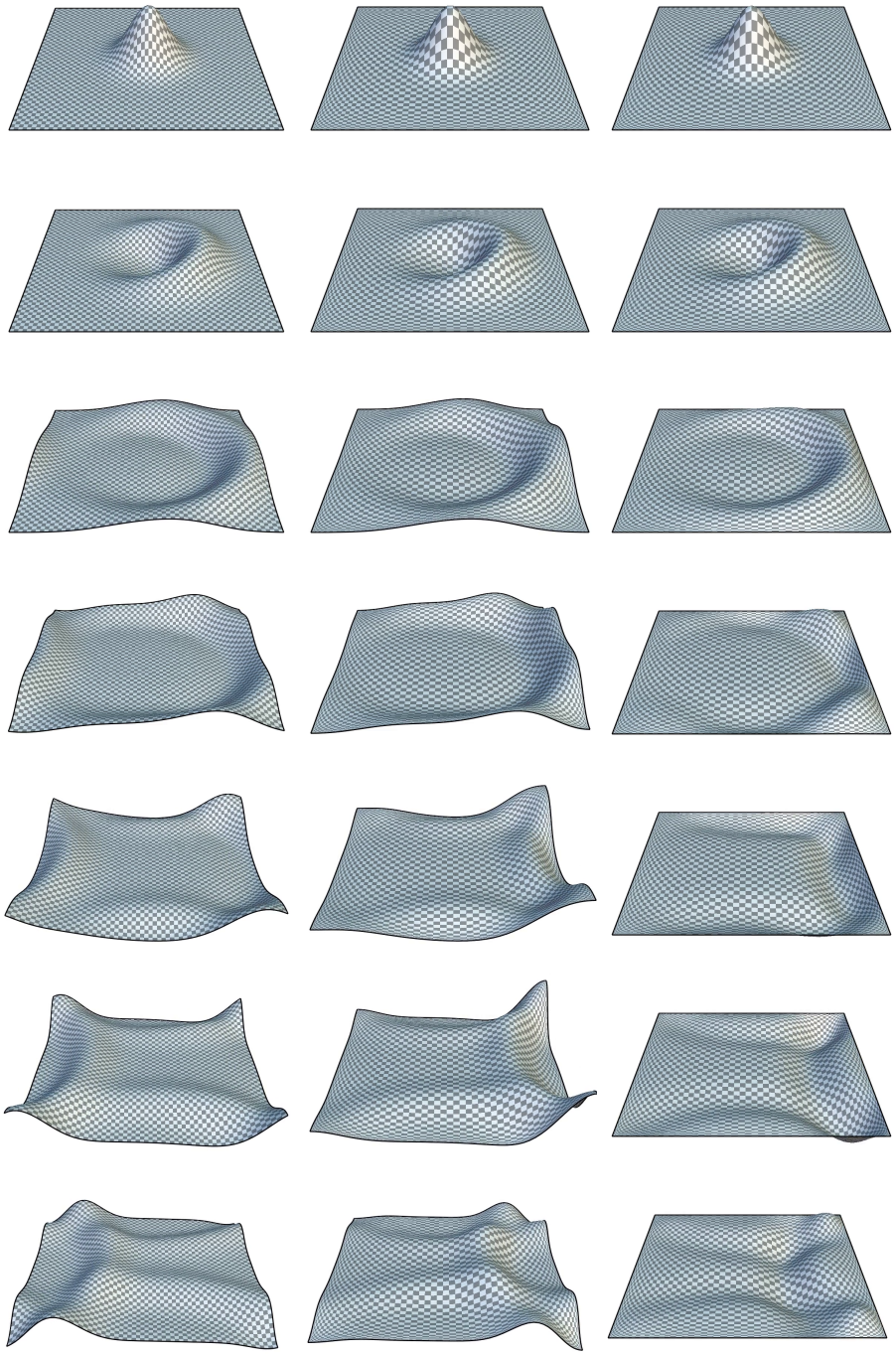


Figure 5.12: Wave propagation with periodic, Neumann and Dirichlet boundary conditions on a 64×64 grid shown at times t = 0.0, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8. The initial condition is set to be a Gaussian function with a time derivative equal to the spacial derivative in the x-direction: $f_t(x, y, t)|_{t=0} = f_x(x, y, 0)$.

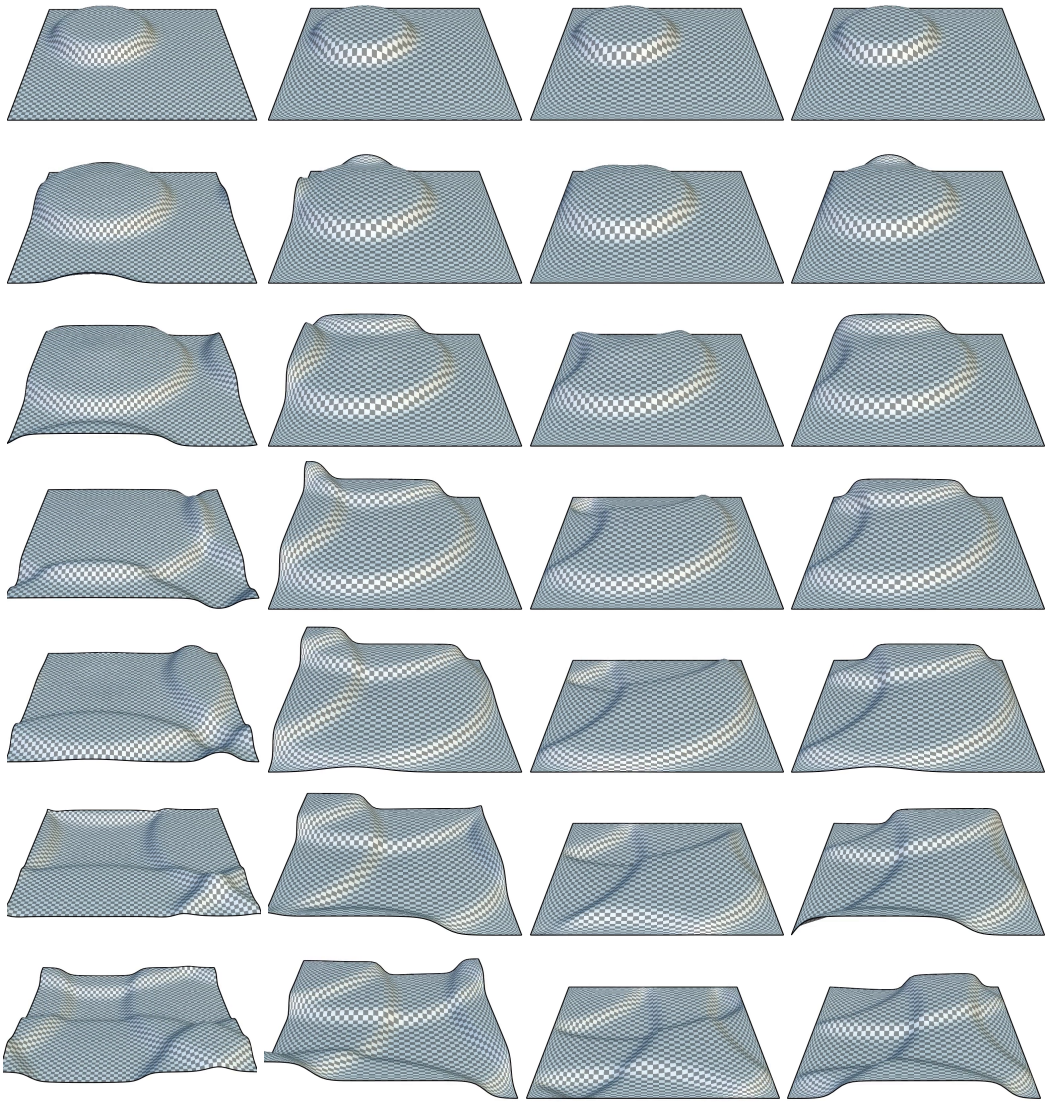


Figure 5.13: Wave propagation with periodic, Neumann, Dirichlet and mixed boundary conditions on a 64×64 grid at $t = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6$ with an initial condition such that the wave propagates radially outward $f_t(x, y, t)|_{t=0} = \left[f_x(x, y, 0)^2 + f_y(x, y, 0)^2\right]^{\frac{1}{2}}$. To aid in visualizing the surface a blue light is shone from the right, and a red light is shone from the left.

to decrease the time step and raise the number of iterations required to maintain a well behaved solution.

Wave Propagation in Curved Space

Motivated by gravitational lensing and light propagation in curved space-time in astrophysics, we now demonstrate how a wave propagates over a curved two-dimensional surface. In order not to distract from the actual geometry of the surface, we plot the wave as a color-map rather than a height-map as we have done in the previous examples.

Figure 5.14 demonstrates the effect of a region of non-zero curvature on the propagation of a flat wave front. We let a flat wave move from left to right over a two-dimensional domain with Neumann boundary conditions. The domain is flat almost everywhere except a localized dip in the center with non-zero curvature. As the wave passes over the dip it becomes distorted. The deeper the dip, the greater the effect on the wave. Although this is a rather simple and naive prototype of gravitational lensing, in principle there should be no difficulty in extending our Spex framework to four-dimensional space-time in order to compute the actual propagation of a light wave around regions of high curvature.

Lie Advection

The Lie advection equation, ubiquitous in many advection phenomena in physics, is given by

$$f_t = -\mathcal{L}_X f,$$

where f is an arbitrary differential k-form that is being advected along the vector field X. A common context for describing advection in numerical physics is the advection of scalar fields, whereas no unified framework exists for advecting non-scalar entities such as vorticity in fluid mechanics. Recent works [47, 29] have shown the benefits of considering the Lie derivative in the context of discrete exterior calculus by leveraging Cartan's magic formula which relies on the exterior derivative $\mathbf{d}$ and the contraction $\lrcorner$. Both of these operators are implemented in Spex and therefore we can define a discrete Lie derivative with ease as their composition.

Figure 5.15 illustrates the results of advecting 1-forms along vector fields, using the discrete Lie derivative of the Spex framework. The top row shows the vector field X along which the 1-form f is being advected, whereas the rows below display snapshots of the advected 1-form f at different times. In the first column, the

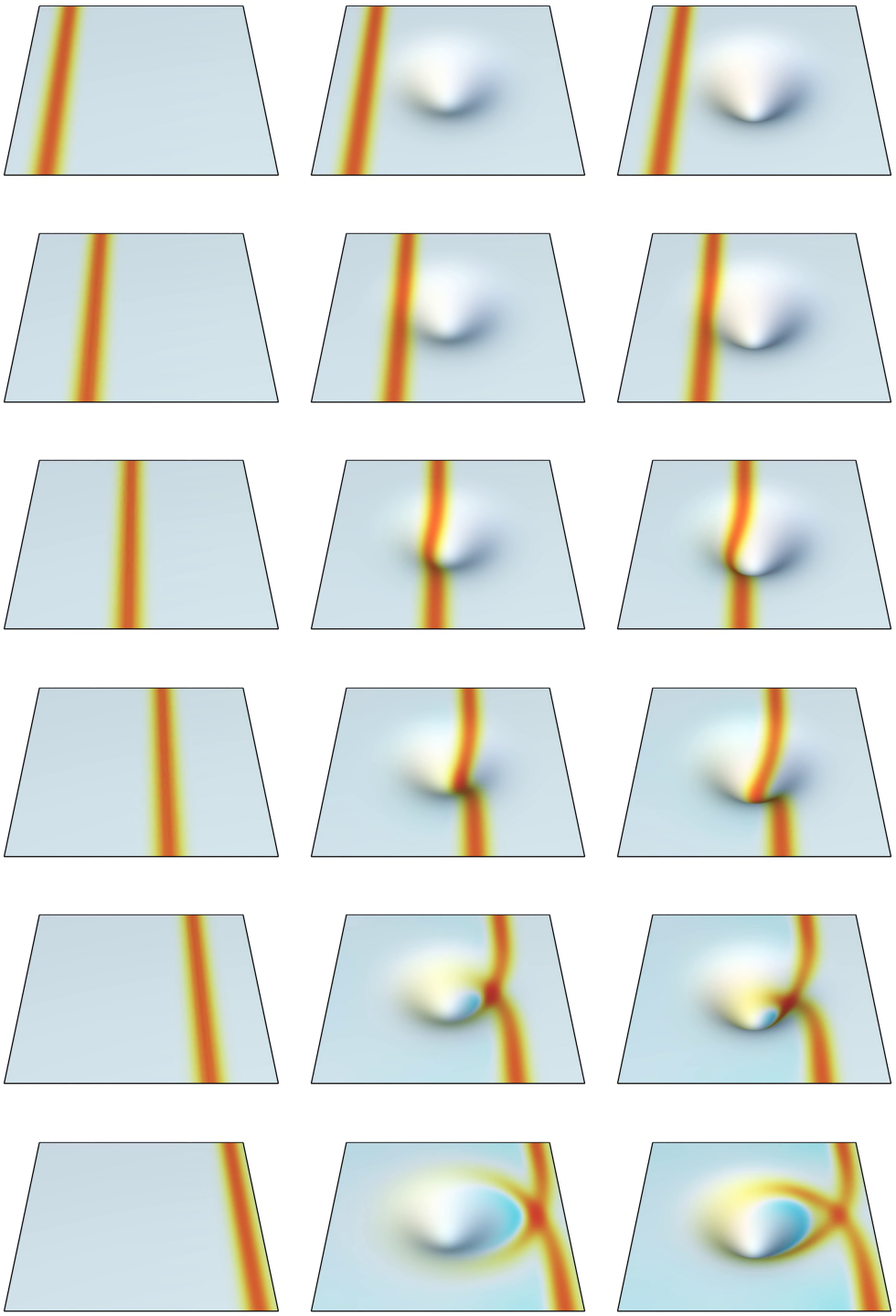


Figure 5.14: Wave equation on a 128×128 Chebyshev grid subject to Neumann boundary conditions. The wave travels from left to right. Snapshots are taken at times $t = 0.15$, 0.30, 0.45, 0.60, 0.75, 0.93. Time integration is explicit with a time step of $\Delta t = 10^{-4}$. The 2D space in which the wave propagates is given by $x(u, v) = u$, $y(u, v) = v$, $z(u, v) = -z_0\, e^{-9(u^2+v^2)}$, where $z_0 = 0, \frac{1}{3}, \frac{1}{2}$ for the first, second, and third column, respectively.

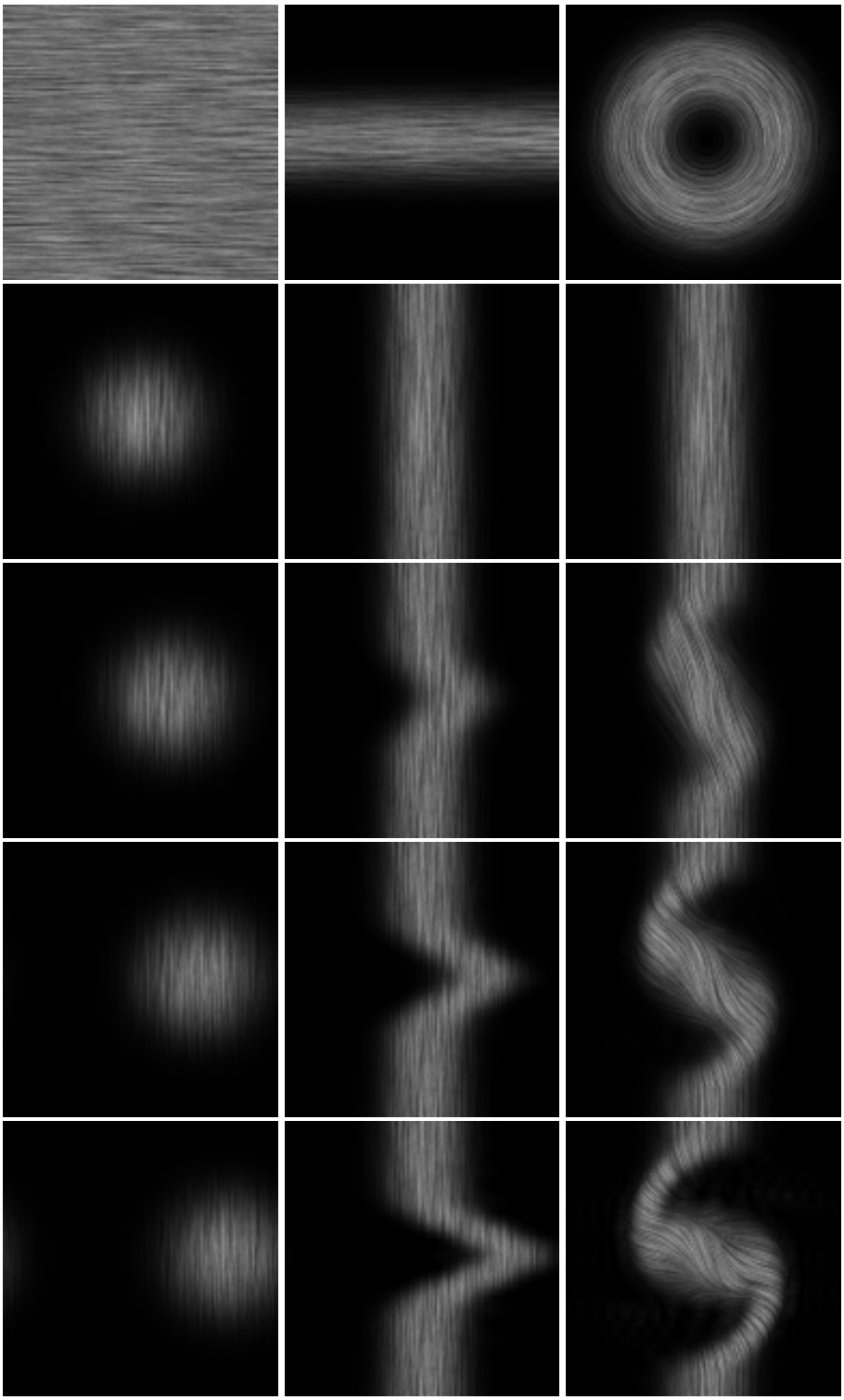


Figure 5.15: Lie advection $f_t = -\mathcal{L}_X f$.

advection is carried along a uniform vector field which results in steady translation along the direction of the field. In the second column, we have a narrow jet which deflects a second vertical jet in the middle. In the last column, we have a circular jet that deflects a vertical jet in mutually opposing directions at each end producing a distortion that looks like the letter $\mathcal{S}$.

5.A Auxilary Operators

In this section we review the auxilary operators used in the implementation.

diff

diff computes the discrete difference of an array.

$$\begin{aligned} \text{diff}: \quad & \mathbb{R}^N \to \mathbb{R}^{N-1} \\ & \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_1 - x_0 \\ x_2 - x_1 \\ \vdots \\ x_{N-1} - x_{N-2} \end{bmatrix} \end{aligned}$$

roll

roll^n rolls an array by n steps. The order is cyclic. Elements that roll beyond the last position are reintroduced at the beginning of the array.

$$\begin{aligned} \text{roll}^n: \quad & \mathbb{R}^N \to \mathbb{R}^N \\ & \begin{bmatrix} x_0 \\ \vdots \\ x_i \\ \vdots \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_{(0-n) \mod N} \\ \vdots \\ x_{(i-n) \mod N} \\ \vdots \\ x_{(N-1-n) \mod N} \end{bmatrix} \end{aligned}$$

slice

A slice extracts elements from an array based on a start index, a stop index and a step. We specify an optional first index, an optional last index, and an optional step.

$$\text{slice}^{\text{start}\,:\,\text{stop}\,:\,\text{step}}$$

The semantics are exactly equivalent to the slice implementation for lists and tuples $a[\text{start} : \text{stop} : \text{step}]$ in the Python programming language. Slice indices have useful

defaults. An omitted first index defaults to zero. An omitted second index defaults to the size of the array that is sliced. An omitted third index defaults to one, and all elements are strided sequentially. A negative step means that the array is traversed in the opposite direction.

even

even picks only the elements at even indices. In terms of slice notation the even function is given by

$$\text{even}(f) := \text{slice}^{0::2}(f).$$

Explicitly, the even function can be expressed as

$$\begin{aligned} \text{even}: \quad & \mathbb{R}^N \to \mathbb{R}^{\lfloor \frac{N}{2} \rfloor} \\ & \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_2 \\ \vdots \\ x_{N-3} \\ x_{N-1} \end{bmatrix}. \end{aligned}$$

odd

odd picks the elements at odd locations

$$\text{odd}(f) := \text{slice}^{1::2}(f),$$

$$\begin{aligned} \text{odd}: \quad & \mathbb{R}^N \to \mathbb{R}^{\lceil \frac{N}{2} \rceil} \\ & \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_3 \\ \vdots \\ x_{N-4} \\ x_{N-2} \end{bmatrix}. \end{aligned}$$

weave

weave mixes two arrays by alternating between their elements:

$$\text{weave}: \quad \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^{2N},$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \times \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ \vdots \\ x_{N-2} \\ y_{N-2} \\ x_{N-1} \\ y_{N-1} \end{bmatrix}.$$

concat

Concatenate two arrays by sequentially joining them end-to-end to create an array of twice the size:

$$\text{concat}: \quad \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^{2N},$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \times \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix}.$$

mid

Pick the middle of an array by removing its endpoints:

$$\text{mid}(f) := \text{slice}^{1:-1:}(f),$$

$$\text{mid}: \quad \mathbb{R}^N \to \mathbb{R}^{N-2}$$
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ \vdots \\ x_{N-2} \end{bmatrix}.$$

rev

Reverse an array:

$$\text{rev}(f) \coloneqq \text{slice}^{::-1}(f),$$

$$\text{rev}: \quad \mathbb{R}^N \to \mathbb{R}^N$$
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_{N-1} \\ x_{N-2} \\ \vdots \\ x_1 \\ x_0 \end{bmatrix}.$$

mirror

Mirror an array so that the resultant array is (anti-)symmetric around its center. The boundary elements can either be kept as singletons $\mathcal{M}_0$, or they can be repeated $\mathcal{M}_1$:

$$\mathcal{M}_0^{\pm}: \quad \mathbb{R}^N \to \mathbb{R}^{2N-2},$$
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ \pm x_{N-2} \\ \vdots \\ \pm x_1 \end{bmatrix},$$

$$\mathcal{M}_1^{\pm}: \quad \mathbb{R}^N \to \mathbb{R}^{2N},$$
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ \pm x_{N-1} \\ \pm x_{N-2} \\ \vdots \\ \pm x_1 \\ \pm x_0. \end{bmatrix}$$

The left inverses of the above operators are

$$\mathcal{M}_0^\dagger : \quad \mathbb{R}^N \to \mathbb{R}^{\frac{N}{2}+1}, \qquad \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \\ \vdots \\ x_{2N-1} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix},$$

$$\mathcal{M}_1^\dagger : \quad \mathbb{R}^N \to \mathbb{R}^{\frac{N}{2}}, \qquad \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \\ \vdots \\ x_{2N} \end{bmatrix} \mapsto \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}.$$

5.B Boundary Conditions for Laplace's Equation

In this section, we illustrate in detail how to solve Laplace's equation with different boundary conditions.

0-forms For 0-forms, we can pick either Dirichlet or Neumann boundary conditions by choosing where to solve the discrete equation - on the primal or the dual grid. Note that the terms Dirichlet and Neumann are meaningful only for 0- forms. There equivalents for 1-forms will be described later on.

Dirichlet

$$f \in \bar{\Lambda}^0, \quad b \in \mathbb{B}^0,$$

$$H\widetilde{\mathbf{D}}^1 H\bar{\mathbf{D}}_b^0 f = 0,$$

$$H\widetilde{\mathbf{D}}^1 H(\bar{\mathbf{D}}^0 f + \mathbb{D}^0 b) = 0.$$

Solve for f:

$$H\widetilde{\mathbf{D}}^1 H\bar{\mathbf{D}}^0 f = -H\widetilde{\mathbf{D}}^1 H\mathbb{D}^0 b,$$

corresponding to the boundary condition

$$f\,|_{\mathcal{B}} = b.$$

Neumann

$$f \in \widetilde{\Lambda}^0, \quad \beta \in \mathbb{B}^1,$$

$$H\bar{\mathbf{D}}_\beta^1 H\widetilde{\mathbf{D}}^0 f = 0,$$

$$H[\bar{\mathbf{D}}^1 (H\widetilde{\mathbf{D}}^0 f) + \mathbb{D}^1 \beta] = 0.$$

Solve for f:

$$H\bar{\mathbf{D}}^1 H\widetilde{\mathbf{D}}^0 f = -H\mathbb{D}^1\beta,$$

corresponding to the boundary condition

$$(\star\mathbf{d}f)\,|_{\mathcal{B}} = \beta.$$

1**-forms** For 1-forms, on the other hand, we will have a richer set of boundary conditions.

Primal

$$\alpha \in \bar{\Lambda}^1, \quad b \in \mathbb{B}^0, \quad \beta \in \mathbb{B}^1,$$

$$(\bar{\mathbf{D}}^0_b H\widetilde{\mathbf{D}}^1 H + H\widetilde{\mathbf{D}}^0 H\bar{\mathbf{D}}^1_\beta)\alpha = 0.$$

Solve for α:

$$(\bar{\mathbf{D}}^0 H\widetilde{\mathbf{D}}^1 H + H\widetilde{\mathbf{D}}^0 H\bar{\mathbf{D}}^1)\alpha = -\mathbb{D}^0 b - H\widetilde{\mathbf{D}}^0 H\mathbb{D}^1\beta,$$

corresponding to the boundary condition

$$(\star\mathbf{d}\star\alpha)\,|_{\mathcal{B}} = b \qquad \alpha\,|_{\mathcal{B}} = \beta.$$

Dual

$$\alpha \in \widetilde{\Lambda}^1, \quad b \in \mathbb{B}^0, \quad \beta \in \mathbb{B}^1,$$

$$(H\bar{\mathbf{D}}^0_b H\widetilde{\mathbf{D}}^1 + \widetilde{\mathbf{D}}^0 H\bar{\mathbf{D}}^1_\beta H)\alpha = 0.$$

Solve for α:

$$(H\bar{\mathbf{D}}^0 H\widetilde{\mathbf{D}}^1 + \widetilde{\mathbf{D}}^0 H\bar{\mathbf{D}}^1 H)\alpha = -H\mathbb{D}^0 b - \widetilde{\mathbf{D}}^0 H\mathbb{D}^1\beta,$$

corresponding to the boundary condition

$$(\star\mathbf{d}\alpha)\,|_{\mathcal{B}} = b, \qquad (\star\alpha)\,|_{\mathcal{B}} = \beta.$$

Figure 5.4 illustrates the four possibilities for $\bar{\alpha}$ and $\widetilde{\alpha}$ with $(b,\beta) = \{(0,1),(1,0)\}$.

Chapter 6

PROGRAMMING CONTRIBUTIONS

Over the course of this thesis a number of libraries have been developed that have been made freely available at github.com/drufat under an open source "copyleft" license. The objective has been to enable third parties to easily reproduce and replicate the results in this work while avoiding unnecessary duplication of effort. Too often nowadays, one is faced with the task of recreating existing libraries from the ground up instead of building on top of what has already been created by others. What follows is an overview of the various programming contributions that have been made both as standalone packages and as additions to existing third party libraries.

6.1 Spexy

The Python module Spexy is the reference implementation of *spectral exterior calculus*. It provides a number of classes for differential forms, both in their discrete and continuous representations, with all of the associated geometric operators implemented as methods and attributes to those classes. Additionally, classes are provided for periodic and bounded grids that discretize the abstract surfaces on which the aforementioned forms reside.

Three differential form classes are provided in the `spexy.form` module

- `sym.Form` - forms in their symbolic representation,
- `coch.Form` - forms as cochains,
- `comp.Form` - forms as pointwise components,

along with the ability to seamlessly map between them. Each of the differential form classes implements the standard set of exterior calculus operators, sometimes directly and other times by transforming into a more convenient representation in which the operator is easier to implement.

Given an instance `f` of a differential form class, we can access its derivative as `f.D`, its Hodge star as `f.H`, the wedge product of two forms `f0` and `f1` as `f0.W(f1)`, the inner product of `f0` and `f1` as `f0.I(f1)`, and the contraction of `f` with a vector `v` as

`v.C(f)`. For convenience, the wedge product and the contraction are also provided via the built-in binary operators `^` and `|` respectively. Thus, `v^f` is an alias for `v.W(f)`, and `v|f` for `v.C(f)`.

The grid classes, on the other hand, contain data about the discretization of the computational domain, more specifically about its representation as a *cellular complex*. They provide a complete description of the positions and relative arrangement of the cells. The grid classes can be imported from the `spexy.grid` module. Currently only one- and two-dimensional grids are supported with the prospect of extending to higher dimensions in the future.

- `Grid_1D` - one-dimensional grids
- `Grid_2D` - two-dimensional grids

In this work we deal with two main types of one-dimensional grids – periodic and bounded ones. Higher-dimensional grids are easily formed by taking Cartesian products of the one-dimensional ones, and different combinations along each dimension are possible. The code below demonstrates how to instantiate grid objects:

```
from spexy.grid import Grid_1D, Grid_2D

g1 = Grid_1D.chebyshev(10)
g2 = Grid_2D.periodic(10, 10)
g3 = Grid_1D.periodic(10) * Grid_1D.chebyshev(12)
```

In the example above `g1` represents a one-dimensional bounded (Chebyshev) grid which has 10 primal edges. `g2` is an instance of a two-dimensional 10×10 periodic grid. `g3` demonstrates the possibility of mixing different grid types by taking their Cartesian product along each direction - periodic along the x-axis, and bounded along the y-axis.

Every grid object stores a reference to its dual which can be accessed via `g.dual`. Since the dual of a dual is the identity, `g.dual.dual` becomes a circular reference to the original grid `g`.

Having described the grid and form classes provided by Spexy, we now turn our attention to how these two main classes interact with each other. Given a symbolic form `fsym` we can project it onto a grid `g` via the reduction operator `fcoch = fsym.P(g)` to obtain a cochain form `fcoch` associated with the cells of the grid `g`. Conversely, given a cochain form `fcoch`, we can reconstruct the original form via

the reconstruction operator `fcoch.R`. The upsampling operator can be accessed via `fcomp = fcoch.Pup` to output a component form `fcomp`, whereas given a component form `fcomp` one can downsample it via `fcoch = fcomp.Pdown(g)` or `fcoch = fcomp.Pdown(g.dual)` onto a primal or a dual cochain form `fcoch` respectively.

The form class encapsulates the degree of the differential form, which for a form instance `f` can be accessed by `f.degree`. The degree is required in order to choose the correct operator. For example, in two dimensions one can choose from multiple Hodge star operators $\{\mathbf{H}^0, \mathbf{H}^1, \mathbf{H}^2\}$ depending on the degree of the input form. With the degree encapsulated in the form data type, rather than the programmer having to manually choose the operator matching the degree, `f.H` automatically selects the correct operator, and assigns the correct degree to the output form. In this way the user is spared the mundane and error prone task of degree tracking. This convenience is provided for all of the Spex operators. The degree computations are done in a manner that is consistent with the mathematical definition of each operator, i.e. for any form `f` the assertions below are always true:

```
assert f.D.degree     == f.degree + 1        # Derivative
assert f.H.degree     == ndim - f.degree     # Hodge star
assert (f ^ g).degree == f.degree + g.degree # Wedge product
```

where `ndim` is the dimension of the computational domain (typically 1 or 2).

Additionally, cochain forms also encapsulate the grid with which they are associated. Given a form `f`, its grid can be accessed via `f.grid`. With respect to the grid, the following assertions hold:

```
assert f.D.grid == f.grid        # Derivative
assert f.H.grid == f.grid.dual   # Hodge star
```

These assertions reflect the fact that when acting on cochain forms, the derivative leaves the grid unchanged, whereas the Hodge star flips to the dual grid.

Given these basic Spex operators, one can implement composite ones such as the Laplacian and the Lie derivative. For example, the Laplacian can be implemented as:

```
def lap(f):
    return f.H.D.H.D + f.D.H.D.H
```

Whereas, the Lie derivative is:

```
def lie(v, f):
    return v.C(f.D) + v.C(f).D
```

Examples

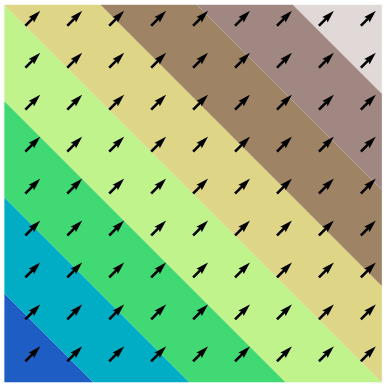
Let us illustrate what kind of computations one can do with the Spexy library. In order to successfully execute the examples below in the Python interpreter, we need to import a few modules and initialize a number of variables as below:

```
import sympy as sy
from spexy.form import sym
from spexy.grid import Grid_2D

x, y = sy.symbols('x, y')
F0, F1, F2 = sym.Form.forms(x, y)

g = Grid_2D.chebyshev(8, 8)
```

First, we import the symbolic form module `sym`. The module attribute `sym.Form.forms` provides convenient initializers `F0`, `F1`, and `F2` that make it easy to instantiate zero-, one-, and two-dimensional symbolic forms respectively. Finally, on the last line we instantiate an 8×8 Chebyshev grid, and we are ready to do actual computations with forms on that grid.

Gradient First, we compute the derivative of the zero-form $x + y$.

```
fsym = F0(x + y)

f0 = fsym.P(g)
bc = fsym.Dbc(g)

f1 = f0.D + bc

plotf(f0)
plotf(f1, color='black')
```

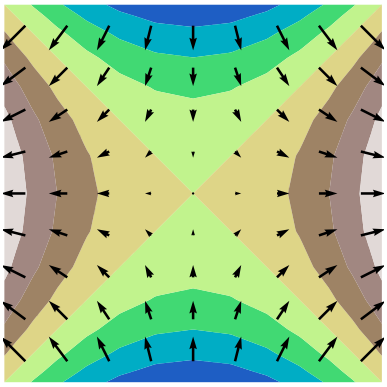


We instantiate the zero-form into the variable `fsym`, then project it onto the primal grid. Because the primal grid has associated boundary vertices, the derivative operator will require an additive boundary condition term `bc`, which can be computed using the attribute `Dbc`. The boundary condition term is obtained by sampling the original form at the boundary vertices. On the next line, we assign the result to

the `f1` variable. Finally, we use the `plotf` function to visualize the results. We plot one-forms indirectly by converting them to vector fields via the sharp operator, which in the case of a flat surface is trivial.

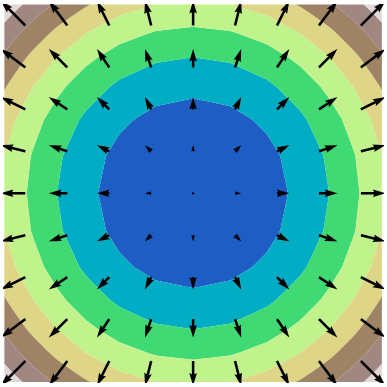
In the next example we compute the derivative of the zero-form $x^2 - y^2$, but instead of projecting onto the primal grid, we project onto the dual. Because the dual grid has no boundary vertices, there will be no associated boundary condition term.

```
f0 = F0(x ** 2 - y ** 2).P(g.dual)
f1 = f0.D

plotf(f0)
plotf(f1, color='black')
```



The next example is identical to the previous one, except we compute the derivative of $x^2 + y^2$

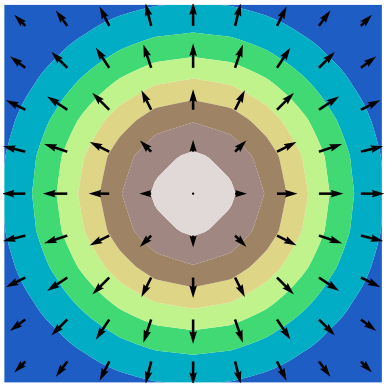
```
f0 = F0(x ** 2 + y ** 2).P(g.dual)
f1 = f0.D

plotf(f0)
plotf(f1, color='black')
```



As expected, the derivative of a zero-form corresponds to the gradients of the a scalar field.

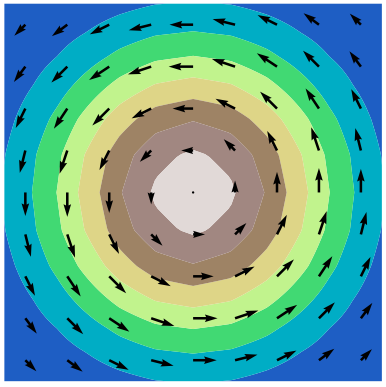
Divergence The divergence of a vector field can be computed by taking the $\star\mathbf{d}\star$ of its corresponding one-form. The example below demonstrates how to compute the divergence of the vector field corresponding to the one-form $A(x\,\mathbf{d}x + y\,\mathbf{d}y)$. The coefficient $A = e^{-(x^2+y^2)}$ ensures the field is localized near the center and decays to zero away from it.

```
A = sy.exp(-(x ** 2 + y ** 2))
f1 = F1(A*x, A*y).P(g)
f0 = f1.H.D.H

plotf(f0)
plotf(f1, color='black')
```



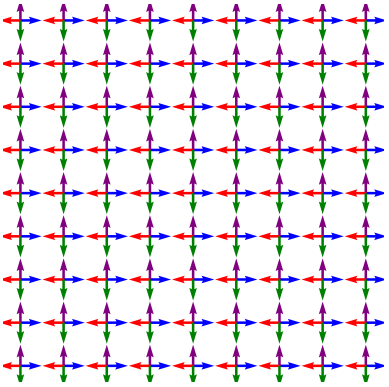
Curl The curl can be computed by applying the $\star\mathbf{d}$ operator to the corresponding one-form, and in the example below we do the computation for $A(-y\,\mathbf{d}x + x\,\mathbf{d}y)$

```
A = sy.exp(-(x ** 2 + y ** 2))
f1 = F1(-A*y, A*x).P(g.dual)
f0 = f1.D.H

plotf(f0)
plotf(f1, color='black')
```

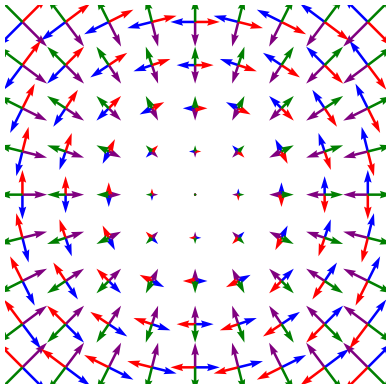


Hodge star On a flat 2D surface, the Hodge star of a one-form corresponds to an anti-clockwise rotation by 90°. The two examples below illustrate that by applying $\star$ multiple times we can achieve such rotations.

```
f = F1(1, 0).P(g)

plotf(f, color='blue')
plotf(f.H, color='purple')
plotf(f.H.H, color='red')
plotf(f.H.H.H, color='green')
```

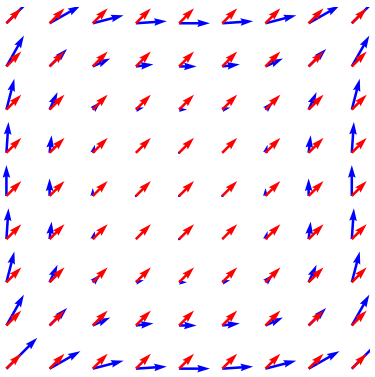


```
f = F1(-y, x).P(g)

plotf(f, color='blue')
plotf(f.H, color='purple')
plotf(f.H.H, color='red')
plotf(f.H.H.H, color='green')
```



Laplacian The Laplacian is a composite operator that can be expressed as $\star\mathbf{d}\star\mathbf{d} + \mathbf{d}\star\mathbf{d}\star$. In the example below, we demonstrate how to compute the Laplacian of the one-form $y^2\,\mathbf{d}x + x^2\,\mathbf{d}y$. Special care must be taken to include the boundary terms when applying the `D` operator on primal forms. Otherwise, one would get incorrect results near the boundary.

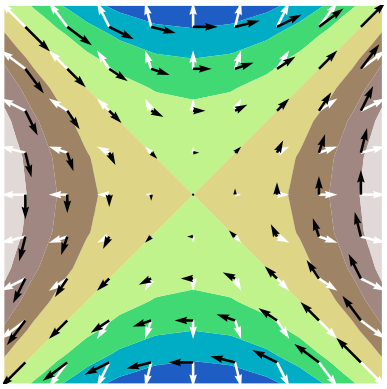
```
fsym = F1(y**2, x**2)

f   = fsym.P(g)
bc0 = fsym.H.D.H.Dbc(g)
bc1 = fsym.Dbc(g)

fl = (f.H.D.H.D + bc0) + (f.D + bc1).H

plotf(f, color='blue')
plotf(fl, color='red')
```



Because the Laplacian is a second derivative, we expect the result of a quadratic function to be a constant field, which is indeed the case.

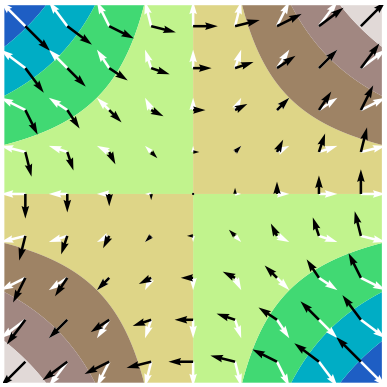
Wedge product Taking the wedge product of two one-forms is like taking the cross product of their corresponding vector fields. The resulting two-form, which is converted to a zero-form before plotting, is proportional to the sine of the angle between the two fields.

```
f1 = F1(x, y).P(g)
f2 = F1(y, x).P(g)

f3 = f1 ^ f2

plotf(f3.H)
plotf(f1, color='white')
plotf(f2, color='black')
```



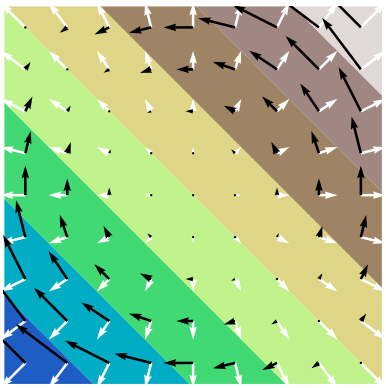
Contraction Let `fv` be a one-form, and `fk` a k-form. Below we illustrates the contraction $\texttt{fv}^\sharp \lrcorner \texttt{fk}$ of `fk` with the vector field corresponding to `fv`. The contraction of a one-form `f1` with $\texttt{fv}^\sharp$ is equivalent to the inner product between `f1` and `fv`, and thus the resulting scalar field must be proportional to the cosine of the angle between the two fields. The plot below demonstrates that is indeed the case.

```
fv = F1(x, y).P(g)
f1 = F1(y, x).P(g)

f0 = fv | f1

plotf(f0)
plotf(fv, color='white')
plotf(f1, color='black')
```



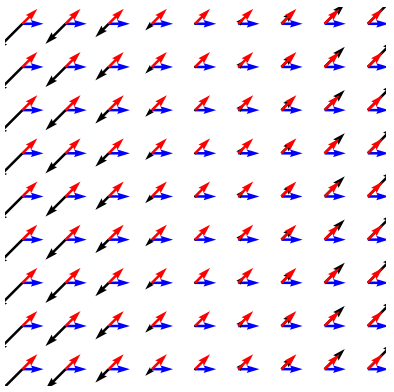
The next programming snippet shows the contraction of a two-form `f2` with $\texttt{fv}^\sharp$.

```
fv = F1(x, y).P(g)
f2 = F2(x + y).P(g)

f1 = fv | f2

plotf(f2.H) #convert to 0-form
plotf(fv, color='white')
plotf(f1, color='black')
```



Lie derivative By relying on Cartan's magic formula we can also compute the Lie derivative.

```
fv = F1(1, 0).P(g.dual)
f1 = F1(x, x).P(g.dual)

f2 = fv.C(f1.D) + fv.C(f1).D

plotf(fv, color='blue')
plotf(f1, color='black')
plotf(f2, color='red')
```



The derivative of the black field is computed along the blue field and the result is plotted as the red field. The result does indeed appear to be equal to the rate of change of the input black field along the chosen direction.

6.2 LicPy

There are a number of techniques to image vector fields. The most common one, known as hedgehog, is to draw an arrow at each point with a heading and a size matching the direction and magnitude of the vector field at that particular point (e.g. `quiver` in Matplotlib). Recently Cabral and Leedom proposed a new method to image vector fields called *line integral convolution* (LIC) [16]. LicPy is our implementation in Python of the LIC method.

The method requires one to take a random noisy background texture, and for each pixel of the texture, streamlines of equal length are computed along the input vector field. One then integrates the background pixel values along each streamline to produce a single pixel value and a corresponding output texture. The output looks like a blurred version of the input, where the blurring is along the direction of the vector field that is being visualized, and is easy for the human eye to interpret.

After generating the random texture, one needs to ensure that the input vector field and the random texture are of the same size, i.e. for each texture pixel there must be a single corresponding vector field value. If that is not the case, for example, if the vector field is sampled less densely, one can easily interpolate in a piecewise linear fashion to compute a value per texture pixel. Then, given a point (marked with x in Figure 6.1) one computes a stream-line going through that point both in the forward (as on the figure) and backward directions.

This is easy to do since the vector field is piecewise constant within each pixel, so the streamlines are straight line segments within each pixel. Computing them is merely a matter of figuring out where the straight line intersects the rectangular boundaries of the pixel. Once the streamlines are computed, the random background texture colors are averaged over their length, resulting in a new color for each pixel.

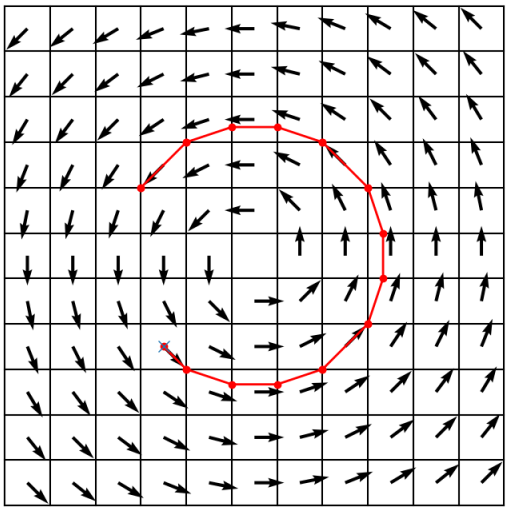


Figure 6.1: Streamline

The output image produced in this fashion is more intuitive to the human eye to interpret because it looks like the smearing of ink along a moving fluid. Figure 6.2 illustrates a number of vector fields that are visualized using both quiver plots and LIC plots. Notice how the LIC plots remain well-behaved even in the presence of singularities near which the quiver plots are hard to parse.

6.3 PyBindCpp

PyBindCpp is a Python module that we have developed in order to make it easy to interface Python with C++. In this section we describe the approach taken by PyBindCpp, and compare it to other similar libraries.

Python has been widely successful as a programming language for scientific computations primarily due to two factors. First of all, as a high level language it is highly readable and expressive, which allows for rapid development. Second of all, because of its open source nature and the access to its internal CPython[1] *application programming interface* (API) it is possible to interface Python with the vast collection of existing scientific libraries without having to reimplement them from the ground up. However, despite these advantages, Python has one important drawback. Although adequate for most applications where speed is not a critical requirement, because of its highly dynamic nature, Python tends to be slow, especially when dealing with tight for-loops, and it is not suitable for computations where such low-level looping constructs are necessary.

[1] The reference implementation of the Python programming language is done in C and is known as CPython. It can be downloaded from python.org.

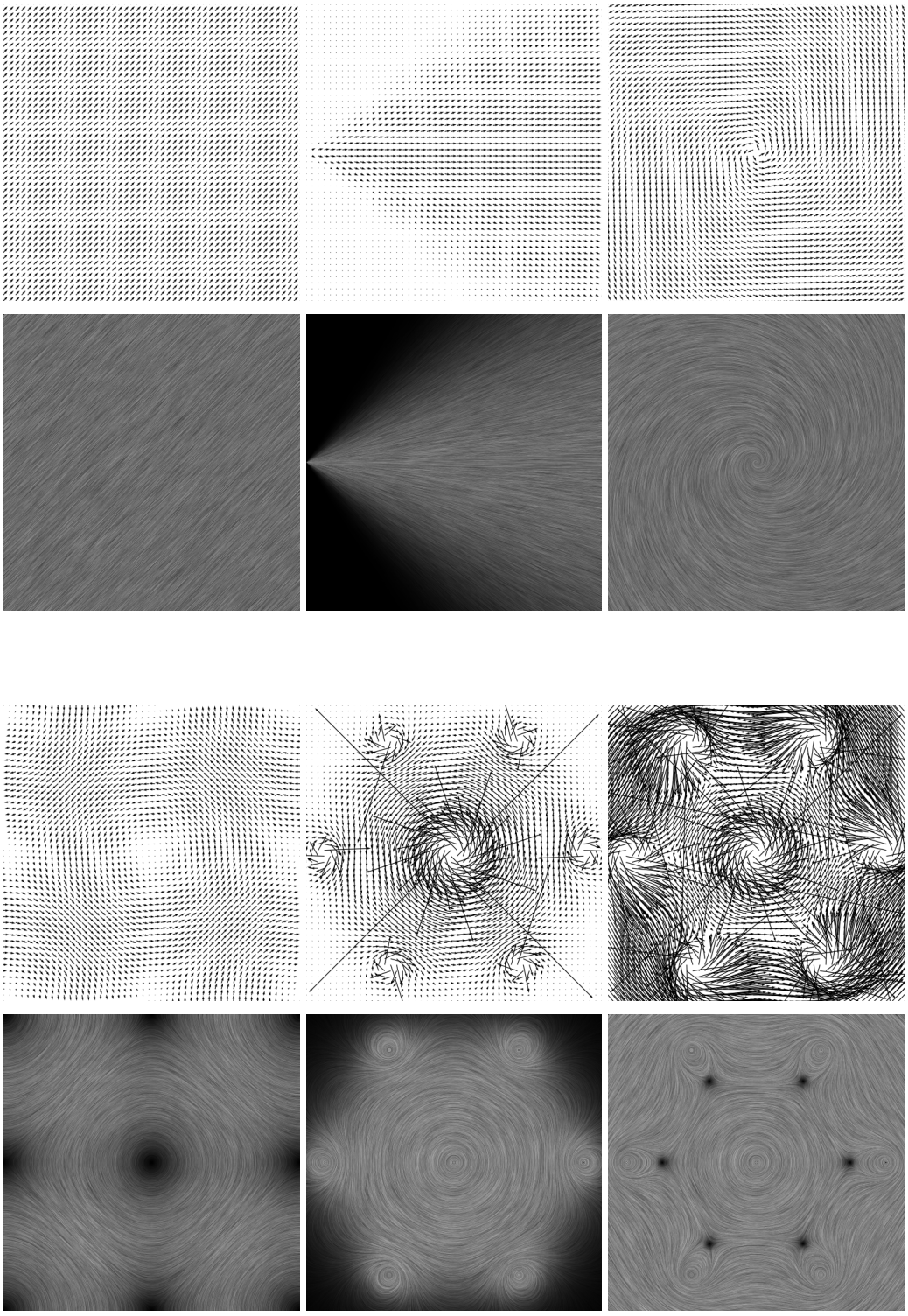


Figure 6.2: Comparison of quiver plots generated using Matplotlib with line integral convolution plots generated by LicPy.

To ameliorate this problem, a number of approaches have been proposed for a hybrid programming model where most of the high level logic is written in Python, and the time critical parts of the application, which usually tend to be a small fraction of the code base, are reimplemented in a low level language, significantly speeding up the application running time.

Among the most popular approaches is the one taken by Cython[6]. In fact, Cython is not merely a library, but an entire programming language, which is a strict superset of the Python language. This means that all Python programs are valid Cython programs, but the converse is not true. Cython adds type annotations to the Python language, where each variable can be optionally assigned a static type. This extra information allows for the source code to be translated directly into C, and compiled and linked against Python, and then imported as a module. While the approach taken by Cython is quite powerful, it does have its drawbacks. First, it seems to be mainly geared towards speeding up existing Python applications by adding type annotations to the critical parts, and not towards porting existing C/C++ libraries. Interfacing with external libraries is quite cumbersome as one has to copy all of the header file declarations into a form that can be parsed by Cython. This violates the "don't repeat yourself" (DRY) principle where every piece of knowledge must have a single unambiguous representation within the system. In the case of Cython, however, declarations must exist both in the C/C++ header files as well as within Cython files. Additionally, just to bridge Python and C/C++ one has to learn a completely new third programming language with its new syntax and semantics.

Another approach for speeding up Python is to write the low level glue code manually in C/C++. This requires intimate knowledge of the CPython API as well as the addition of a large amount of boilerplate code for manual data type conversion and memory management. While this gives a lot of control to the programmer, it also places limitations on productivity due to the time consuming nature of programming at such a low level. Additionally, this approach intimately depends on the internals of the original Python implementation, and is not portable to other implementations of the same programming language such as PyPy, Jython, or IronPython.

A third approach is to use the Ctypes module that is included by default in the Python interpreter. Ctypes is a foreign function library for Python that provides C compatible data types that can be used to call functions directly from compiled DLL or shared library files. Ctypes works at the *application binary interface* (ABI) level rather than the *application programming interface* (API) level. One still needs

to provide descriptions in Python of the exact signatures of the functions that are being exported. In fact one has to be quite careful, because at the ABI level if there is any mismatch, one can get quite nasty segmentation faults and data corruption errors, whereas the worst that can happen at the API level is for the program not to compile. Again, like the approach taken by Cython, this violates the DRY principle since declarations must be copied into Python.

PyBindCpp is our own proposed solution to the problem which leverages Ctypes and C++14 in order to provide highly portable and seamless interoperability between C/C++ and Python. Most of the logic is implemented in Python via the Ctypes module which enables one to call functions from shared library files and do the appropriate type conversions. The key insight is to use C++14 for type deductions, and to feed that information back to Python in order to generate the appropriate Ctypes wrappers. In order to deduce the types of the variables and the signatures of the functions, we have relied heavily on recent features added to C++ such as template metaprogramming, type traits, parameter packs, and the auto specifier. Because of this, the minimum version of C++ necessary to compile PyBindCpp is C++14. Without these features, the only way to obtain the type and signature information of functions would have been to parse C++, which is a very difficult problem given the size and complexity of the language.

In designing PyBindCpp, we have been inspired by ChaiScrit[70], and we have borrowed heavily from its design. ChaiScript is a scripting language that targets C++ directly and takes advantage of its many modern features. The typesystem of ChaiScript is identical to the C++ type system with a few minor additions. Unlike ChaiScript, PyBindCpp is not a new language, but a module of Python, and it provides its own type conversion layer between C++ data types and Python data types. Like ChaiScript, PyBindCCpp is *headers only* and does not require any external tools or libraries to compile, just a modern C++ compiler.

PyBind11[38] is another library that is very similar in its design goals to PyBindCpp. However, there are some significant differences. Most of the programming logic in PyBind11 is coded in C++ and the library relies heavily on the CPython API making it difficult to port to other implementations of the Python language. PyBindCpp, on the other hand, uses Ctypes which is provided by most Python interpreters, and can therefore be easily ported. A design goal of PyBindCpp is to minimize reliance on the API to a bare minimum, perhaps even eliminate it completely, and instead leverage the ABI via Ctypes. Additionally, PyBindCpp is coded mainly in Python,

and C++ is minimally used for type deduction. Many of the Python functions are made available to the C++ level via Ctypes callbacks. This saves us the effort of having to code them in C++, especially for code that is rarely executed (e.g., only once at import time) where speed is not that critical. This small speed penalty is more than offset by the simplicity of the code when written in Python.

Next we present a minimal working example of PyBindCpp usage. Consider the following C++ code which includes variables and functions that we wish to export to Python:

```
#include <cmath>

constexpr double half = 0.5;
constexpr double pi = M_PI;

int
f(int N, int n, int x)
{
  return N + n + x;
}

double
mycos(double x)
{
  return cos(x);
}
```

To port the above code, we merely need to write a single function before compiling into a shared library that can be directly imported from Python. We must define a function which takes as its input a reference to an `ExtModule` object, and uses that to register the entities that need to exported. We can use the `var` attribute to register simple types, and the `fun` attribute to register functions. Finally, we must add the `PYMODULE_INIT` preprocessor macro which takes two arguments - the name of the module that we are creating and the name of the function that we just defined. This is necessary in order for Python to find the correct entry point when loading the extension module.

```
#include <pybindcpp/module.h>

using namespace pybindcpp;

void
module(ExtModule& m)
{
  m.var("half", half);
  m.var("pi", pi);
```

```
  m.var("one", (long)1);
  m.var("two", (unsigned long)2);
  m.var("true", true);
  m.var("false", false);
  m.var("name", "pybindcpp");

  m.fun("f", f);
  m.fun("mycos", mycos);
  m.fun("cos", [](double x) -> double { return cos(x); });
  m.fun("sin", static_cast<double (*)(double)>(sin));
}

PYMODULE_INIT(example, module)
```

After compiling the code above into a shared module, we can use the script below to import it from Python and run some tests to ensure that everything works as expected.

```
import math
import example as m

def test_example():
    assert round(m.pi, 2) == 3.14
    assert m.half == 0.5
    assert m.one == 1
    assert m.true == True
    assert m.false == False
    assert m.name == b'pybindcpp'

    assert m.f(1, 2, 3) == 1 + 2 + 3
    assert m.mycos(0.1) == math.cos(0.1)
    assert m.cos(0.1) == math.cos(0.1)
    assert m.sin(0.1) == math.sin(0.1)

if __name__ == '__main__':
    test_example()
```

6.4 Third Party Modules

Additional code contributions made during the course of our graduate studies in the form of patches and pull requests have been submitted to third party modules that are maintained by others. The most significant additions have been made to the Python libraries GlumPy[61] and SymPy[46].

GlumPy is a library for creating scientific visualizations. It provides an object oriented interface to OpenGL and even a novice not acquainted with all the intricate details of the OpenGL API can use GlumPy to generate advanced 2D and 3D graphics and animations. The module comes packaged with many examples, along

with complete vertex and fragment shaders, that allow the user to quickly get up to speed with the capabilities of the library. Our contributions have been to port the library to Python version 3, to simplify the animation interface, to fix minor bugs, and to add extra examples. We use GlumPy primarily for drawing scalar- and vector-fields on non-flat surfaces.

SymPy is a library for symbolic computations and manipulations. Its capabilities include basic arithmetic operations, expression simplification, symbolic integration and differentiation, taking limits, and much more. An important functionality for our purposes is the ability to "lambdify" symbolic expressions, meaning they can be converted to programming functions that are directly callable from code to get numerical results. Our contributions have centered around expanding the types of expressions that can be lambdified to include sums, piecewise expressions, and relational operators. The lambdify method is crucial for us in order to convert symbolic differential forms to their numerical counterparts. Throughout this work we have relied heavily on SymPy for computations with symbolic differential forms, and to verify that basis functions associated with grid cells do indeed satisfy the properties that are required of them.

Additionally, although no contributions have been made to them, in this work we have extensively used NumPy[71] and Matplotlib[37]. NumPy is a Python module that deals with multi-dimensional arrays, and provides a large number of fast operators on those arrays, including many mathematical functions from linear algebra. Matplotlib, on the other hand, is a plotting library which can be used for line, contour, scatter, and vector plots, and has been helpful in generating many of the convergence graphs and mesh visualizations in this work.

All of the third party packages mentioned so far are required dependencies of the Spexy module, and without them the work on implementing *spectral exterior calculus* would have been much more laborious and involved. The author is grateful for the availability of the aforementioned packages as open source libraries. Access to the source code has enabled us to learn from the work of others, to easily modify the software when necessary, to port to newer versions of Python, and to submit improvements to the authors for inclusion in future releases. In return, by releasing all our modules under a permissive open source license we hope to give back to the community, and to similarly benefit from contributions and new functionality by others.

Chapter 7

CONCLUSION

In this chapter we review the main contributions of this thesis followed by a discussion of possible future avenues for extending the present work.

7.1 Review of Contributions

The main contribution of this thesis has been to describe and implement a complete framework of *spectral exterior calculus* (SPEX) that encompasses the entire hierarchy of differential geometry operators ($\mathbf{d}$, $\star$, $\wedge$, $\lrcorner$, Δ, $\mathcal{L}$, $\cdot$, $\flat$, $\sharp$).

We have introduced a twin representation for discrete differential forms, both as cochains and as pointwise components along with the ability to seamlessly transform between them. The cochain representation is appropriate for topological operators such as the derivative, whereas the component representation is more adapted for operators that require multiplication, including the metric dependent ones.

The framework is general enough to work on any domain with logically rectangular boundaries that can have any arbitrary metric. Numerous structural identities from the continuous world are preserved in the computational realm, such as Stokes' theorem, Cartan's magic formula, or involutivity of the Hodge star.

Boundary conditions are straightforward to incorporate as additive terms to the discrete derivative operator. Multiple boundary conditions as well as their combinations have been implemented and demonstrated.

Guarantees are provided for spectral convergence of all the operators, all of which are computed efficiently and implemented in the most appropriate form using the fast Fourier transform with a time complexity of $\mathcal{O}(N \log N)$.

A reference implementation is provided in the Python programming language as a self-contained open source library. The code has been extensively unit tested. It has been initially set up in 1D, and then extended trivially to 2D through tensor products. In principle, there should be no difficulty in extending it to even higher-dimensional spaces like 3D space or the 4D space-time of general relativity with the Minkowski metric.

7.2 Future Work

There are numerous avenues that one can pursue in order to extend the work presented in this document.

Domain Decomposition

In this thesis we have focused on trivial topologies given by our periodic and bounded domains. However, general manifolds can have far richer topologies, and to be able to adequately capture their structure we need a method to combine multiple simple rectilinear domains into more complicated surfaces by patching them together. The theory of domain decomposition [45] is already well established in numerical analysis. It describes a way to split a computational grid into multiple domains, that while coupled, can each be simulated separately in order to better leverage parallel computer architectures. While we want to achieve the opposite, combine computational domains into more complicated ones rather than split existing ones, the basic results still apply. We can adapt domain decomposition to our present framework and implement all the operators of Spex on a general manifold. The individual domains that form the discrete manifold will still be fairly independent of each other with the coupling given by the boundary term of the $\mathbf{d}$ operator for each domain. For example, if A and B are neighboring domains, then the derivative operator on A, denoted by $\mathbf{d}_A$, will have a boundary term due to domain B, and vice versa for $\mathbf{d}_B$. All other operators will be completely independent of each other, and information will be exchanged between the domains only via the boundary term of the derivative operator.

Infinite and Semi-infinite Domains

In his book on numerical spectral methods [11], Boyd describes infinite and semi-infinite domains and their associated basis functions, in addition to the periodic and bounded domains considered here. We can use those basis functions in order to extend the Spex framework to those domains as well. Many simulations in fluid mechanics are actually implemented in exactly such infinite domains [66].

Spectral Connection

On a Euclidian space one can compare vectors at different points directly, but on a general manifold one needs to introduce the connection in order to compare vectors at different tangent spaces. There have been various works proposing structure-preserving approaches to disretization of connections [19], but those have been

low order. Adapting the present work to a discrete connection and its associated covariant derivative that converges spectrally may prove to be a fruitful undertaking. It would also allow us to spectrally compute the curvature and torsion 2-forms for any discrete computational domain.

SPEX on the GPU

All of the SPEX operators are ideally suited to be implemented on parallel computer architectures. As heterogeneous computer architectures become more prevalent, it may be useful to port the present SPEX implementation to the GPU in order to speed up its execution and enable realtime simulations with instant feedback for the developer rather than relying on long-running computations on the CPU.

Appendix A

APPENDIX

A.1 Invariance of the Discrete Wedge Product

Consider a manifold $\mathcal{M}$ and its polyhedrization in the context of discrete exterior calculus, and let σ_i denote the simplices and ϕ_i the corresponding basis functions, such that the pairing given by integration is natural:

$$\langle \sigma_i, \phi_j \rangle \equiv \int_{\sigma_i} \phi_j = \delta_{ij}.$$

Given the reduction

$$\begin{aligned} \mathcal{P}: \quad & \Lambda^k \to \bar{\Lambda}^k \\ & \alpha \mapsto \bar{\alpha}_i = \int_{\bar{\sigma}_i} \alpha \end{aligned}$$

and reconstruction operators

$$\begin{aligned} \mathcal{R}: \quad & \bar{\Lambda}^k \to \Lambda^k \\ & \bar{\alpha}_i \mapsto \alpha = \bar{\alpha}_i \phi^i \end{aligned}$$

the discrete wedge product is given by

$$\bar{\gamma} = \mathbf{W}(\bar{\alpha}, \bar{\beta}) = \mathcal{P}\left((\mathcal{R}\bar{\alpha}) \wedge \left(\mathcal{R}\bar{\beta}\right)\right)$$

$$\bar{\gamma}_i = \sum_{jk} \mathbf{W}_{ijk} \bar{\alpha}_j \bar{\beta}_k,$$

where

$$\mathbf{W}_{ijk} = \left[\mathbf{W}(\phi_j, \phi_k)\right]_i = \int_{\sigma_i} \phi_j \wedge \phi_k.$$

Consider now a second manifold $\mathcal{M}'$ and a diffeomorphism

$$\varphi: \quad \mathcal{M} \to \mathcal{M}',$$

which induces a new polyderization with the corresponding simplices $\sigma' = \varphi_* \sigma$ and basis functions $\phi' = \varphi_* \phi$. Given a simplex σ and a form ϕ, the following identities hold for push-forwards and pull-backs:

$$\int_{\varphi_* \sigma} \varphi_* \phi = \int_{\sigma} \phi, \tag{A.1}$$

$$\varphi_*(\alpha \wedge \beta) = (\varphi_*\alpha) \wedge (\varphi_*\beta). \tag{A.2}$$

The discrete wedge product on this new manifold will be *elementwise equal* to the wedge product on the original manifold. To see why this is the case:

$$\begin{aligned}
\mathbf{W}'_{ijk} &= \int_{\sigma'_i} \phi'_j \wedge \phi'_k \\
&= \int_{\varphi_*\sigma_i} \left(\varphi_*\phi_j\right) \wedge \left(\varphi_*\phi_k\right) \\
&= \int_{\varphi_*\sigma_i} \varphi_* \left(\phi_j \wedge \phi_k\right) \quad \text{(eq.}A.2\text{)} \\
&= \int_{\sigma_i} \phi_j \wedge \phi_k \quad \text{(eq.}A.1\text{)} \\
&= \mathbf{W}_{ijk}.
\end{aligned}$$

So we have shown that the discrete wedge product is indeed invariant under diffeomorphisms:

$$\mathbf{W}'_{ijk} = \mathbf{W}_{ijk}.$$

This result is consistent with the theory in differential geometry, where the wedge product is thought of as a purely topological operator that is independent of the metric.

A.2 Involutivity of the Discrete Hodge Star Operator

In the continuous case, the Hodge star is its own inverse (up to a sign):

$$\star^{n-k} \star^{k} = \star^{k} \star^{n-k} = (-1)^{k(n-k)}\, \mathbf{id}, \tag{A.3}$$

where **id** is the identity map. We would like to mirror this property in the discrete setting. For that, the discrete mesh (representing a discrete polyhedrization of the continuous manifold) must meet certain requirements. We want to study under what conditions the discrete Hodge star operator satisfies the discrete counterpart of Eq. (A.3):

$$\widetilde{\mathbf{H}}\mathbf{H} = \mathbf{H}\widetilde{\mathbf{H}} = (-1)^{k(n-k)}\, \mathbf{Id}, \tag{A.4}$$

where $\mathbf{H} : \bar{\Lambda}^k \to \widetilde{\Lambda}^{n-k}$ and $\widetilde{\mathbf{H}} : \widetilde{\Lambda}^{n-k} \to \bar{\Lambda}^k$, i.e. under what conditions is it an exact involution. The pairing between simplices and bases functions is given by integration

$$\langle \sigma_i, \phi_j \rangle \equiv \int_{\sigma_i} \phi_j \in \mathbb{R}.$$

The cardinal basis functions ϕ associated with the simplices σ must by definition satisfy

$$\langle \sigma_i, \phi_j \rangle = \delta_{ij} = \langle \widetilde{\sigma}_j, \widetilde{\phi}_i \rangle.$$

The elements of the discrete Hodge star matrix acting on primal ($\mathbf{H}$) and dual ($\widetilde{\mathbf{H}}$) forms are given respectively by:

$$\begin{aligned} \mathbf{H} &= \widetilde{\mathcal{P}} \star \mathcal{R}, & \widetilde{\mathbf{H}} &= \mathcal{P} \star \widetilde{\mathcal{R}}, \\ H_{ij} &= \langle \widetilde{\sigma}_i, \star \phi_j \rangle, & \widetilde{H}_{ij} &= \langle \sigma_i, \star \widetilde{\phi}_j \rangle. \end{aligned}$$

These are the definitions of the discrete Hodge star operators, and in the general case it is not necessary that $\widetilde{\mathbf{H}}$ is an exact inverse of $\mathbf{H}$ as mandated by Eq. (A.4). In order to see when condition Eq. (A.4) holds, let us construct the dual basis functions as linear combinations of the primal basis functions, namely

$$\star \widetilde{\phi}_i = \sum_j A_{ij} \phi_j, \tag{A.5}$$

where the matrix $\mathbf{A}$ represents the transformation between the two basis sets. The Hodge star ($\star$) is present in the equation in order for the dual basis functions to have the correct degree. If ϕ is the basis function for k-forms, then $\widetilde{\phi}$ must be the basis function for $(n-k)$-forms, and therefore only $\star\widetilde{\phi}$ can be a linear combination of ϕ. It turns out that the matrix $\mathbf{A}$ is equal to the transpose of $\widetilde{\mathbf{H}}$:

$$\mathbf{A}^T = \widetilde{\mathbf{H}}.$$

To see this,

$$\begin{aligned} \widetilde{H}_{ij} &= \langle \sigma_i, \star \widetilde{\phi}_j \rangle \\ &= \sum_\alpha A_{j\alpha} \langle \sigma_i, \widetilde{\phi}_\alpha \rangle \\ &= \sum_\alpha A_{j\alpha} \delta_{i\alpha} \\ &= A_{ji}. \end{aligned}$$

Thus, Eq. (A.4) becomes $\mathbf{A}^T\mathbf{H} = \mathbf{H}\mathbf{A}^T = (-1)^{k(n-k)}\mathbf{Id}$ and since $\mathbf{H}$ is of full rank, this implies that $\mathbf{A} = (-1)^{k(n-k)}\mathbf{H}^{-T}$. Since $\mathbf{H}$ only depends on the the primal basis functions ϕ and the dual simplices $\widetilde{\sigma}$, then they completely determine $\mathbf{A}$ and through Eq. (A.5) they completely determine the dual basis functions. In terms of the discrete Hodge star equation Eq. (A.5) and its dual become

$$\star\widetilde{\phi}_i = \sum_j \phi_j \widetilde{H}_{ji}, \qquad \star\phi_i = \sum_j \widetilde{\phi}_j H_{ji}.$$

In summary, in order for the discrete Hodge star to have an exact inverse, the basis functions (ϕ and $\widetilde{\phi}$) and simplices (σ and $\widetilde{\sigma}$) must satisfy the following constraints simultaneously:

$$\star\,\widetilde{\phi}_i = \sum_j \phi_j \langle \sigma_j, \star\widetilde{\phi}_i \rangle \qquad \star\,\phi_i = \sum_j \widetilde{\phi}_j \,\langle \widetilde{\sigma}_j, \star\phi_i \rangle. \tag{A.6}$$

If the basis functions and their corresponding simplices (or cells for a regular mesh) satisfy the constraints given by Eq. (A.6), the discrete Hodge star will be its own exact inverse. The basis functions for the periodic grid and the bounded grid described in this work satisfy this requirement.

A.3 Types of Hodge Star Matrices

The discrete Hodge star operator transforms k-forms on the primal mesh into $(n-k)$-forms on the dual mesh, and vice-versa. It is completely defined by the basis functions ϕ and the simplices (or cells) σ, and its matrix elements are given by

$$\bar{\mathbf{H}}_{ij} = \int_{\widetilde{\sigma}_i} \star\bar{\phi}^j \qquad \widetilde{\mathbf{H}}_{ij} = \int_{\bar{\sigma}_i} \star\widetilde{\phi}^j.$$

In this section we briefly review the types of Hodge star matrices categorized by their corresponding basis functions. The reader interested in more information may refer to, e.g.,[8, 2, 5, 67, 72].

Diagonal

The diagonal Hodge star is commonly used because of its simplicity. Its basis functions are piecewise constant (i.e. $\phi \in \mathcal{C}^0$) and only have constant non-zero values over their corresponding simplices. These basis functions are discontinuous across cell boundaries where their value abruptly changes to zero. The Hodge star, as implied by its name, is a diagonal matrix whose diagonal elements are given by

$$\bar{\mathbf{H}}^k_{ii} = \frac{|\widetilde{\sigma}_i^{n-k}|}{|\bar{\sigma}_i{}^k|}, \qquad \widetilde{\mathbf{H}}^k_{ii} = \frac{|\bar{\sigma}_i^{n-k}|}{|\widetilde{\sigma}_i^k|}.$$

From the above equations it is evident that this simple Hodge star is trivially its own involution (see A.2). Because of the low order of its basis function, it has the lowest convergence rate, but approaches have been proposed to reduce its error by optimizing the underlying geometry of the simplices [47].

Whitney

Given piecewise linear basis functions associated with vertices ϕ_i, one can define the whole set of basis functions associated with a simplex of any degree using a

trick first introduced by Whitney [73]:

$$\begin{aligned}
\phi_i^0 &= \phi_i, \\
\phi_{ij}^1 &= \phi_i \, \mathbf{d}\phi_j - \phi_j \, \mathbf{d}\phi_i, \\
\phi_{ijk}^2 &= 2\big(\phi_i \, \mathbf{d}\phi_j \wedge \mathbf{d}\phi_k + \phi_j \, \mathbf{d}\phi_k \wedge \mathbf{d}\phi_i + \phi_k \, \mathbf{d}\phi_i \wedge \mathbf{d}\phi_j\big).
\end{aligned}$$

The basis functions above are associated with the simplices σ_i^0, σ_{ij}^1, and σ_{ijk}^2 respectively, and they result in piecewise linear functions $\phi \in \mathcal{C}^1$. The Hodge star matrix associated with those functions is sparse. In the 1D case it has a very narrow band along the diagonal. However, this Hodge star matrix is not its own involution in the exact sense, but only in the approximate sense as the mesh size h tends to zero, $h \to 0$.

Spectral

The spectral basis functions are smooth $\phi \in \mathcal{C}^\infty$, and they span the entire computational domain. Therefore, the Hodge star operator is no longer a sparse matrix, but a dense one. However, it is a special kind of dense matrix. On the periodic grid $\mathbf{H}$ is a circulant matrix, whereas the one on the Chebyshev grid is a centrosymmetric matrix. For both of these cases the Hodge star is its own exact involution. Below we explore some other properties of such matrices.

Circulant The $N \times N$ circulant matrices are defined by

$$\mathcal{C} = \big\{ C \,\big|\, C_{ij} = c_{(i-j) \mod N} \big\}$$

or in matrix form

$$C = \begin{pmatrix}
c_1 & c_N & \cdots & c_3 & c_2 \\
c_2 & c_1 & \cdots & c_4 & c_3 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
c_{N-1} & c_{N-2} & \cdots & c_1 & c_N \\
c_N & c_{N-1} & \cdots & c_2 & c_1
\end{pmatrix}.$$

The set of matrices $\mathcal{C}$ forms a group. The proof is left as an exercise to the reader.

Centrosymmetric The centrosymmetric matrices are those $N \times N$ matrices that satisfy the property

$$\mathcal{H} = \big\{ H \,\big|\, H_{ij} = H_{N+1-i,N+1-j}, \det H \neq 0 \big\}$$

or in matrix forms

$$H = \begin{pmatrix} a_1 & a_2 & \cdots & a_{N-1} & a_N \\ b_1 & b_2 & \cdots & b_{N-1} & b_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_N & b_{N-1} & \cdots & b_2 & b_1 \\ a_N & a_{N-1} & \cdots & a_2 & a_1 \end{pmatrix}.$$

Let

$$J = \begin{bmatrix} . & . & \dots & . & 1 \\ . & . & \dots & 1 & . \\ & & & & \\ . & 1 & \dots & . & . \\ 1 & . & \dots & . & . \end{bmatrix} \qquad J_{ij} = \begin{cases} 1 & \text{if } i = N - j + 1 \\ 0 & \text{otherwise} \end{cases}.$$

The matrix J is sometimes referred to as the exchange matrix.

Using the exchange matrix the above set of matrices can also be conveniently defined as

$$\mathcal{H} = \{A \,|\, AJ = JA\}.$$

Proposition. The set of matrices $\mathcal{H}$ forms a group.

Proof. It can be easily shown that the identity is in the group, i.e. $I \in \mathcal{H}$. Also the set is closed under multiplication, i.e. if $A \in \mathcal{H}$ and $B \in \mathcal{H}$ then $AB \in \mathcal{H}$. To see why, consider

$$ABJ = AJB = JAB.$$

If $A \in \mathcal{H}$, then $AJ = JA$,

$$(AJ)^{-1} = (JA)^{-1},$$

$$J^{-1}A^{-1} = A^{-1}J^{-1},$$

$$JA^{-1} = A^{-1}J.$$

Therefore, $A^{-1} \in \mathcal{H}$. This completes the proof that $\mathcal{H}$ forms a group. *QED*

A.4 In Coordinates

In this section we describe some of the exterior calculus operators in coordinates.

Wedge Product

Consider the **2D** case. The wedge product between a 0-form and a 1-form is

$$\begin{aligned}\alpha^0 &= \alpha,\\ \beta^1 &= \beta_x dx + \beta_y dy,\\ \alpha^0 \wedge \beta^1 &= \alpha\beta_x dx + \alpha\beta_y dy.\end{aligned}$$

Between two 1-forms it is

$$\begin{aligned}\alpha^1 &= \alpha_x dx + \alpha_y dy,\\ \beta^1 &= \beta_x dx + \beta_y dy,\\ \alpha^1 \wedge \beta^1 &= \left(\alpha_x\beta_y - \beta_x\alpha_y\right) dx \wedge dy.\end{aligned}$$

In the **3D** case the wedge product between a 0-form and a 1-form is

$$\begin{aligned}\alpha^0 &= \alpha,\\ \beta^1 &= \beta_x dx + \beta_y dy + \beta_z dz,\\ \alpha^0 \wedge \beta^1 &= \alpha\beta_x dx + \alpha\beta_y dy + \alpha\beta_z dz.\end{aligned}$$

Between two 1-forms it is

$$\begin{aligned}\alpha^1 =& \alpha_x dx + \alpha_y dy + \alpha_z dz,\\ \beta^1 =& \beta_x dx + \beta_y dy + \beta_z dz,\\ \alpha^1 \wedge \beta^1 =& (\alpha_x\beta_y - \alpha_y\beta_x) dx \wedge dy +\\ &+ (\alpha_y\beta_z - \alpha_z\beta_y) dy \wedge dz +\\ &+ (\alpha_z\beta_x - \alpha_x\beta_z) dz \wedge dx.\end{aligned}$$

Between a 1-form and a 2-form it is

$$\begin{aligned}\alpha^1 &= \alpha_x dx + \alpha_y dy + \alpha_z dz,\\ \beta^2 &= \beta_{xy} dx \wedge dy + \beta_{yz} dy \wedge dz + \beta_{zx} dz \wedge dx,\\ \alpha^1 \wedge \beta^2 &= (\alpha_x\beta_{yz} + \alpha_y\beta_{zx} + \alpha_z\beta_{xy}) dx \wedge dy \wedge dz.\end{aligned}$$

Contraction

Given a manifold $\mathcal{M}$, the interior product is defined as the contraction of a differential form with a vector field:

$$\lrcorner : \quad \mathfrak{X} \times \Lambda^k \to \Lambda^{k-1}.$$

The contraction of a 1-form $\alpha = \alpha_i e^i$ with a vector field $X = X^i e_i$ is given by

$$\begin{aligned} X \lrcorner \alpha &= \langle X, \alpha \rangle \\ &= \alpha_i X^j \langle e^i, e_j \rangle \\ &= \alpha_i X^j \delta^i_j \\ &= \alpha_i X^i. \end{aligned}$$

The contraction of a 2-form $\beta = \beta_{ij} e^i \wedge e^j$ with the same vector field is given by

$$\begin{aligned} X \lrcorner \beta &= \beta_{ij} X \lrcorner (e^i \wedge e^j) \\ &= \beta_{ij} \left((X \lrcorner e^i) \wedge e^j - e^i \wedge (X \lrcorner e^j) \right) \\ &= \beta_{ij} \left(X^i e^j - X^j e^i \right). \end{aligned}$$

The contraction of a 3-form $\gamma = \gamma_{ijk} e^i \wedge e^j \wedge e^k$ is going to be

$$\begin{aligned} X \lrcorner \gamma &= \gamma_{ijk} X \lrcorner (e^i \wedge e^j \wedge e^k) \\ &= \gamma_{ijk} \left(\left(X \lrcorner e^i \right) \wedge e^j \wedge e^k - e^i \wedge \left(X \lrcorner e^j \right) \wedge e^k + e^i \wedge e^j \wedge \left(X \lrcorner e^k \right) \right) \\ &= \gamma_{ijk} \left(X^i e^j \wedge e^k - X^j e^i \wedge e^k + X^k e^i \wedge e^j \right). \end{aligned}$$

In **2D** the explicit formulas for the contractions are

$$\begin{aligned} \alpha &= \alpha_x dx + \alpha_y dy, \\ X \lrcorner \alpha &= X^x \alpha_x + X^y \alpha_y, \end{aligned}$$

$$\begin{aligned} \beta &= \beta_{xy}\, dx \wedge dy, \\ X \lrcorner \beta &= -\beta_{xy} X^y dx + \beta_{xy} X^x dy. \end{aligned}$$

If **3D**, on the other hand, the contractions become

$$\begin{aligned} \alpha &= \alpha_x dx + \alpha_y dy + \alpha_z dz \\ X \lrcorner \alpha &= X^x \alpha_x + X^y \alpha_y + X^z \alpha_z, \end{aligned}$$

$$\begin{aligned} \beta &= \beta_{xy} dx \wedge dy + \beta_{yz} dy \wedge dz + \beta_{zx} dz \wedge dx, \\ X \lrcorner \beta &= \left(\beta_{zx} X^z - \beta_{xy} X^y \right) dx + \\ &\quad + \left(\beta_{xy} X^x - \beta_{yz} X^z \right) dy + \\ &\quad + \left(\beta_{yz} X^y - \beta_{zx} X^x \right) dz, \end{aligned}$$

$$\begin{aligned} \gamma &= \gamma_{xyz} dx \wedge dy \wedge dz, \\ X \lrcorner \gamma &= \gamma_{xyz} \left(X^z dx \wedge dy + X^x dy \wedge dz + X^y dz \wedge dx \right). \end{aligned}$$

Lie Derivative

Here we will study the Lie derivative in coordinates. Let f be a scalar (zero-form), X a vector field, and α be a one-form, where

$$X = X^i e_i = X^i \frac{\partial}{\partial x^i},$$
$$\alpha = \alpha_i e^i = \alpha_i dx^i.$$

$\{e_1, \ldots, e_n\} = \{\frac{\partial}{\partial x^1}, \ldots, \frac{\partial}{\partial x^n}\}$ form the bases that span the space of vector fields, and $\{e^1, \ldots, e^n\} = \{dx^1, \ldots, dx^n\}$ are the basis elements for the space of one-forms. Using Cartan's magic formula the Lie derivative is given algebraically by

$$\mathcal{L}_X \alpha = \mathbf{d}(X \lrcorner \alpha) + X \lrcorner \mathbf{d}\alpha. \tag{A.7}$$

In coordinate form the Lie derivative of a scalar, a one-form, and a vector field become

$$\mathcal{L}_X f = X \lrcorner \mathbf{d}f = X^i \frac{\partial f}{\partial x^i},$$
$$\mathcal{L}_X Y = [X, Y] = X^i \frac{\partial Y^j}{\partial x^i} \frac{\partial}{\partial x^j} - Y^i \frac{\partial X^j}{\partial x^i} \frac{\partial}{\partial x^j}.$$

To compute the Lie derivative of a one form we use

$$\mathbf{d}\alpha = \frac{\partial \alpha_i}{\partial x^j} dx^j \wedge dx^i,$$

$$\begin{aligned} X \lrcorner \mathbf{d}\alpha &= X^k \frac{\partial}{\partial x^k} \lrcorner \mathbf{d}\alpha \\ &= X^k \frac{\partial \alpha_i}{\partial x^j} \frac{\partial}{\partial x^k} \lrcorner \left(dx^j \wedge dx^i \right) \\ &= X^k \frac{\partial \alpha_i}{\partial x^j} \left(\left(\frac{\partial}{\partial x^k} \lrcorner dx^j \right) \wedge dx^i - \left(dx^j \wedge \frac{\partial}{\partial x^k} \lrcorner dx^i \right) \right) \\ &= X^k \frac{\partial \alpha_i}{\partial x^j} \left(\delta^j_k dx^i - \delta^i_k dx^j \right) \\ &= X^j \frac{\partial \alpha_i}{\partial x^j} dx^i - X^i \frac{\partial \alpha_i}{\partial x^j} dx^j, \end{aligned}$$

$$\begin{aligned} X \lrcorner \alpha &= X^i \lrcorner \frac{\partial}{\partial x^i} \alpha \\ &= X^i \alpha_j \frac{\partial}{\partial x^i} \lrcorner dx^j \\ &= X^i \alpha_j \delta^j_i \\ &= X^i \alpha_i, \end{aligned}$$

$$\begin{aligned}\mathbf{d}(X \lrcorner \alpha) &= \mathbf{d}\left(X^i \alpha_i\right) \\ &= \left(\frac{\partial X^i}{\partial x^j}\alpha_i + X^i \frac{\partial \alpha_i}{\partial x^j}\right) dx^j.\end{aligned}$$

Combining the terms above we obtain the results for the Lie derivative of a 1-form

$$\mathcal{L}_X \alpha = X^j \frac{\partial \alpha_i}{\partial x^j} dx^i + \frac{\partial X^i}{\partial x^j}\alpha_i dx^j,$$

$$\mathcal{L}_X \alpha = \left(X^j \frac{\partial \alpha_i}{\partial x^j} + \frac{\partial X^j}{\partial x^i}\alpha_j\right) dx^i.$$

The Lie derivative is a derivation and so it must satisfy a product rule over the pairing of forms and vector fields:

$$\mathcal{L}_X(Y \lrcorner \alpha) = (\mathcal{L}_X Y) \lrcorner \alpha + Y \lrcorner (\mathcal{L}_X \alpha). \tag{A.8}$$

To see that the above expressions for the Lie derivative do indeed satisfy Eq. (A.8):

$$\begin{aligned}\mathcal{L}_X(Y \lrcorner \alpha) &= \mathcal{L}_X\left(\alpha_i Y^i\right) \\ &= X^j \frac{\partial}{\partial x^j}\left(\alpha_i Y^i\right) \\ &= X^j \frac{\partial \alpha_i}{\partial x^j} Y^i + X^j \alpha_i \frac{\partial Y^i}{\partial x^j},\end{aligned}$$

$$\begin{aligned}Y \lrcorner \mathcal{L}_X \alpha &= \left(Y^k \frac{\partial}{\partial x^k}\right) \lrcorner \left(X^j \frac{\partial \alpha_i}{\partial x^j} dx^i + \frac{\partial X^i}{\partial x^j}\alpha_i dx^j\right) \\ &= X^j \frac{\partial \alpha_i}{\partial x^j} Y^k \left(\frac{\partial}{\partial x^k} \lrcorner dx^i\right) + \frac{\partial X^i}{\partial x^j}\alpha_i Y^k \left(\frac{\partial}{\partial x^k} \lrcorner dx^j\right) \\ &= X^j \frac{\partial \alpha_i}{\partial x^j} Y^k \delta^i_k + \frac{\partial X^i}{\partial x^j}\alpha_i Y^k \delta^j_k \\ &= X^j \frac{\partial \alpha_i}{\partial x^j} Y^i + \frac{\partial X^i}{\partial x^j}\alpha_i Y^j,\end{aligned}$$

$$\begin{aligned}(\mathcal{L}_X Y) \lrcorner \alpha &= \left(X^i \frac{\partial Y^j}{\partial x^i}\frac{\partial}{\partial x^j} - Y^i \frac{\partial X^j}{\partial x^i}\frac{\partial}{\partial x^j}\right) \lrcorner (\alpha_k dx^k) \\ &= \left(\alpha_k X^i \frac{\partial Y^j}{\partial x^i} - \alpha_k Y^i \frac{\partial X^j}{\partial x^i}\right)\left(\frac{\partial}{\partial x^j} \lrcorner dx^k\right) \\ &= \left(\alpha_k X^i \frac{\partial Y^j}{\partial x^i} - \alpha_k Y^i \frac{\partial X^j}{\partial x^i}\right)\delta^k_j \\ &= \alpha_j X^i \frac{\partial Y^j}{\partial x^i} - \alpha_j Y^i \frac{\partial X^j}{\partial x^i}.\end{aligned}$$

After a simple relabeling of the dummy indices it can be shown that Eq. (A.8) is indeed satisfied. Thus, in two dimensions it can be shown that the Lie derivative acts on scalars, vectors, and 1-forms as follows:

$$
\begin{aligned}
\mathcal{L}_X f &= X^x \frac{\partial f}{\partial x} + X^y \frac{\partial f}{\partial y}, \\
\mathcal{L}_X Y &= \left[X^x \frac{\partial}{\partial x} + X^y \frac{\partial}{\partial y}, Y^x \frac{\partial}{\partial x} + Y^y \frac{\partial}{\partial y} \right] \\
&= X^x \frac{\partial Y^x}{\partial x} \frac{\partial}{\partial x} + X^y \frac{\partial Y^x}{\partial y} \frac{\partial}{\partial x} + X^x \frac{\partial Y^y}{\partial x} \frac{\partial}{\partial y} + X^y \frac{\partial Y^y}{\partial y} \frac{\partial}{\partial y} - \\
&- Y^x \frac{\partial X^x}{\partial x} \frac{\partial}{\partial x} - Y^y \frac{\partial X^x}{\partial y} \frac{\partial}{\partial x} - Y^x \frac{\partial X^y}{\partial x} \frac{\partial}{\partial y} - Y^y \frac{\partial X^y}{\partial y} \frac{\partial}{\partial y} \\
&= \left(X^x \frac{\partial Y^x}{\partial x} + X^y \frac{\partial Y^x}{\partial y} - Y^x \frac{\partial X^x}{\partial x} - Y^y \frac{\partial X^x}{\partial y} \right) \frac{\partial}{\partial x} + \\
&+ \left(X^x \frac{\partial Y^y}{\partial x} + X^y \frac{\partial Y^y}{\partial y} - Y^x \frac{\partial X^y}{\partial x} - Y^y \frac{\partial X^y}{\partial y} \right) \frac{\partial}{\partial y}, \\
\mathcal{L}_X \alpha &= X^x \frac{\partial \alpha_x}{\partial x} dx + X^y \frac{\partial \alpha_x}{\partial y} dx + X^x \frac{\partial \alpha_y}{\partial x} dy + X^y \frac{\partial \alpha_y}{\partial y} dy + \\
&+ \frac{\partial X^x}{\partial x} \alpha_x dx + \frac{\partial X^y}{\partial x} \alpha_y dx + \frac{\partial X^x}{\partial y} \alpha_x dy + \frac{\partial X^y}{\partial y} \alpha_y dy \\
&= \left(X^x \frac{\partial \alpha_x}{\partial x} + X^y \frac{\partial \alpha_x}{\partial y} + \frac{\partial X^x}{\partial x} \alpha_x + \frac{\partial X^y}{\partial x} \alpha_y \right) dx + \\
&+ \left(X^x \frac{\partial \alpha_y}{\partial x} + X^y \frac{\partial \alpha_y}{\partial y} + \frac{\partial X^x}{\partial y} \alpha_x + \frac{\partial X^y}{\partial y} \alpha_y \right) dy.
\end{aligned}
$$

Given a two form

$$\Omega = \omega \, dx \wedge dy,$$

by Cartan's magic formula

$$
\begin{aligned}
\mathcal{L}_X \Omega &= \mathbf{d}(X \lrcorner \Omega) \\
&= \left(\frac{\partial}{\partial x}(\omega X^x) + \frac{\partial}{\partial y}(\omega X^y) \right) dx \wedge dy.
\end{aligned}
$$

A.5 Variational Derivation of the Equation of Motion of an Ideal Fluid

For ideal fluids, the configuration space is the group of volume-preserving diffeomorphisms $\mathrm{Diff}_{\mathrm{Vol}}(\mathcal{M})$ on the fluid container (in general the region may change with time as a result of deformations of the underlying manifold). A particle located at a point $x_0 \in \mathcal{M}$ will travel to a point $x_t = \varphi_t(x_0) \in \mathcal{M}$ at time t,

$$\varphi_t : \quad \mathcal{M} \to \mathcal{M}.$$

Since there is no potential energy, the Lagrangian is simply the kinetic energy, which is a mapping from the tangent bundle to the real numbers:

$$L: \quad T\mathrm{Diff}_{\mathrm{Vol}} \to \mathbb{R},$$

where

$$L(\varphi, \varphi) = \frac{1}{2} \int_{\mathcal{M}} \|\varphi \circ \varphi^{-1}\|^2 dV.$$

Notice that the Lagrangian satisfies the particle relabeling symmetry expressed as invariance under right composition:

$$L(\varphi \circ \phi, \varphi \circ \phi) = L(\varphi, \varphi),$$

where $\phi, \varphi \in \mathrm{TDiff}_{Vol}$. The action is given by

$$S(\varphi(t), \varphi(t)) = \int_{t_0}^{t_1} L \, dt.$$

Due to the particle relabeling symmetry the system can be described in terms of a reduced Lagrangian

$$\begin{aligned} l: \quad & \mathfrak{diff}_{Vol} \to \mathbb{R} \\ & \xi \mapsto l(\xi) = L(e, \xi) = L(\phi, \xi \circ \phi), \end{aligned}$$

where $\xi \in \mathfrak{diff}_{Vol}$ is in the Lie algebra of volume-preserving diffeomorphisms, and e is the identity element of the TDiff_{Vol} group. To obtain the reduced action:

$$\begin{aligned} S(\varphi(t), \varphi(t)) &= \int_{t_0}^{t_1} L(\varphi(t), \varphi(t)) \, dt \\ &= \int_{t_0}^{t_1} L(e, \varphi(t) \circ \varphi(t)^{-1}) \, dt \\ &= \int_{t_0}^{t_1} l(\xi(t)) \, dt \quad \text{where } \xi = \varphi \circ \varphi^{-1} \\ &=: s(\xi(t)) \quad \text{Reduced Action Function .} \end{aligned}$$

Variations must satisfy the Lin Constraint:

$$\begin{aligned} \delta\xi &= \delta(\varphi\varphi^{-1}) \\ &= \delta\varphi\varphi^{-1} - \varphi\varphi^{-1}\delta\varphi\varphi^{-1} \\ &= \eta + [\xi, \eta], \quad \text{where } \eta = \delta\varphi \circ \varphi^{-1}. \end{aligned}$$

The variation in the reduced action is

$$\begin{aligned}\delta s &= \delta\left(\frac{1}{2}\int_{t_1}^{t_2}\int_{\mathcal{F}} v^2\, dt dV\right)\\ &= \int_{t_1}^{t_2}\int_{\mathcal{M}} \langle v^\flat, \delta v\rangle dt dV\\ &= \int_{t_1}^{t_2}\int_{\mathcal{M}} \left(\langle v^\flat, u\rangle + \langle v^\flat, [v,u]\rangle\right) dt\, dV,\end{aligned}$$

$$\langle v^\flat, [v,u]\rangle = \langle v^\flat, L_v u\rangle = L_v\langle v^\flat, u\rangle - \langle L_v v^\flat, u\rangle.$$

For a divergence free vector field,

$$\int_{\mathcal{M}} L_v f \overset{\nabla\cdot v=0}{=} \int_{\partial\mathcal{M}} (v,n) f = 0.$$

Therefore,

$$\delta s = \int_{t_1}^{t_2}\int_{\mathcal{M}} \left(\langle v^\flat, u\rangle - \langle L_v v^\flat, u\rangle\right)\, dt\, dV.$$

Setting the variation $\delta s = 0$ and integrating by parts:

$$\int_{\mathcal{M}} dV\, \langle v^\flat + L_v v^\flat, u\rangle = 0.$$

This implies that the integrand must be the gradient of a function

$$v^\flat + L_v v^\flat + dp = 0,$$

which is equivalent to the Euler equation in its familiar coordinate form:

$$\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla\mathbf{p} = 0.$$

BIBLIOGRAPHY

[1] Ralph Abraham, Jerrold E. Marsden, and Tudor Ratiu. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer, 1988.

[2] D. N. Arnold, R. S. Falk, and R. Winther. "Finite Element Exterior Calculus, Homological Techniques, and Applications". In: *Acta Numerica* 15 (2006), pp. 1–155.

[3] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. "Finite element exterior calculus: from Hodge theory to numerical stability". In: *Bull. Amer. Math. Soc. (N.S.)* 47 (2010), pp. 281–354.

[4] Douglas N Arnold et al. *Compatible spatial discretizations*. Vol. 142. Springer Science & Business Media, 2007.

[5] Bernhard Auchmann and Stefan Kurz. "A geometrically defined discrete Hodge operator on simplicial cells". In: *IEEE Transactions on Magnetics* 42.4 (2006), pp. 643–646.

[6] S. Behnel et al. "Cython: The Best of Both Worlds". In: *Computing in Science Engineering* 13.2 (2011), pp. 31–39. ISSN: 1521-9615. DOI: 10.1109/MCSE.2010.118.

[7] Pavel B Bochev and James M Hyman. "Principles of mimetic discretizations of differential operators". In: *Compatible spatial discretizations*. Springer, 2006, pp. 89–119.

[8] A. Bossavit. *Computational Electromagnetism*. Academic Press, Boston, 1998.

[9] Alain Bossavit. "Generating Whitney forms of polynomial degree one and higher". In: *IEEE Transactions on Magnetics* 38.2 (Mar. 2002), pp. 341–344. DOI: 10.1109/20.996092.

[10] Mick Bouman et al. "A conservative spectral element method for curvilinear domains". In: *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 111–119.

[11] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[12] John P. Boyd and Rolfe Petschek. "The Relationships Between Chebyshev, Legendre and Jacobi Polynomials: The Generic Superiority of Chebyshev Polynomials and Three Important Exceptions". In: *J. Sci. Comput.* 59.1 (Apr. 2014), pp. 1–27.

[13] F. Brezzi, A. Buffa, and G. Manzini. “Mimetic scalar products of discrete differential forms”. In: *Journal of Computational Physics* 257, Part B (2014), pp. 1228–1259.

[14] Oscar P Bruno, Youngae Han, and Matthew M Pohlman. “Accurate, high-order representation of complex three-dimensional surfaces via Fourier continuation analysis”. In: *Journal of Computational Physics* 227.2 (2007), pp. 1094–1125.

[15] William L. Burke. *Applied Differential Geometry*. Cambridge University Press, 1985.

[16] Brian Cabral and Leith Casey Leedom. “Imaging vector fields using line integral convolution”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. 1993, pp. 263–270.

[17] C.G. Canuto et al. *Spectral Methods: Fundamentals in Single Domains*. Scientific Computation Series. Springer, 2006.

[18] Élie Cartan. *Les Systémes Differentiels Exterieurs et leurs Applications Géometriques*. Paris: Hermann, 1945.

[19] Keenan Crane, Mathieu Desbrun, and Peter Schröder. “Trivial Connections on Discrete Surfaces”. In: *Computer Graphics Forum* 29.5 (2010), pp. 1525–1533.

[20] M. Desbrun, A. N. Hirani, and J. E. Marsden. “Discrete exterior calculus for variational problems in computer vision and graphics”. In: *Proc. CDC* 42 (2003), pp. 533–538.

[21] Mathieu Desbrun, Eva Kanso, and Yiying Tong. “Discrete Differential Forms for Computational Modeling”. In: *Discrete Differential Geometry*. Ed. by A. Bobenko and P. Schröder. Springer, 2007, pp. 287–324.

[22] Mathieu Desbrun et al. “Discrete Exterior Calculus”. In: *arXiv:math/0508341v2* (2005).

[23] Sharif Elcott and Peter Schroder. “Building your own DEC at home”. In: *ACM SIGGRAPH 2006 Courses*. 2006, pp. 55–59.

[24] Stanley J Farlow. *Partial differential equations for scientists and engineers*. Courier Corporation, 1993.

[25] Harley Flanders. *Differential Forms with Applications to the Physical Sciences by Harley Flanders*. Vol. 11. Elsevier, 1963.

[26] Theodore Frankel. *The Geometry of Physics*. Second Edition. United Kingdom: Cambridge University Press, 2004.

[27] Matteo Frigo and Steven G Johnson. “FFTW: An adaptive software architecture for the FFT”. In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 3. IEEE. 1998, pp. 1381–1384.

[28] Marc Gerritsma. “Edge functions for spectral element methods”. In: *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 199–207.

[29] Marc Gerritsma, Jeroen Kunnen, and Boudewijn de Heij. “Discrete Lie Derivative”. In: *Numerical Mathematics and Advanced Applications ENUMATH 2015*. Springer. 2016, pp. 635–643.

[30] Fernando de Goes et al. “Subdivision Exterior Calculus for Geometry Processing”. In: *ACM Trans. Graph.* 35.4 (July 2016), Art. 133.

[31] Leo J. Grady and Jonathan R. Polimeni. *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer, 2010.

[32] P. W. Gross and P. R. Kotiuga. “Electromagnetic theory and computation: a topological approach”. In: *Mathematical Sciences Research Institute Publications, Cambridge University Press* 48 (2004).

[33] J. Harrison. “Ravello lecture notes on geometric calculus”. In: *arXiv:math-ph/0501001* (2005).

[34] R.R. Hiemstra et al. “High order geometric methods with exact conservation properties”. In: *Journal of Computational Physics* 257 (2014), pp. 1444–1471.

[35] A. N. Hirani. “Discrete Exterior Calculus”. PhD thesis. Pasadena, CA: California Institute of Technology, 2003.

[36] Michael Holst and Ari Stern. “Geometric variational crimes: Hilbert complexes, finite element exterior calculus, and problems on hypersurfaces”. In: *Found. Comput. Math.* 12.3 (2012), pp. 263–293.

[37] John D Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95.

[38] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 — Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2016.

[39] David A. Kopriva. “A Staggered-Grid Multidomain Spectral Method for the Compressible Navier-Stokes Equations”. In: *Journal of Computational Physics* 143.1 (1998), pp. 125–158.

[40] P Robert Kotiuga. “Theoretical limitations of discrete exterior calculus in the context of computational electromagnetics”. In: *IEEE Transactions on Magnetics* 44.6 (2008), pp. 1162–1165.

[41] E. Kreyszig. *Differential Geometry*. Dover Books on Mathematics. Dover Publications, 2013.

[42] Mehrdad Lakestani and Mehdi Dehghan. "The use of Chebyshev cardinal functions for the solution of a partial differential equation with an unknown time-dependent coefficient subject to an extra measurement". In: *Journal of Computational and Applied Mathematics* 235.3 (Dec. 2010), pp. 669–678. DOI: 10.1016/j.cam.2010.06.020.

[43] Melvin Leok. "Foundations of computational geometric mechanics". PhD thesis. Pasadena, CA: California Institute of Technology, 2004.

[44] Nam Mai-Duy and Roger I Tanner. "A spectral collocation method based on integrated Chebyshev polynomials for two-dimensional biharmonic boundary-value problems". In: *Journal of Computational and Applied Mathematics* 201.1 (2007), pp. 30–47.

[45] T. Mathew. *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2008. ISBN: 9783540772057.

[46] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (2017), e103.

[47] Patrick Mullen et al. "HOT: Hodge Optimized Triangulations". In: *ACM Transactions on Graphics* 30.3 (July 2011), Art. 103.

[48] P. Mullen et al. "Discrete Lie Advection of Differential Forms". In: *Foundations of Computational Mathematics* 11.2 (Sept. 2010), pp. 131–149. DOI: 10.1007/s10208-010-9076-y.

[49] James R. Munkres. *Elements of Algebraic Topology*. Menlo Park, CA: Addison-Wesley, 1984.

[50] M. Nakahara. *Geometry, Topology and Physics, Second Edition*. Graduate student series in physics. Taylor & Francis, 2003. ISBN: 9780750306065.

[51] J.-C. Nédélec. "Mixed finite elements in $\mathbb{R}^3$". In: *Numer. Math.* 35 (1980), pp. 315–341.

[52] Roy A. Nicolaides and Xiaonan Wu. "Covolume Solutions of Three Dimensional Div-Curl Equations". In: *SIAM J. Numer. Anal.* 34 (1997), p. 2195.

[53] J. O'Rourke. *Computational geometry in C (second edition)*. Cambridge University Press, 1998.

[54] Artur Palha and Marc Gerritsma. "Mimetic spectral element method for Hamiltonian systems". In: *arXiv:1505.03422* (2015).

[55] Artur Palha and Marc Gerritsma. "Spectral Element Approximation of the Hodge-$\star$ Operator in Curved Elements". In: *Spectral and High Order Methods for Partial Differential Equations*. Ed. by Jan S. Hesthaven et al. Vol. 76. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2011, pp. 283–291.

[56] Artur Palha and Marc Gerritsma. “Spectral element approximation of the Hodge-$\star$ operator in curved elements”. In: *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 283–291.

[57] Artur Palha et al. “Physics-compatible discretization techniques on single and dual grids, with application to the Poisson equation of volume forms”. In: *Journal of Computational Physics* 257 (Jan. 2014), pp. 1394–1422. DOI: 10.1016/j.jcp.2013.08.005.

[58] Dmitry Pavlov et al. “Structure-preserving discretization of incompressible fluids”. In: *Physica D: Nonlinear Phenomena* 240.6 (2011), pp. 443–458.

[59] Nicolas Robidoux. *Polynomial Histopolation, Superconvergent Degrees Of Freedom, And Pseudospectral Discrete Hodge Operators*. Preprint. 2006.

[60] Nicolas Robidoux and Stanly Steinberg. “A discrete vector calculus in tensor grids”. In: *Comput. Methods Appl. Math.* 1.2001 (2011), pp. 1–44.

[61] Nicolas P Rougier. “Glumpy”. In: *EuroScipy*. 2015.

[62] Dzhelil Rufat et al. “The chain collocation method: A spectrally accurate calculus of forms”. In: *Journal of Computational Physics* 257 (Jan. 2014), pp. 1352–1372. DOI: 10.1016/j.jcp.2013.08.011.

[63] Jonathan Richard Shewchuk. “Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator”. In: *Applied computational geometry towards geometric engineering*. Springer, 1996, pp. 203–222.

[64] Ari Stern et al. “Geometric Computational Electrodynamics with Variational Integrators and Discrete Differential Forms”. In: *Geometry, Mechanics, and Dynamics: The Legacy of Jerry Marsden*. Ed. by Dong Eui Chang et al. Springer New York, 2015, pp. 437–475.

[65] Ari Stern et al. “Variational integrators for Maxwell’s equations with sources”. In: *PIERS Online* 4.7 (2008), pp. 711–715.

[66] Kunihiko Taira and Tim Colonius. “The immersed boundary method: a projection approach”. In: *Journal of Computational Physics* 225.2 (2007), pp. 2118–2137.

[67] Timo Tarhasaari, Lauri Kettunen, and Alain Bossavit. “Some realizations of a discrete Hodge operator: a reinterpretation of finite element techniques”. In: *IEEE Transactions on magnetics* 35.3 (1999), pp. 1494–1497. DOI: 10.1109/20.767250.

[68] Yiying Tong et al. “Designing quadrangulations with discrete harmonic forms”. In: *Symposium on Geometry Processing*. 2006, pp. 201–210.

[69] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Philadelphia: SIAM, 2000.

[70] Jonathan Turner and Jason Turner. *ChaiScript: Easy to use scripting for C++*.

[71] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.

[72] Ke Wang et al. “Edge subdivision schemes and the construction of smooth vector fields”. In: *ACM Transactions on Graphics*. Vol. 25(3). 2006, pp. 1041–1048.

[73] H. Whitney. *Geometric integration theory*. Princeton University Press, 1957.