ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Hodge-Laplacians and Dirac Operators on the Surface of the 3-Sphere

Bachelor Thesis

Nico Graf

September 13th 2022

Advisor: Prof. Dr. Ralf Hiptmair
Department of Mathematics, ETH Zürich

Abstract

The thesis discusses the h-convergence of the discrete solutions for the Dirac operator and the Hodge Laplacians on the surface of the 3-sphere. At first, it describes the discretization with finite element methods based on the Lehrfem++. The h-convergence experiments then show algebraic convergence with rate one for all source problems. Moreover, the Dirac operator turned out to be solvable numerically with the Laplace Operator and a suitable modification of the load function.

Contents

Chapter 1

Introduction

This thesis discusses the discretization of the Hodge Laplacians for the Whitney zero, one, and two forms, as well as the Dirac operator using the same discrete function spaces. This includes the derivation of variational formulations, discretization of the function spaces, the realization of the discrete source problems, and an h-convergence study of these source problems on the surface of the 3-sphere. Moreover, for the convergence studies, we provide a triangulation of the sphere.

According to the previously described main goals, the thesis comprises five parts. The first part, starting in chapter 2, discusses the mathematical foundations of the problem and introduces the necessary notation. More specifically, we define the function spaces and state the definitions of the Hodge-Laplacian and Dirac operators. Furthermore, we introduce the source problems, on which the h-convergence studies will be done in chapter 6. The chapter also touches on the variational problems which can be solved with the finite element method.

The second part, chapter 3, concerns the discretization of the function spaces used in the source problems. Namely, these are the three first Whitney k-forms.

The third part, chapter 4, discusses the Implementation based on the Lehrfem++ framework, a c++ library developed at the Seminar for Applied Mathematics at ETH Zürich [4]. This chapter starts with a description of the triangulation of the sphere and continues by providing the technical details needed for the realization of the element providers for each of the bilinear forms in the variational problems. Then it also discusses how these element providers cooperate to represent the Dirac operator and the Hodge Laplacians.

Chapter 5 introduces subsequently the methods used to debug the Dirac operator, which at first had a sign error in one of the matrix providers, leading to a nonconverging solution.

Finally, in the fifth part, chapter 6 and 7, discusses how well the method performs in terms of converge under h-refinement. Chapter 6 starts with a introduction of the tested functions and presents the results of the numerical experiments. This is followed by a final discussion and conclusions in chapter 7.

Chapter 2

Mathematical Foundations

This chapter first defines the surface differential operators occurring in the problems of this thesis. These definitions are necessary not only for the discretization discussed in chapter 3, but also in the analytical derivation of test functions for the experiments presented in chapter 6. Then we introduce definitions of the Hodge Laplacian operators and the Dirac operators, the main components of the source problems. The last section of the chapter is devoted to the source problems themselves, of which discretization and the h-convergence are the main goals.

2.1 Surface Differential Operators

The problems in this thesis consist of four basic differential operators, introduced in the following subsections. The last subsection is devoted to the involved function spaces. Throughout the thesis, surface differential operators are marked with a subscript Γ, for example $\mathbf{grad}_\Gamma$. The subscript denotes a projection onto the two-dimensional tangential space of a two-dimensional Manifold Γ in the three-dimensional world. In the experiments, chapter 6, we restrict this Manifold to be the surface of the 3-sphere $\Gamma = \partial\mathbb{S}$.

2.1.1 Surface Gradient

The surface gradient is the gradient projected on the tangent space of the surface. Formally we have

$$\mathbf{grad}_\Gamma u := (I_3 - \mathbf{n}\,\mathbf{n}^T)\,\mathbf{grad}\,\widetilde{u} \qquad\qquad u : \Gamma \to \mathbb{R}, \tag{2.1}$$

where the unit normal vector,

$$\mathbf{n} := \mathbf{n}_\Gamma(\mathbf{x}) \qquad\qquad \mathbf{x} \in \mathbb{S}, \tag{2.2}$$

is implicitly defined with respect to the surface Γ and a coordinate $\mathbf{x} \in \Gamma$. For example on the surface of the sphere with radius one the normal is equivalent to

$$\mathbf{n}_{\partial S}(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|}. \tag{2.3}$$

And $\widetilde{u}_{|\Gamma}$ is an extension of u such that

$$\widetilde{u}_{|\Gamma} = u. \tag{2.4}$$

2.1.2 Surface Divergence

Similar to the surface gradient, the surface divergence is the divergence with a correction term for the divergence in non-tangential direction given in

$$\operatorname{div}_\Gamma \mathbf{u} := \operatorname{div} \widetilde{\mathbf{u}} - \mathbf{n}^T D\widetilde{\mathbf{u}}\, \mathbf{n} \qquad \mathbf{u} : \Gamma \to \mathbb{R}^3, \tag{2.5}$$

with the normal $\mathbf{n}$, defined above in (2.2), and the Jacobian $D\widetilde{\mathbf{u}}$ of the surface-vectorfield $\widetilde{\mathbf{u}}$. And $\widetilde{\mathbf{u}}$ is an extension of $\mathbf{u}$ analogue to 2.4.

2.1.3 Surface Curl

We define two surface curls, one operating on tangential surface-vectorfields and one operating on scalar surface-functions in

$$\begin{aligned} \operatorname{curl}_\Gamma \mathbf{u} &:= \operatorname{div}_\Gamma(R_{\mathbf{n}}\mathbf{u}) && (2.6)\\ &= \operatorname{div}_\Gamma(\mathbf{n} \times \mathbf{u}) \qquad \mathbf{u} : \Gamma \to \mathbb{R}^3, && (2.7) \end{aligned}$$

with $R_{\mathbf{n}}$ a rotation operator, rotating the field 90 degrees around $\mathbf{n}$.

For the vector-curl operating on scalar functions, we define

$$\begin{aligned} \mathbf{curl}_\Gamma u &:= R_{\mathbf{n}}(\mathbf{grad}_\Gamma u) && (2.8)\\ &= \mathbf{n} \times (\mathbf{grad}_\Gamma u) \qquad u : \Gamma \to \mathbb{R}. && (2.9) \end{aligned}$$

2.1.4 Function Spaces

For the variational problems, derived later in section 2.5 to be meaningful, we need to choose a compliant pair of trial and test spaces. The choice of the function spaces depends on the operators acting on the trial and test functions in the integrals of the variational forms. Hence, in this chapter, we introduce these function spaces $H^1(\Gamma), \mathbf{H}(\operatorname{curl}_\Gamma, \Gamma), \mathbf{H}(\operatorname{div}_\Gamma, \Gamma)$, and $L^2(\Gamma)$, of which definitions are drawn from [7, section 1.3] and [3, section 2]. The

above spaces have the interesting properties

$$\forall u \in H^1(\Gamma) \qquad \int_\Gamma |\mathbf{grad}_\Gamma\, u|^2 \, d\mathbf{x} < \infty \tag{2.10}$$

$$\forall \mathbf{u} \in \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma) \qquad \int_\Gamma |\mathrm{curl}_\Gamma\, \mathbf{u}|^2 \, d\mathbf{x} < \infty \tag{2.11}$$

$$\forall \mathbf{u} \in \mathbf{H}(\mathrm{div}_\Gamma, \Gamma) \qquad \int_\Gamma |\mathrm{div}_\Gamma\, \mathbf{u}|^2 \, d\mathbf{x} < \infty \tag{2.12}$$

$$\forall u \in L^2(\Gamma) \qquad \int_\Gamma |u|^2 \, d\mathbf{x} < \infty, \tag{2.13}$$

from which we deduce that the variational forms in section 2.5 are well defined.

2.2 Hodge Laplacians

This section introduces the three Hodge Laplacian operators on the zero, one, and two forms. More general formulations can be found in [1, section 7.1]. The special cases discussed in the current section are stated in [1, section 7.4].

The first Hodge-Laplacian on the zero form is defined as

$$\Delta_0\, u := (\mathrm{div}_\Gamma \circ \mathbf{grad}_\Gamma)\, u, \tag{2.14}$$

with div_Γ from (2.5) and $\mathbf{grad}_\Gamma$ defined in (2.1).

The second Hodge-Laplacian on the 1-form is defined to be

$$\Delta_1\, \mathbf{u} := (\mathbf{grad}_\Gamma \circ \mathrm{div}_\Gamma - \mathbf{curl}_\Gamma \circ \mathrm{curl}_\Gamma)\, \mathbf{u}, \tag{2.15}$$

with operators $\mathrm{div}_\Gamma, \mathbf{grad}_\Gamma$ as above. Moreover the vector-curl $\mathbf{curl}_\Gamma$ is defined in (2.8) and the scalar-curl curl_Γ in (2.6).

Then we have the third Hodge-Laplacian acting on the 2-form given as

$$\Delta_2\, u := (\mathrm{div}_\Gamma \circ \mathbf{grad}_\Gamma)\, u, \tag{2.16}$$

which formally agrees with the definition of the first Hodge-Laplacian (2.14). The difference will become obvious when we discuss the variational formulations in section 2.5.

2.3 Dirac Operator

For the Dirac operator, we have the definition based on [10, section 2.2]

$$D := \begin{bmatrix} 0 & \mathbf{grad}_\Gamma^* & 0 \\ \mathbf{grad}_\Gamma & 0 & \mathrm{curl}_\Gamma^* \\ 0 & \mathrm{curl}_\Gamma & 0 \end{bmatrix}. \tag{2.17}$$

Where $\mathbf{grad}_\Gamma$ and curl_Γ are defined as above (2.1, 2.6) and for $\mathbf{grad}^*_\Gamma$, curl^*_Γ we have

$$\mathbf{grad}^*_\Gamma \equiv -\mathrm{div}_\Gamma \tag{2.18}$$
$$\mathrm{curl}^*_\Gamma \equiv \mathbf{curl}_\Gamma. \tag{2.19}$$

2.4 Source Problems

In the experiments, chapter 6, we study the h-convergence of the discretization of the source problems

$$-\Delta_0 u + k^2 u = f_0 \tag{2.20}$$
$$-\Delta_1 \mathbf{u} + k^2 \mathbf{u} = \mathbf{f}_1 \tag{2.21}$$
$$-\Delta_2 u + k^2 u = f_2 \tag{2.22}$$
$$D\vec{u} + \imath k \vec{u} = \vec{f}, \tag{2.23}$$

for some $k \neq 0, k \in \mathbb{R}$ and functions $f_0 : \Gamma \to \mathbb{R}$, $\mathbf{f}_1 : \Gamma \to \mathbb{R}^3$,$f_2 : \Gamma \to \mathbb{R}$ and $\vec{f} : \Gamma \to \mathbb{C} \times \mathbb{C}^3 \times \mathbb{C}$. Therefore the discretization of these problems is one of the main goals in this thesis, discussed in chapter 3. In the experiments section, we then study, apart from the h-convergence the stability of the method for different values of k.

2.5 Variational Problems

For the source problems above, we formulate the corresponding variational problems, which can be solved with finite element methods. That is, we transform the source problems (2.20, 2.21, 2.22, 2.23) into the following equivalent formulations under the assumption that the domain Γ does not have a boundary $\partial\Gamma = \emptyset$. This assumption is reasonable for $\Gamma = \partial\mathbb{S}$, which will be the only domain studied in the experiments. For the first source problem we have

$$u \in H^1(\Gamma) \tag{2.24}$$
$$\int_\Gamma \mathbf{grad}_\Gamma u \cdot \mathbf{grad}_\Gamma v \, d\mathbf{x} + k^2 \int_\Gamma u\, v \, d\mathbf{x} = \int_\Gamma f_0 \, v \, d\mathbf{x} \qquad \forall v \in H^1(\Gamma). \tag{2.25}$$

For the second source problem we introduce an additional variable and set the trial-space of $\mathbf{u}$ to

$$\mathbf{u} \in \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma) \tag{2.26}$$
$$p := \mathrm{div}_\Gamma \mathbf{u} \qquad\qquad p \in H^1(\Gamma), \tag{2.27}$$

and get the two equations for $\forall \mathbf{v} \in \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma), \forall q \in H^1(\Gamma)$,

$$\int_\Gamma \mathrm{curl}_\Gamma \mathbf{u} \cdot \mathrm{curl}_\Gamma \mathbf{v} \, d\mathbf{x} + k^2 \int_\Gamma \mathbf{u} \cdot \mathbf{v} \, d\mathbf{x} + \int_\Gamma \mathbf{v} \cdot \mathbf{grad}_\Gamma p \, d\mathbf{x} = \int_\Gamma \mathbf{f}_1 \cdot \mathbf{v} \, d\mathbf{x} \quad (2.28)$$

$$\int_\Gamma \mathbf{u} \cdot \mathbf{grad}_\Gamma q \, d\mathbf{x} + \int_\Gamma p \, q \, d\mathbf{x} = 0. \quad (2.29)$$

The third source problem, which formally agrees with the first one, will be solved with a different discretization and hence need a different variational formulation. Therefore we again introduce a new variable and set the trialspace of u to

$$u \in L^2(\Gamma) \quad (2.30)$$

$$\mathbf{j} := \mathbf{grad}_\Gamma u \qquad \mathbf{j} \in \mathbf{H}(\mathrm{div}_\Gamma). \quad (2.31)$$

Then we get the following equations

$$\int_\Gamma \mathbf{j} \cdot \mathbf{v} \, d\mathbf{x} + \int_\Gamma q \, \mathrm{div}_\Gamma \mathbf{v} \, d\mathbf{x} = 0 \qquad \forall \mathbf{v} \in \mathbf{H}(\mathrm{div}_\Gamma, \Gamma) \quad (2.32)$$

$$\int_\Gamma u \, \mathrm{div}_\Gamma \mathbf{j} \, d\mathbf{x} + k^2 \int_\Gamma u \, q \, d\mathbf{x} = \int_\Gamma f_2 \, q \, d\mathbf{x} \qquad \forall q \in L^2(\Gamma). \quad (2.33)$$

For the Dirac source problem we get three equation because of the corresponding three components. The trialspace is hence given as

$$\vec{u} \in H^1(\Gamma) \times \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma) \times L^2(\Gamma), \quad (2.34)$$

and the variational problem

$$\imath k \int_\Gamma \vec{u}_1 \, v \, d\mathbf{x} + \int_\Gamma \vec{u}_2 \cdot \mathbf{grad}_\Gamma v \, d\mathbf{x} = \int_\Gamma \vec{f}_1 \, v \, d\mathbf{x} \quad (2.35)$$

$$\int_\Gamma \mathbf{v} \cdot \mathbf{grad}_\Gamma \vec{u}_1 \, d\mathbf{x} + \imath k \int_\Gamma \vec{u}_2 \cdot \mathbf{v} \, d\mathbf{x} + \int_\Gamma \vec{u}_3 \, \mathrm{curl}_\Gamma \mathbf{v} \, d\mathbf{x} = \int_\Gamma \vec{f}_2 \cdot \mathbf{v} \, d\mathbf{x} \quad (2.36)$$

$$\int_\Gamma q \, \mathrm{curl}_\Gamma \vec{u}_2 \, d\mathbf{x} + \imath k \int_\Gamma \vec{u}_3 \, q \, d\mathbf{x} = \int_\Gamma \vec{f}_3 \, q \, d\mathbf{x}. \quad (2.37)$$

$$\forall v \in H^1(\Gamma)$$
$$\forall \mathbf{v} \in \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma)$$
$$\forall q \in L^2(\Gamma)$$

The derivations of the variational problems in this chapter and their equivalence to the source problems stated in (2.20, 2.21, 2.22, 2.23), can be found in appendix A.

Chapter 3

Discretization

This section is concerned with the definition of a finite-dimensional subspace, or rather a finite-dimensional approximation of a subspace, for the given function spaces in the variational forms. Hence we focus on the following four function spaces $H^1(\Gamma), \mathbf{H}(\mathrm{curl}_\Gamma, \Gamma), \mathbf{H}(\mathrm{div}_\Gamma, \Gamma), L^2(\Gamma)$. We will define each discrete function space by finite set of basis functions. Replacing the function spaces by their finite-dimensional approximations then leads to the discretizations of the source problems given in section 2.4. details concerning the discretizations of the source problems will be discussed in chapter 4.

3.1 Mesh

The dimensionality of the discrete function spaces strongly depends on the chosen mesh. Moreover, it has a large influence on the shape of the basis functions and therefore is a fundamental part of discretizations. In this thesis, we require the mesh to be a triangulation of the Domain, the surface of the 3-sphere $\partial\mathbb{S}$. In the following, we denote this triangulation as $\mathcal{M}$. $\mathcal{V}$ refers to the set of vertices in $\mathcal{M}$, $\mathcal{E}$ to the set of edges and $\mathcal{C}$ to the set of cells, i.e. the triangles. Furthermore we require the vertices, edges and cells to be indexed and edges to be oriented. This is achieved with the definitions

$$\mathcal{V} = \{v_j \,|\, j \in \{0, ..., n-1\}, n = |\mathcal{V}|\} \tag{3.1}$$

$$\mathcal{E} = \{e_j \,|\, e_j = (v_{j_0}, v_{j_1}), j \in \{0, ..., m-1\}, m = |\mathcal{E}|\} \tag{3.2}$$

$$\mathcal{C} = \{K_j \,|\, K_j = (v_{j_0}, v_{j_1}, v_{j_2}), j \in \{0, ..., |\mathcal{C}| - 1\}\}. \tag{3.3}$$

These definitions imply that edges have a induced and a intrinsic orientation. Of which the intrinsic orientation is given by the edge definition and the induced orientation are implied by the cell definitions through the order of the vertices. We require the edges within one cell to be oriented counterclockwise, which is equivalent to requiring outward pointing normal vectors.

The induced and intrinsic orientation of edges then further implies a relative orientation s_{j_i} for the edges in a triangle $K_j \in \mathcal{C}$ with $K_j = (v_{j_0}, v_{j_1}, v_{j_2})$

$$s_{j_0} = \begin{cases} 1 & (v_{j_0}, v_{j_1}) \in \mathcal{E} \\ -1 & (v_{j_1}, v_{j_0}) \in \mathcal{E} \end{cases} \tag{3.4}$$

$$s_{j_1} = \begin{cases} 1 & (v_{j_1}, v_{j_2}) \in \mathcal{E} \\ -1 & (v_{j_2}, v_{j_1}) \in \mathcal{E} \end{cases} \tag{3.5}$$

$$s_{j_2} = \begin{cases} 1 & (v_{j_2}, v_{j_0}) \in \mathcal{E} \\ -1 & (v_{j_0}, v_{j_2}) \in \mathcal{E} \end{cases}. \tag{3.6}$$

These relative orientations indicate if the induced orientation agrees with the intrinsic orientation of the edge.

The above definition of the mesh complies with the definition in [7, section 2.5.3]. Hence we can further rely on this reference and associate basis functions with exactly one entity of the mesh. This holds for all the basis functions in this thesis.

The concept of Parametric Finite Element Methods [7, section 2.8] enables us to define the basis function only on the reference triangle and use a bijective pullback function to get basis functions on general triangles of the mesh. This reference triangle is defined to be

$$\widehat{K} := ((0,0),(1,0),(0,1)). \tag{3.7}$$

The pullback function for on an arbitrary triangle in three dimensions $K_j \in \mathcal{C}$ can then be defined as

$$K_j = (v_{j_0}, v_{j_1}, v_{j_2}) \tag{3.8}$$

$$\varphi_{K_j} : \widehat{K} \to K_j \tag{3.9}$$

$$\varphi_{K_j}((x_0, x_1)) := v_{j_0} + x_0\, v_{j_1} + x_1\, v_{j_2}. \tag{3.10}$$

3.2 Discrete Function Spaces

This section provides a definition of the discrete function spaces by a set of basis functions, the global shape functions. In this thesis, we restrict ourselves to linear basis functions with local support. This is, the basis functions are linear only on cells to which their associated entity is adjacent and zero everywhere else.

3.2.1 Whitney Zero-Forms

The discretization of the $H^1(\Gamma)$ space is given by

$$H^1(\Gamma) \to S_1^0(\mathcal{M}). \tag{3.11}$$

This is the piecewise linear continuous finite element space, the Whitney zero-forms [7, section 2.3.4]. The basis functions are associated with the vertices of the mesh. And the local definition of the basis function on the reference triangle are given as the barycentric coordinate functions $\widehat{K}$ (3.7)

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \in \widehat{K} \tag{3.12}$$

$$\widehat{\lambda}_0(\mathbf{x}) = 1 - x_0 - x_1 \tag{3.13}$$

$$\widehat{\lambda}_1(\mathbf{x}) = x_0 \tag{3.14}$$

$$\widehat{\lambda}_2(\mathbf{x}) = x_1. \tag{3.15}$$

This definition can now be used to define global shape functions, using the above definition of the pullback function (3.10) and a restriction to ensure local support

$$\lambda_{j_i}(\mathbf{x}) = \begin{cases} \widehat{\lambda}_i(\varphi_{K_j}^{-1}(\mathbf{x})) & \mathbf{x} \in K_j \subseteq \mathrm{Supp}(\lambda_{j_i}) \\ 0 & \text{otherwise} \end{cases}, \tag{3.16}$$

where we use indexing compliant with the definition of cells (3.3). Moreover we define the Support of a basis function b to be

$$\mathrm{Supp}(b) := \bigcup_{K_j \in A} K_j \tag{3.17}$$

$$A := \{K_j \mid b \text{ is sub-entity of } K_j\}, \tag{3.18}$$

where sub-entities of a given cell are all vertices and edges of that cell and the cell itself.

Then we can also compute the gradients of the basis functions λ_i in the region of their support. For this we rely on lemma [7, lemma 2.8.3.10]

$$\mathbf{grad}_\Gamma \lambda_{j_i}(\mathbf{x}) = D\varphi_{K_j}(\widehat{\mathbf{x}})(D\varphi_{K_j}(\widehat{\mathbf{x}})^T D\varphi_{K_j}(\widehat{\mathbf{x}}))^{-1} \mathbf{grad}\, \widehat{\lambda}_i(\widehat{\mathbf{x}}). \tag{3.19}$$

In the following we will use the following notation for the Gramian matrix

$$G_{K_j}(\widehat{\mathbf{x}}) := D\varphi_{K_j}(\widehat{\mathbf{x}})^T D\varphi_{K_j}(\widehat{\mathbf{x}}). \tag{3.20}$$

Expressing the gradients with the reference gradients will become handy in the implementation. Which is the rational to introduce the on first sight cumbersome expression.

3.2.2 Whitney One-Forms

The space $\mathbf{H}(\mathrm{curl}_\Gamma, \Gamma)$ is discretized by

$$\mathbf{H}(\mathrm{curl}_\Gamma, \Gamma) \to \mathcal{W}^1(\mathcal{M}), \tag{3.21}$$

the space of piecewise linear vector fields, the Whitney 1-forms, surface edge elements [8, section 5]. The basis functions are associated with edges and in a cell K_j we define the local basis functions

$$\mathbf{b}_{j_0|K_j} = s_{j_0}\left(\lambda_{j_0}\,\mathbf{grad}\,\lambda_{j_1} - \lambda_{j_1}\,\mathbf{grad}\,\lambda_{j_0}\right) \tag{3.22}$$

$$\mathbf{b}_{j_1|K_j} = s_{j_1}\left(\lambda_{j_1}\,\mathbf{grad}\,\lambda_{j_2} - \lambda_{j_2}\,\mathbf{grad}\,\lambda_{j_1}\right) \tag{3.23}$$

$$\mathbf{b}_{j_2|K_j} = s_{j_2}\left(\lambda_{j_2}\,\mathbf{grad}\,\lambda_{j_0} - \lambda_{j_0}\,\mathbf{grad}\,\lambda_{j_2}\right). \tag{3.24}$$

Where the factors $s_i \in \{-1, 1\}$ are indicators of the relative orientation of the edges as introduced in (3.4, 3.5, 3.6). The global $\mathbf{b}_i$ is the combination of the two induced $\mathbf{b}_{j_u|K_j}, \mathbf{b}_{l_k|K_l}$ functions in the two triangles K_j, K_l in the support of $\mathbf{b}_i$, analogue to the definition of the support of the barycentric basis functions (3.16).

3.2.3 Rotated Whitney One-Forms

For $\mathbf{H}(\mathrm{div}_\Gamma, \Gamma)$ we define the discretized space by

$$\mathbf{H}(\mathrm{div}_\Gamma, \Gamma) \to \mathcal{W}^1_\times(\mathcal{M}), \tag{3.25}$$

the space of piecewise linear vector fields, the rotated Whitney 1-forms, surface Raviart-Thomas elements. We get these functions by locally rotating the basis functions $\mathbf{b}_i$ around the outward pointing normal of the cell $\mathbf{n}_{K_j}$. Therefore, we again define these functions locally on a triangle K_j

$$\mathbf{b}'_{j_0} = \mathbf{n}_{K_j} \times \mathbf{b}_{j_0} \tag{3.26}$$

$$\mathbf{b}'_{j_1} = \mathbf{n}_{K_j} \times \mathbf{b}_{j_1} \tag{3.27}$$

$$\mathbf{b}'_{j_2} = \mathbf{n}_{K_j} \times \mathbf{b}_{j_2}. \tag{3.28}$$

As with the Whitney 1-form basis function, the global $\mathbf{b}'_i$ result from combining two local basis functions in the support of $\mathbf{b}'_i$

3.2.4 Whitney Two-Forms

The last space we need in this thesis is $L^2(\Gamma)$ which is discretized by

$$L^2(\Gamma) \to S_0^{-1}(\mathcal{M}) \tag{3.29}$$

These are the piecewise constant, discontinuous functions, the Whitney 2-forms. Their basis functions are associated with the cells and globally defined to be one on the cell and zero everywhere else on the mesh

$$\mu_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in c_i \\ 0 & \text{otherwise} \end{cases}. \tag{3.30}$$

Chapter 4

Implementation

This chapter discusses the implementations necessary to conduct the experiments in this thesis. The chapter consists of four main parts, starting with a triangulation of the surface of the sphere, then the implementations of element providers for the bilinear and linear forms in the source problems, followed by the implementation of the operators, namely the Dirac operator and the three Hodge Laplacian operators, and finally, the implementation of the source problems. The whole code is based on the library Lehrfem++ [4]. Moreover, we will restrict ourselves to merely pointing out the coupling points to the code not implemented in the scope of this thesis. The source code of the cloned repository is available in the git repository [5]. Links to the documentation are provided throughout the chapter for the readers convenience.

4.1 Triangulation of $\partial\mathbb{S}$

The triangulation has to meet the requirements described in section 3.1. Moreover, the implementation has to compute arbitrarily fine meshes such that we can later use it for h-convergence studies. The key idea here is to define refinement levels. With increasing refinement level, the mesh width, which in triangulations is equivalent to the largest edge, has to decrease. Conceptually we achieve this by refining the mesh of the previous refinement level, starting with an octagon as the mesh with refinement level zero.

For the refinement, we then introduce the concept of rings, which are sets of vertices sharing the same θ-coordinate in spherical coordinates. This is also equivalent to sharing the same z-coordinate in Cartesian coordinates. However, we will stick to the spherical coordinates because there we define the rings as evenly spaced. From this follows that the refinement level zero, the octagon, has three rings. We add one new ring between every two consecutive rings in one conceptual refinement step. Furthermore, we assign

vertices to all the rings as follows: The first and last rings only contain one vertex. Then the second and second last ring contain precisely four vertices. And for all the other rings, it must hold that the number of vertices increases by four, with every ring going towards the largest in the middle. For the first two refinement steps and the upper hemisphere, this is illustrated in figure 4.1, where the new rings are colored in red for every step. This is equivalent to splitting each cell into four new cells and then projecting the newly created points on the sphere, such that they are located on the rings.

The documentation and source code is available in `SphereTriagMeshBuilder`. In the following, we describe the algorithm which directly builds a mesh for a given refinement level without the incremental refinement steps, but such that the resulting mesh is the same. For this, we start by describing the rings in more detail and then discuss how the vertices and cells are computed. Using the `lf::mesh::MeshFactroy` class in the Lehrfem++ library [4], we only have to specify the positions of vertices, and for the cells, we only have to specify the defining three vertices in the correct local order. Edges are then added automatically based on the cells. Moreover, the numbering of vertices and cells then follows the order in which they are added, and the ordering of the edges is assumed to be arbitrarily assigned but provided by the `lf::mesh::MeshFactroy` class, which suffices for the purpose of this thesis.

4.1.1 Rings

In the implementation, we heavily rely on the concept of rings, therefore we first derive the number of rings and define a helpful notation to exploit the symmetry of the upper and lower hemisphere.

Let l be the refinement level. Then we get

$$\#r = 3 + \sum_{i=1}^{l} 2^i = 2^{l+1} + 1 \tag{4.1}$$

rings in total. The reasoning is, that we start with three rings and in every refinement we place one ring in between every ring. The prove by induction can be found in appendix C, together with the derivation of the number of vertices, edges and cells for a given refinement level. These numbers are then also presented in the code documentation.

In the following we consider that the rings r and $\#r - r - 1$ are symmetric with zero indexing. Hence we define

$$\tilde{r} = \begin{cases} r & , r \leq \lfloor \frac{\#r}{2} \rfloor \\ \#r - r - 1 & , \text{else} \end{cases} \tag{4.2}$$

for further simplifications throughout the chapter.

4.1.2 Vertices

We only need to compute the positions of vertices. Their total number is given in appendix C. However for the algorithm, we only need the number of vertices per ring since we build the vertices in a loop over the rings. Hence we rely on the rings and we place the vertices equally spaced on these rings. The number of vertices per ring is given by

$$\#v_r = \begin{cases} 1 & ,\tilde{r} = 0 \\ \tilde{r} \cdot 4 & ,\text{else} \end{cases}, \tag{4.3}$$

because on the ring with index $\tilde{r} = 0$ we have only one vertex. Further, on the ring with index $\tilde{r} = 1$ we have four vertices. Then with every ring we have 4 more vertices until we reach the middle ring, as illustrated in figure 4.1.

The vertex position can then be determined with spherical coordinates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \cos(\varphi) \cdot \sin(\theta) \\ r \cdot \cos(\varphi) \cdot \sin(\theta) \\ r \cdot \cos(\theta) \end{pmatrix}. \tag{4.4}$$

With $d\theta = \frac{2\pi}{\#r}$ and $d\varphi = \frac{2\pi}{\#v_r}$ this is then equivalent to

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \cos(v_{idx} \cdot d\varphi) \cdot \sin(r_{idx} \cdot \theta) \\ r \cdot \sin(v_{idx} \cdot d\varphi) \cdot \sin(r_{idx} \cdot \theta) \\ r \cdot \cos(r_{idx} \cdot \theta) \end{pmatrix}. \tag{4.5}$$

Note that v_{idx} is the index on the ring on which v is located. By the fact that the rings are evenly spaced, θ is well defined and by defining that the first vertex v_0 on every ring has $\varphi_{v_0} = 0$ coordinate the positions for the vertices are well defined.

4.1.3 Cells

The last entities that need to be defined are the cells, because the edges will be built based on the cells as described above.

For the algorithm, cells are associated with vertices such that they imply a natural ordering according to which we can create them. The vertex to which we associate the cells are the first vertices of the ring, on which the cells have an edge, traversing the vertices on every ring in counterclockwise order. We point out that every cell only has precisely one edge on a ring. Hence, every cell will be associated with exactly one vertex. If this was not the case, either the rings would touch at some point, or the cell would need more than three edges. Moreover, we create two cells for every vertex except for the two

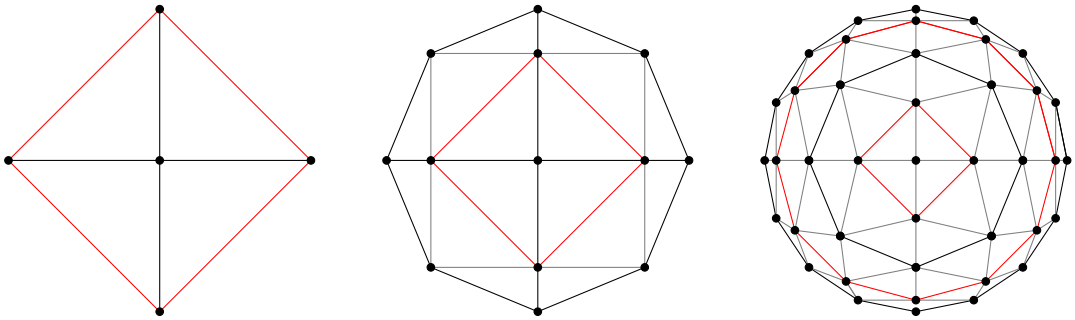

Figure 4.1: Mesh on the upper hemisphere for refinement levels 0 to 2

vertices on the first and last ring with only one vertex. With this, we have twice as many cells as there are vertices minus two.

We can then construct one cell above and one cell below the ring. For each of the two, we pick the right vertex on the ring above or below, respectively, so the result looks like in figure 4.1. For the formal definition of the cells, we introduce quarters $q \in \{0, 1, 2, 3\}$. The cell associated with vertex i in direction of a smaller ring r' and on the upper hemisphere is then given by

$$\left(v_{(r,i)}, v_{(r,i+1)}, v_{(r',i-q)}\right). \tag{4.6}$$

On the lower hemisphere we have to change the local ordering such that the local ordering remains counterclockwise

$$\left(v_{(r,i)}, v_{(r',i-q)}, v_{(r,i+1)}\right). \tag{4.7}$$

The cell in direction of the larger ring r'' on the upper hemisphere, if such a ring exists, is

$$\left(v_{(r,i)}, v_{(r'',i+1+q)}, v_{(r,i+1)}\right), \tag{4.8}$$

and on the lower hemisphere

$$\left(v_{(r,i)}, v_{(r,i+1)}, v_{(r'',i+1+q)}\right), \tag{4.9}$$

where q for the above definitions of triangles is chosen according to

$$q = \left\lfloor \frac{i}{\text{number of vertices on ring } r} \right\rfloor. \tag{4.10}$$

This already defines the whole mesh since the local orientations are implicitly given by the vertex order. With these considerations we can build a list of all vertices and cells with a double for loop over the rings and vertices in each ring.

4.2 Assembly

This section introduces all the local assembly algorithms needed for the source problems. By relying on the Lehrfem++ [4] library, we only have to implement element providers, which are classes compliant with one of the global assembly method `lf::assemble::AssembleMatrixLocally()` or `lf::assemble::AssembleVectorLocally()`. The element providers are then only responsible for computing local assembly matrices and vectors restricted on a single triangle. A local assembly matrix for a triangle K and a bilinear form $b(v,u)$ is the matrix A containing the evaluations of the bilinear form over the triangle with all the combinations of corresponding basis functions. So let v be in the span of the basis $\beta_1 = \{b_{11}, b_{12}, ..., b_{1n}\}$ and and u in the span of the basis $\beta_2 = \{b_{21}, b_{22}, ..., b_{2m}\}$. Then we have $A_{ij} = b_{|K}(b_{1j}, b_{2i})$, where we only have to include those basis functions for which $b_{1j\,|K} \not\equiv 0$ and $b_{2i\,|K} \not\equiv 0$. The local assembly vectors are defined in the same way but instead of a bilinear $b(u,u)$ we have a linear form $l(u)$ which leads to the element vector $V_i = l_{|K}(b_{2i})$. These considerations allow us only to reason about the computation of the restricted bilinear forms $b_{|K}(b_{1j}, b_{2i})$ and linear forms $l_{|K}(b_{2i})$. The element matrices as well as the element vectors can then be obtained by a loop over the basis functions.

We start with the element providers for the first Hodge Laplacian source problem, then continue with the providers for the second and the third Hodge Laplacians, and finally, we touch on the Dirac operator. The classes are mainly named according to the basis functions they use, and the operators acting on these basis functions.

Note that for all the derivations and definitions, we do not consider the exact parametrization of the sphere but only its triangulation. From this follows that φ_K, as defined in the previous chapter (3.10), is affine and hence $D\varphi_K$ constant.

4.2.1 Laplace Matrix Provider

This first element matrix provider, of which the documentation and source code is available in `LaplaceMatrixProvider`, is responsible for the computation of the bilinear form resulting from the standard Laplace-operator

$$\int_\Gamma \mathbf{grad}_\Gamma\, u \cdot \mathbf{grad}_\Gamma\, v \, d\mathbf{x}. \tag{4.11}$$

The matrix provider uses barycentric coordinate functions on triangles in a three dimensional world which leads to

$$\int_K \mathbf{grad}_\Gamma\, \lambda_i \, \mathbf{grad}_\Gamma\, \lambda_j \, d\mathbf{x} = (\mathbf{grad}\, \widehat{\lambda}_j)^T \, G_K^{-T} \, D\varphi_K^T \, D\varphi_K \, G_K^{-1} (\mathbf{grad}\, \widehat{\lambda}_i) |K|, \tag{4.12}$$

of which the equivalence follows form (3.19) and the fact that the gradients $\mathbf{grad}_\Gamma \lambda_l$ as well as $D\varphi_K$ are constant within cells. Note that, as already pointed out above, this only applies, because we defined the mapping φ_K form the reference triangle to the triangulation of the sphere. This can then be computed relying on the `lf::uscalfe` namespace which includes functions $\widehat{\lambda}_i$ on the reference triangle and their gradients $\mathbf{grad}\,\widehat{\lambda}_i$, further we can then make use of the `lf::geometry` namespace which provides us with methods to compute $|K|$ and the Jacobian-inverse-Gramian matrices $D\varphi_K\, G_K^{-1}$. These tools, provided by Lehrfem++ [4], will be the main building blocks for most of the matrix and vector providers in this chapter.

4.2.2 Mass Zero Matrix Provider

The second element matrix provider, which is available under the link `MassMatrixProvider`, computes the mass matrix for barycentric basis functions on triangles in a three-dimensional world. This is, it computes the bilinear form

$$\int_{\partial\Gamma} u\, v\, d\mathbf{x} \tag{4.13}$$

locally, which results in

$$\int_K \lambda_i\, \lambda_j\, d\mathbf{x} = \begin{cases} \frac{|K|}{6} & i = j \\ \frac{|K|}{12} & i \neq j \end{cases}, \tag{4.14}$$

and directly follows from [7, lemma 2.7.5.5]. With the above already mentioned namespace `lf::geometry` the implementation is straight forward.

4.2.3 Load Vector Provider

The element vector provider `LoadVectorProvider` implements the linear form

$$\int_K f_0\, v\, d\mathbf{x}, \tag{4.15}$$

restricted on a triangle. This can be approximated with numerical quadrature provided by the Lehrfem++ [4] library in the namespace `lf::quad`. The quadrature rules in `lf::quad` provide us with weights w_i and points $\mathbf{p}_i$ on the reference triangle $\widehat{K}$, which then leads to the following formula for the j-th element in the element vector

$$\sum_i |K|\; w_i\; f_0(\varphi_K(\mathbf{p}_i))\; \widehat{\lambda}_j(\mathbf{p}_i). \tag{4.16}$$

The function φ_K is as well already implemented in Lehrfem++ [4] and is available through the function `Global()` of the `lf::geometry::Geometry` object representing the triangle K. The caveat is that the function f_0 is not necessarily defined on K, because K is generally not a subspace of Γ. To address this issue, we require the functor passed to the load vector provider to be defined on K. This puts the burden on the caller. Such that we can only recommend projecting input coordinates $\mathbf{x} \in K$ onto Γ inside the functor. For $\Gamma = \partial\mathbb{S}$ this can be done with $\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|}$.

4.2.4 Whitney One Curl Curl Matrix Provider

The naming refers to the basis functions, which in this case are the Whitney one-forms, and the operations in the bilinear form, which is a scalar curl on the first function and a scalar curl on the second function. The documentation and source code can be found in `WhitneyOneCurlCurlMatrixProvider`. Formally we have the bilinear form

$$\int_\Gamma \operatorname{curl}_\Gamma \mathbf{u} \cdot \operatorname{curl}_\Gamma \mathbf{v} \, d\mathbf{x} \tag{4.17}$$

for the triangle K and the local basis functions $\mathbf{b}_i, \mathbf{b}_j$ this results in

$$\int_K \operatorname{curl}_\Gamma \mathbf{b}_i \cdot \operatorname{curl}_\Gamma \mathbf{b}_j \, d\mathbf{x} = s_i s_j \frac{1}{|K|}, \tag{4.18}$$

of which the derivation can be found in appendix B.1. The righthandside of (4.18) can then be implemented by getting $|K|$ as described previously and getting the s_k with the method `RelativeOrientations()`, which, when called on a mesh triangle, returns the relative orientations for the edges of that triangle. The cells inherit this method from the `lf::mesh::Entity` class in the Lehrfem++ [4] library.

4.2.5 Whitney One Mass Matrix

This matrix provider computes the mass matrix for surface vector fields, i.e. the following bilinear form

$$\int_\Gamma \mathbf{u} \cdot \mathbf{v} \, d\mathbf{x} \tag{4.19}$$

with the derivations in appendix B.2 we arrive at

$$c_{ij} := ((\mathbf{grad}\,\widehat{\lambda}_i)^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1}\, (\mathbf{grad}\,\widehat{\lambda}_j))\, \frac{|K|}{12}\, (1+\delta_{ij}) \tag{4.20}$$

$$d_{ij} := ((\mathbf{grad}\,\widehat{\lambda}_i)^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1}\, (\mathbf{grad}\,\widehat{\lambda}_j))\, \frac{|K|}{12}\, (1+\delta_{i-1\,j+1}) \tag{4.21}$$

$$\int_K \mathbf{b}_j \cdot \mathbf{b}_j\, d\mathbf{x} = s_i s_l\, (c_{i+1\,j+1} - d_{i+1\,j} - d_{j+1\,i} + c_{ij}) \tag{4.22}$$

for the entries of the assembly matrix. In the implementation, available under the link `WhitneyOneMassMatrixProvider`, we can get all the components in (4.22) the same way as in the previous subsections.

4.2.6 Whitney One Grad Matrix Provider

This matrix provider, `WhitneyOneGradMatrixProvider`, computes the element matrices of

$$\int_\Gamma \mathbf{v} \cdot \mathbf{grad}_\Gamma\, p\, d\mathbf{x}. \tag{4.23}$$

For this bilinear form we get the element matrix entries for K

$$\int_K \mathbf{b}_i \cdot \mathbf{grad}_\Gamma\, \lambda_j\, d\mathbf{x} = \begin{aligned} &\frac{s_i\,|K|}{3} \Big((\mathbf{grad}\,\widehat{\lambda}_i)^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1}\, (\mathbf{grad}\,\widehat{\lambda}_j) \\ &-(\mathbf{grad}\,\widehat{\lambda}_{i+1})^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1}\, (\mathbf{grad}\,\widehat{\lambda}_j) \Big), \end{aligned} \tag{4.24}$$

of which the derivation can be found in appendix B.3. Again, we can use the same components already discussed for the previous subsections of this chapter.

4.2.7 Whitney One Vector Provider

The vector provider of the Whitney one basis functions, given under the link `WhitneyOneVectorProvider`, with the linear form

$$\int_K \mathbf{f}_1\, \mathbf{v}\, d\mathbf{x} \tag{4.25}$$

can as well be obtained with numerical quadrature. Moreover, as in section 4.2.3, we assume that the functor providing $\mathbf{f}_1$ is well defined on K. This then leads to

$$\sum_i |K|\, w_i\, \mathbf{f}_1(\varphi_K(\mathbf{p}_i)) \cdot \mathbf{b}_j(\mathbf{p}_i) \tag{4.26}$$

for the j-th component of the element vector. All the needed components can be obtained analogously to section 4.2.3.

4.2.8 Rot Whitney One Div Matrix Provider

The matrix provider `RotWhitneyOneDivMatrixProvider` computes the bilinear form

$$\int_{\Gamma} q \operatorname{div}_{\Gamma} \mathbf{v} \, d\mathbf{x}. \tag{4.27}$$

Due to the cell wise constant basis functions for q we get the element matrix entries

$$\int_{K} \operatorname{div}_{\Gamma} \mathbf{b}_i' \, d\mathbf{x} = \int_{K} \operatorname{curl}_{\Gamma} \mathbf{b}_i \, \mathrm{d}x \tag{4.28}$$

$$= -\frac{s_i}{|K|}. \tag{4.29}$$

The second equation is given in appendix B.9 and the first equation follows form the definition of $\operatorname{curl}_{\Gamma}$ which is, the surface divergence of the rotated vector field. Together with the fact that $\mathbf{b}'$ is the rotated $\mathbf{b}$. We point out that this will result in a matrix of dimensions 3×1 because the second basis of q in (4.27) only has one constant basis function per triangle.

4.2.9 Whitney Two Mass Provider

This matrix provider, of which the documentation and implementation is given in `WhitneyTwoMassProvider` computes the mass matrix for the locally constant basis functions

$$\int_{\Gamma} u \, q \, d\mathbf{x}, \tag{4.30}$$

which gives the matrix entry

$$\int_{K} d\mathbf{x} = |K|. \tag{4.31}$$

We point out that there is only one matrix entry because of the single basis function with K in its support.

4.2.10 Whitney Two Vector Provider

The last vector provider, for the bilinear form

$$\int_{K} f_2 \, q \, d\mathbf{x}, \tag{4.32}$$

can alike the other vector providers be obtained with numerical quadrature. With the requirement for the functor providing f_2 to be defined on K, we get

$$\sum_i |K| \; w_i \; f_2(\varphi_K(\mathbf{p}_i)), \tag{4.33}$$

which is only a scalar because there is only a single constant basis function per cell. Again, all the needed components can be obtained analogously to section 4.2.3. And the implementation can be found under the link `WhitneyTwoVectorProvider`.

4.3 Operators

This section discusses the Galerkin matrices of the variational problems for the three Hodge Laplace operators and the Dirac operator. Which agrees with the Galerkin matrices of the variational problems (2.25), (2.28, 2.29) and (2.32, 2.33), without the mass terms. Further, we postpone the discussion of the right-hand side vectors of the source problems to the next chapter.

With the element matrix providers in the last section, we get the Galerkin matrix for one bilinear form with the function `AssembleMatrixLocally()` provided in lehrfem++ [4]. Hence, we restrict ourselves in this section to discuss which element providers are used and how the resulting matrices are arranged. All the classes described in the following can be found in the namespace `opertors`.

4.3.1 Hodge Laplacian Zero Form

In this problem, which is part of (2.25), we only have one bilinear form, and therefore the resulting Galerkin matrix is the Galerkin matrix we get from global assembly, which is calling the function `AssembleMatrixLocally` on the matrix element provider defined by (4.12). This is implemented in the class `WhitneyZeroHodgeLaplace`.

4.3.2 Hodge Laplacian One form

In this problem (2.28, 2.29) we have four bilinear forms excluding the mass term. These bilinear forms involve functions in two function spaces. The method `AssembleMatrixLocally()` produces a Galerkin matrix A for exactly one bilinear form. Hence, in order to obtain the full Galerkin matrix for this Hodge Laplacian we can combine the matrices of the four bilinear forms in a natural way

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \in \mathbb{R}^{(|\mathcal{E}|+|\mathcal{V}|)\times(|\mathcal{E}|+|\mathcal{V}|)}. \tag{4.34}$$

Of which we obtain $B_{11} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ by global assembly on the matrix element provider defined by (4.18). Because the storage format of those matrices is a list of triplets, to include B_{11} in B we can just copy the triplets in B_{11} to B. Then the second matrix $B_{12} \in R^{|\mathcal{E}|\times|\mathcal{V}|}$ we get with the element provider resulting from (4.24). After global assembly, we can simply add the entries of B_{12} to the matrix B by adding the triplets of B_{12} to the list of B and updating the column indexes by the additional offset $|\mathcal{E}|$. By taking a closer look at the variational problem (2.28, 2.29) we see that $B_{21} = -B_{12}^T$. Which can be included into B by swapping the row and column index in the triplets of B_{12}, negating the value and adding an offset $|\mathcal{E}|$ to the row such that the whole matrix gets placed in the bottom left corner. For the computation of $B_{22} \in R^{|\mathcal{V}|\times|\mathcal{V}|}$ we use the element matrix provider defined in (4.22). And it can be placed in B with the row and column offsets of $|\mathcal{E}|$. The implementation can then be found in the class `WhitneyOneHodgeLaplace`.

4.3.3 Hodge Laplacian two Form

This last Hodge Laplacian implies the variational problem (2.32, 2.33) without the mass term. Then analogously to the previous subsection this variational problem yields the Galerkin matrix

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(|\mathcal{E}|+|\mathcal{C}|)\times(|\mathcal{E}|+|\mathcal{C}|)}. \tag{4.35}$$

The matrix C_{11} in this case is obtained with the element matrix provider (4.22) this is becuase we have

$$\int_K \mathbf{b}_i' \cdot \mathbf{b}_j' \,\mathrm{d}x = \int_K \mathbf{b}_i^T \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{b}_j \,\mathrm{d}x \tag{4.36}$$

$$= \int_K \mathbf{b}_i \cdot \mathbf{b}_j \,\mathrm{d}x \tag{4.37}$$

Then for the matrix C_{12} we can use the matrix element provider form (4.29). Moreover, we again have $C_{12} = -C_{21}^T$. The matrix C is then built precisely the same way as in the previous section explained. The class for this problem is named `WhitneyTwoHodgeLaplace`.

4.3.4 Dirac Opeartor

For the dirac operator variational problems (2.35, 2.36, 2.37) we have three defining test and trial spaces. These lead to the matrix

$$D = \begin{pmatrix} \mathbf{0} & D_{12} & \mathbf{0} \\ D_{21} & \mathbf{0} & D_{23} \\ \mathbf{0} & D_{32} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(|\mathcal{V}|+|\mathcal{E}|+|\mathcal{C}|)\times(|\mathcal{V}|+|\mathcal{E}|+|\mathcal{C}|)}. \tag{4.38}$$

For the non zero matrix blocks we get D_{12} with the element matrix provider `WhitneyOneGradMatrixProvider` in section 4.2.6. For D_{21} we have $D_{21} = D_{12}^T$ and hence we can use the same element matrix provider. Then for $D_{23} = D_{32}^T$ we can use `RotWhitneyOneDivMatrixProvider`, section 4.2.8 , because of

$$\int_K \operatorname{div}_\Gamma \mathbf{b}_i' \, d\mathbf{x} = \int_K \operatorname{curl}_\Gamma \mathbf{b}_i \, \mathrm{d}x,$$

which follows form the definition of $\mathbf{b}_i'$. Hence, we can directly take the same resulting matrix and for the transposition we additionally have to swap the row and column index. Fitting everything together into the matrix D can then be done with the proper offsets as similar to the previous sections. For this last operator we implemented the above descriptions in `DiracOperator`.

4.4 Source Problems

The source problems in all the discussed forms consist of either the sum of the negative Hodge Laplace operator or the Dirac operator combined with a stabilization term scaled by some constant k^2 or $\imath k$ respectively. The two classes implementing these problems can be found under the links `HodgeLaplaceSourceProblems` and `DiracOperatorSourceProblem`.

For the Galerkin matrices A, B, C, D this can then be achieved by adding the Galerkin matrix obtained by the corresponding mass matrix provider. Let us denote the mass matrices M_0 obtained with the matrix provider implemented in the class `MassMatrixProvider`, M_1 obtained using the matrix provider `WhitneyOneMassMatrixProvider`, and M_2 with the `WhitneyTwoMassProvider`.

Then we get the Galerkin matrix for the first source problem with

$$A_s = A + k^2 M_0. \tag{4.39}$$

For the second source problem we get the Galerkin matrix

$$B_s = \begin{pmatrix} B_{11} + k^2 M_1 & B_{12} \\ B_{21} & B_{22} \end{pmatrix}. \tag{4.40}$$

The third source problem then gets the Galerkin matrix

$$C_s = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & +k^2 M_2 \end{pmatrix}. \tag{4.41}$$

And lastly for the Dirac opeartor we have to add all three mass matrices resulting in

$$D_s = \begin{pmatrix} \imath k M_0 & D_{12} & \mathbf{0} \\ D_{21} & \imath k M_1 & D_{23} \\ \mathbf{0} & D_{32} & \imath k M_2 \end{pmatrix}. \tag{4.42}$$

In order to optain the load vectors we denote the output vector $V_0$of global assembly with the `LoadVectorProvider` and a functor for the function f_0. Then let V_1 be the vector obtained with `WhitneyOneVectorProvider`, and a functor for the function f_1, and V_2 the vector resulting from `WhitneyTwoVectorProvider` and a functor for f_2. With these notations and the variational problems, we get the load vector for the first source problem with the zero forms

$$V_{s_A} = V_0. \tag{4.43}$$

For the second source problem with the two forms we get

$$V_{s_B} = \begin{pmatrix} V_1 \\ \mathbf{0} \end{pmatrix}. \tag{4.44}$$

And for the thrid soruce problem involving the Hodge Laplacians we then get

$$V_{s_C} = \begin{pmatrix} \mathbf{0} \\ V_2 \end{pmatrix}. \tag{4.45}$$

For the load vector of the Dirac operator, we need the same vector providers but with different functors, which gives $\vec{V}_i$ with the functor for $\vec{f}_i$. This then implies the load vector for the Dirac source problem

$$V_{s_D} = \begin{pmatrix} \vec{V}_0 \\ \vec{V}_1 \\ \vec{V}_2 \end{pmatrix}. \tag{4.46}$$

The solutions to the problems in the representation of the basis expansion coefficients for the involved discrete funciton spaces can then be obtained by solving the linear systems

$$A_s\, \mu_A = V_{s_A} \tag{4.47}$$
$$B_s\, \mu_B = V_{s_B} \tag{4.48}$$
$$C_s\, \mu_C = V_{s_C} \tag{4.49}$$
$$D_s\, \mu_D = V_{s_D}. \tag{4.50}$$

And for this we relay on the `Eigen::SparseLU` [6] solver to solve the systems in the implementation.

Chapter 5

Debugging

Due to the problem with the implementation, it is necessary to insert this chapter. Therefore, this chapter discusses the approaches we took in order to debug the not properly working parts of the code.

5.1 Problems

We first describe what part of the code did not work properly and needed a closer inspection. The experiments for the Hodge Laplacians seemed to work because the L^2-Error of the discrete solution showed a linear h-convergence. However, the code approximating the Dirac operator did not have the same behavior. With the same test functions as for the Hodge Laplacians defined in (6.5), only the solution of the first component with the Whitney zero form converged to the desired function. This behaviour is depicted in the figure 5.1. For a properly working code we would expect something similar to figure 6.3. In other words, we would expect convergence for all components.

5.2 Analytical Computations

Due to the suspicious result of the converging zero component, the first place to look for errors in the analytical computations of the solution and source functions. Since the computations for the Hodge Laplacians seem to work, we can test $-\Delta = D^2$. This test is done in the Mathematica [9] notebook testfunctionDirac.nb. It turned out to be successful. Furthermore, the computations for the Dirac operator involve the same differential operators as for Hodge Laplacians, namely the surface divergence, surface gradient, scalar-cul and vector-curl. Therefore, we assumed the analytical computations to be correct.

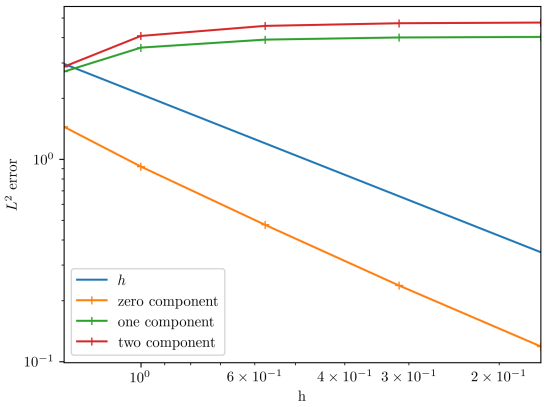

Figure 5.1: L^2-errornorm of Dirac source problems $u_h - u$ under h-refinement for erroneous code where h is denotes the mesh width.

5.3 Convergence of the Solution

The next test we conducted was a convergence test for the expression

$$\left| \|u_k\|_{L^2}^2 - \|u_{k-1}\|_{L^2}^2 \right| = \left| \int_{\partial S} (u_k - u_{k-1})\,(u_k + u_{k-1})\, d\mathbf{x} \right| \tag{5.1}$$

$$\leq \|u_k + u_{k-1}\|_{L^2}\, \|u_k - u_{k-1}\|_{L^2} \tag{5.2}$$

in which the inequality follows from the Cauchy-Schwarz inequality. The term u_k denotes the discrete solution of the k-th refinement step. The term $\|u_k - u_{k-1}\|_{L^2}$ is then expected to converge with an algebraic rate. Therefore, it is expected that the expression (5.1) also converges at an algebraic rate. The code for this can be found in `DiracConvergenceTest`. And as expected, we get algebraic convergence for all three components of the approximated function $\vec{u}_h$. This empirical convergence is depicted in figure 5.2. For the one form, we get an algebraic rate of about 1.5, and for the Whitney one and two components, we get about rate two. From this, we conclude that the numerical computations converge. But at the same time, we do not learn to what value it converges, and hence we do not know whether this is the desired function.

5.4 Bilinear Forms

The section on debugging finite element inspired the next approach methods in [7, section 3.8]. The rationale here is to test the Galerkin discretization of individual bilinear forms $b(v, u)$ by studying the convergence of the expression

$$|b(v, u) - \vec{v}_h\, B\, \vec{u}_h|. \tag{5.3}$$

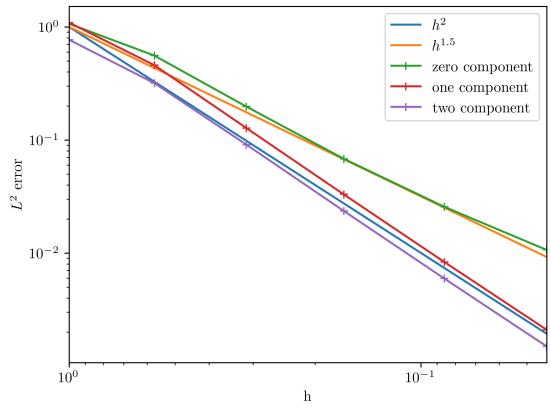

Figure 5.2: Convergence test for the Dirac Opeartor $|\|u_k\|^2 - \|u_{k-1}\|^2| \overset{k\to\infty}{\longrightarrow} 0$. For $k \to \infty$ we get for the mesh width $h \to 0$.

The bilinear form $b(v, u)$ is the analytical solution, which can be computed for known v, u, B is the Galerkin matrix of that bilinear form and $\vec{v}_h, \vec{u}_h$ are the analytical basis expansion coefficients. The vectors $\vec{v}_h, \vec{u}_h$ can easily be computed if the corresponding basis is a cardinal basis. In this case, we can just evaluate the analytical solution at the points where the basis is defined to be one. In such a case, the expression (5.3) is expected to converge with an algebraic rate.

For the Whitney one-form basis, this leads to the problem that the basis does not have the cardinal basis property. Since the functions are vector-valued, this is not possible. In order to find the analytical basis expansion coefficients, we theoretically have to solve the system such that at the edge midpoints, the analytical vector field agrees with the vector field produced by the basis. This then leads to three constraints per basis function. And gives a system with three times more equations than unknowns, of which the solution is not defined in the general case.

An alternative approach is to solve the system with least squares, which minimizes the sum of the squared residuals at the evaluated points. This approach is discussed in the following subsection.

Note that the state of the art is to use path integrals along the edges, which then give the tangential components.

5.4.1 Basis Expansion Coefficients for Whitney One-Form

For this approach, we will first study the local constraints on triangles and then by assigning equal weight to both local constraints per function from the two triangles of the support we compute the global constraint representing one row in the matrix.

We consider $K = (v_0, v_1, v_2)$ and the midpoint of the edge $i \in \{0, 1, 2\}$. Then we get the value of the discrete solution by

$$\begin{aligned} \mathbf{u}_{h\,|K}(\mathbf{x}_{\mathrm{mid}_i}) &= \mu_{i+1}\,\mathbf{b}_{i+1}(\mathbf{x}_{\mathrm{mid}_i}) + \mu_{i+2}\,\mathbf{b}_{i+2}(\mathbf{x}_{\mathrm{mid}_i}) + \mu_i\,\mathbf{b}_i(\mathbf{x}_{\mathrm{mid}_i}) && (5.4)\\ &= \mu_{i+1}\,\frac{s_{i+1}}{2}\,\mathbf{grad}\,\lambda_{i+2}(\mathbf{x}_{\mathrm{mid}_i}) \\ &\quad - \mu_{i+2}\,\frac{s_{i+2}}{2}\,\mathbf{grad}\,\lambda_{i+2}(\mathbf{x}_{\mathrm{mid}_i}) \qquad , && (5.5)\\ &\quad + \mu_i\,\frac{s_i}{2}\,(\mathbf{grad}\,\lambda_{i+1}(\mathbf{x}_{\mathrm{mid}_i}) - \mathbf{grad}\,\lambda_i(\mathbf{x}_{\mathrm{mid}_i})) \end{aligned}$$

in which we used that the barycentric coordinate functions at the edge midpoints are either zero or one half.

Let $\widetilde{K}$ the second triangle adjacent to the edge we are building the constraints for, with which we mean edge i in K. Then we can weight the two resulting vectors with weights $w_K, w_{\widetilde{K}}$ and $w_K + w_{\widetilde{K}} = 1$ which leads to the global constraint on μ

$$\mathbf{u}(\mathbf{x}_{\mathrm{mid}_e}) = w_K\,\mathbf{u}_{h\,|K}(\mathbf{x}_{\mathrm{mid}_e}) + w_{\widetilde{K}}\,\mathbf{u}_{h\,|\widetilde{K}}(\mathbf{x}_{\mathrm{mid}_e}). \tag{5.6}$$

For the implementation in the debugging experiments, we choose for simplicity $w_K = \frac{1}{2}$. We point out that each such equations gives in fact three constraints because the functions are vector fields.

The linear system is then built in the spirit of global assembly as described in [7, section 2.4]. This means, we compute the entries by iterating over the cells and adding the necessary entries to the matrix, which is realized with the class `lf::assemble::COOMatrix` in [4].

Because there is no reference for this procedure, we present as a small experiment, showing that the solution $\mathbf{u}_h$ then converges to $\mathbf{u}$, with mesh refinement. This is needed in order to get valuable solutions for the testing of bilinear forms. The experiment depicted in figure 5.3 shows an algebraic convergence of the L^2-error with the orange line. As functions, we used the experiment function in (6.4). The green line shows the minimized quantity of the sum of squared residuals at the edge midpoints, which formally is

$$\sum_{e=1}^{|\mathcal{E}|} \|\mathbf{u}_h(\mathbf{x}_{\mathrm{mid}_e}) - \mathbf{u}(\mathbf{x}_{\mathrm{mid}_e})\|^2.$$

Then the orange line show the convergence of the L^2-error. That is the convergence of

$$\|\mathbf{u}_h - \mathbf{u}\|_{L^2}$$

under h-refinement. The implementation and it's documentation can be found in `WhitneyOneBasisExpansionCoeffs`.

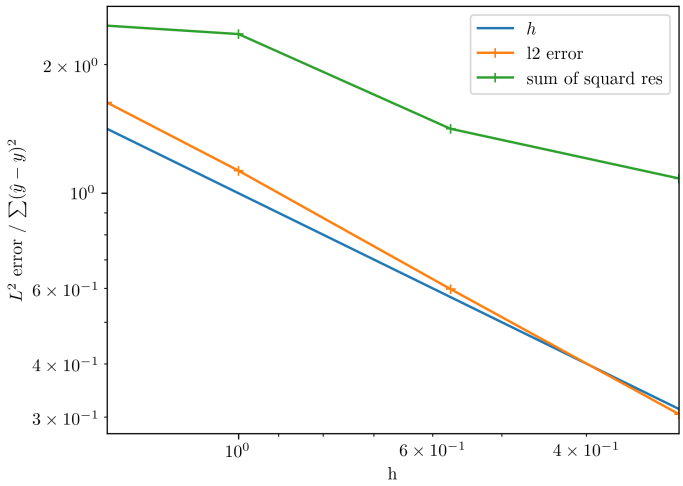

Figure 5.3: Convergence of the approximation with knowledge of the analytical solution under h-refinement with mesh width h.

5.4.2 Bilinear Form Convergence

With the basis expansion coefficients μ_1 for the Whitney one form as computed in the previous section and the basis expansion coefficients μ_2 for the Whitney two forms, which we get by evaluating the analytical solution at the cell midpoints, we can compute the experiment in (5.3). The computation of this can be found in the Mathematica [9] notebook testfunctionDirac.nb. The code can be found in `WhitneyOneCurlTest`. The bilinear form we test is given by

$$b(q, \mathbf{v}) = \int_\Gamma q \operatorname{curl}_\Gamma \mathbf{v} \, d\mathbf{x}, \tag{5.7}$$

of which the Galerkin matrix agrees with the one in (4.27). Because of the equation (4.28). Hence in order to build the Galerkin matrix we can use the `RotWhitneyOneDivMatrixProvider`, and then take the resulting Galerkin matrix.

For the experiment we use the functions

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5.8}$$

$$q(\mathbf{x}) = \sin(z) \tag{5.9}$$

$$\mathbf{v}(\mathbf{x}) = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}. \tag{5.10}$$

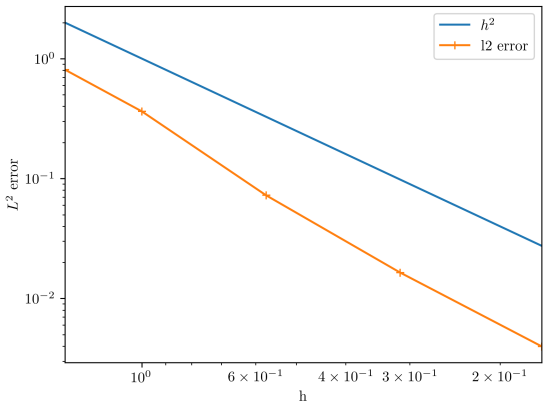

Figure 5.4: Convergence test for bilinear form (5.3) under h-refinement where h denotes the mesh width.

For this we finally got the convergence plot in figure 5.4. The convergence in this plot showed that the L^2-error norm converges faster than algebraic rate two. But this result might be misleading, because the convergence of the approximated basis expansion coefficients and the actual convergence of this experiment might add up.

This was the experiment, which revealed the error. It turned out that the error was a wrong sign in the computation of $\operatorname{curl}_\Gamma \mathbf{b}$ which lead to the wrong equation $\operatorname{curl}_\Gamma \mathbf{b} = -\operatorname{div}_\Gamma \mathbf{b}'$. This sign error was cancelled out for the Hodge Laplacian, because there we directly used $\operatorname{div}_\Gamma \mathbf{b}'$ in the computations.

Chapter 6

Experiments

In this chapter, we discuss the experiments, run on the implementation of the source problems. First, we introduce the chosen test functions, the stability of the solutions for a range of k. In other words we inspect how stable the rate of convergence is, when changing the value k. Furthermore, we will inspect the convergence in terms of h-refinement and the L^2-errors of the source problems with a stable k. The experiments are then concluded by comparing the solutions computed with Dirac operator and Hodge Laplacians.

6.0.1 Error

In this short subsection, we describe how we measure the error of the discrete solution. So let u_h be the discrete solution, and u the corresponding analytic solution. Then we get the L^2 error

$$\|u_h - u\| = \sqrt{\int_{\mathcal{M}} |u_h(\mathbf{x}) - u(\mathbf{x})|^2 \, d\mathbf{x}}. \tag{6.1}$$

In the following we then approximate the integral under the root with local quadrature on the triangles in $\mathcal{M}$ and then summing up over the results. We implement such a functionality in the function `L2norm()` of the namespace `post_processing`. The function takes a functor argument for the function $u_h - u$ and a functor, computing the square. We face the same issue here as with the vector provider classes, namely, that the domains of u_h and u are not the same. This detail is once more resolved by projecting the points in $\mathcal{M}$ onto ∂S.

6.1 Experiment Functions

In order to conduct experiments, we first chose experiment functions defined on ∂S. The criteria were only smoothness, such that the differential operators

are defined and non-vanishing load functions. Smoothness has the additional advantage that smooth functions generally have better convergence properties when approximated with Finite Element Methods [7, chapter 3]. Based on this we then chose the test functions for the four source problems (2.20, 2.21, 2.22, 2.23),

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \tag{6.2}$$

$$u_0(\mathbf{x}) = u_2(\mathbf{x}) = \sin(x_0) + \cos(x_1) + \sin(x_2) \tag{6.3}$$

$$\mathbf{u}_1 = (I_3 - \mathbf{n}\mathbf{n}^T) \begin{pmatrix} \sin(x_1) \\ \sin(x_2) \\ \sin(x_0) \end{pmatrix} \tag{6.4}$$

$$\vec{u} = \begin{pmatrix} u_0 \\ \mathbf{u}_1 \\ u_2 \end{pmatrix}. \tag{6.5}$$

The load functions resulting when we plug these function in the four source problems

$$-\Delta_0 u + k^2 u = f_0 \tag{6.6}$$

$$-\Delta_1 \mathbf{u} + k^2 \mathbf{u} = \mathbf{f}_1 \tag{6.7}$$

$$-\Delta_2 u + k^2 u = f_2 \tag{6.8}$$

$$D\vec{u} + \imath k \vec{u} = \vec{f} \tag{6.9}$$

are then computed with Mathematica [9]. The full computations and the source functions can be found in the documents testfunctionsLaplace.nb and testfunctionDirac.nb. We omit to state the resulting in explicit terms of the load functions because of their complexity. Moreover, we point out that the function $\mathbf{u}_1$ needs to be a tangential vector field, and hence the complexity increases rapidly when applying the differential operators.

6.2 Stability of the solution

As already introduced, in this experiment we are numerically looking how stable the solution converges for a range of values of the positive parameter k. In order to determine this, we iterated over a list of evenly spaced values for k and approximated the L^2 norm of the error function. For the discrete solution u_h, we get the error by $\|u_h - u\|_{L^2}$. The rates of algebraic convergence are computed for each refinement step with the formulas in [7, section 3.2.2].

In order to assess how well the computed rates of algebraic convergence can be trusted, we measure how much they estimated rates change for each

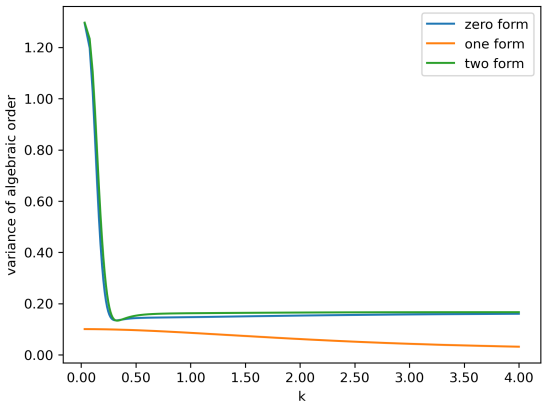

Figure 6.1: Empirical variance of the approximated rates of algebraic convergence for the Hodge Laplacian source problems as a function of k

iteration step. Because in the formula [7, section 3.2.2] we estimate these rates based the numerical solution of two consecutive refinement steps. We then regard the result of the computation, for two refinement step as a random variable X. With this definition of X, the variance of X measures exactly the fluctuation around the mean. Note that under the assumption that our algorithm converges stably to a solution, the randomness only contains numerical differences between the refinement steps such as numerical errors or the regularity of the individual mesh refinement steps. For this case we then expect a rather small variance of the variable X. On the other hand, if the problem is singular, for example by choosing $k = 0$, then X is can take any value and we expect a much higher variance. For each value of the parameter k this random variable is different and therefore we will denote X_k the random variable to express that it is dependent on k

For the experiment we chose 100 evenly spaced $k^2 \in [0.001, 16)$ and for each k we computed the solution for 5 refinement levels. Then we computed the resulting four empirical algebraic orders and regarded them as sampled data of the random variable X_k. Further we used the samples to compute a empirical variance of of X_k for each k. These empirical variances of the estimated rates of algebraic convergence are then plotted in 6.1 as a function of k.

The plot shows that for values of k over about 0.3, the solution seems to converge with a stable rate.

In figure 6.2 we have the same experiment for the Dirac operator, which seems to have stable convergence for minimal values of k already compared to the Hodge Laplacian experiment. We point out once more that although the two plots in firgures 6.1 and 6.2 have difference ranges for k the stabilization terms in the source problems will have the same weights because in the

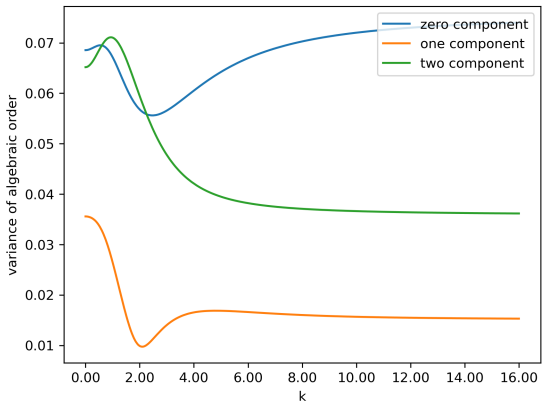

Figure 6.2: Empirical variance of the approximated rates of algebraic convergence for the Dirac operator source problems as a function of k

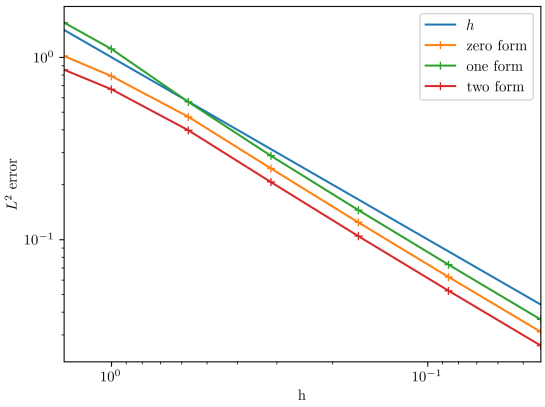

Figure 6.3: L^2-Norm of Hodge Laplacian source problems $u_h - u$ under h-refinement

Hodge Laplacian source problems we use k^2 for the scaling of the mass term.

From this, we conclude that taking a value of $k = 0.5$ will give stable results for the convergence studies of the test functions for both source problems.

6.3 H-Convergence study

This section analyzes the convergence of the discrete solution under h-refinement. We do it for the chosen test functions in section 6.1 and for the value $k = 0.5$. We start with the source problems of the three Hodge Laplacians and then analyze the individual components of the Dirac operator.

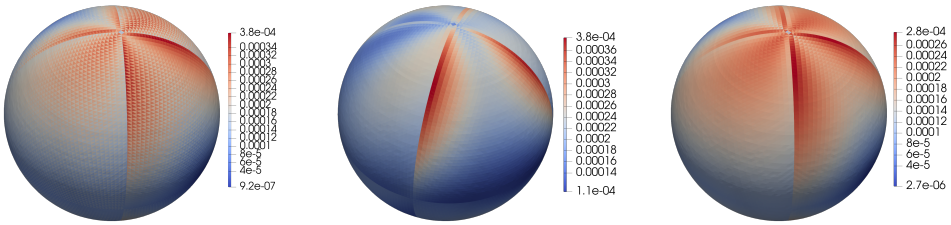

Figure 6.4: Cellwise contributions to the squared L^2 error norm for the Lapalce Opeartors zero form (left), one form (middle) and two form (right). For the refinement level six which has $h = 0.031$.

6.3.1 Hodge Laplacians

In figure 6.3 we have an h-refinement experiment of the above-described test functions for the Hodge Laplacians. The study involves seven refinement levels with mesh widths form about 1.4 to 0.031. We observe that the experiments yield an algebraic convergence of rate one. This holds for all three Hodge Laplacians. Moreover, for this particular experiment the approximation with the piecewise constant basis, the two-form gives a slightly better approximation than the solution of the zero-form, with piecewise linear basis function, solving the same differential equation.

When looking at the cell wise contributions to the squared L^2-error, We observe a pattern depicted in figure 6.4. Namely, the errors seem more severe in some areas, separated quite clearly by straight lines. However, on a closer look, we realize that these separation lines lay precisely on the border of the quarters, according to which we defined our mesh. The pattern indicates the mesh structure has an influence, at least on the error distribution on the sphere.

6.3.2 Dirac Operator

Similar to the above convergence study of the three Hodge Laplacians, we conduct an h-convergence study for the Dirac operator on the surface of the 3-sphere ∂S. The source problem of the Dirac operator produces a result $\vec{u}_h$ with three components

$$\vec{u}_h = \begin{pmatrix} \vec{u}_{0h} \\ \vec{u}_{1h} \\ \vec{u}_{2h} \end{pmatrix}. \tag{6.10}$$

The individual components live in different function spaces and hence we analyse them separately. However, by the definition of the problem and the way we solve it described in chapter 4, we get that the solutions are strongly connected and hence we expect similar behaviour in all three components. In figure 6.5 we then observe the expected behaviour with the same convergence

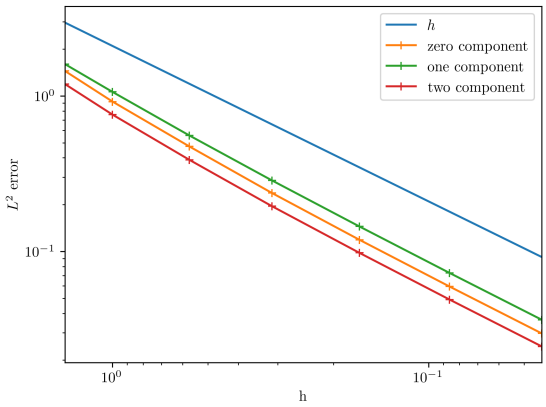

Figure 6.5: L^2-error $u_h - u$ of the Dirac Opeartor source problem under h-refinement

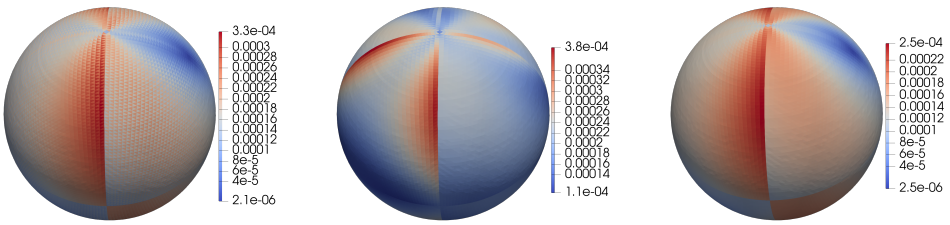

Figure 6.6: Cell wise contributions to the squared L^2-error norm for the Dirac operators zero form (left), one form (middle) and two form (right). For the refinement level six which has $h = 0.031$.

for all three components. And similar to the Hodge Laplace operator, we again observe the same pattern in the local contributions for the squared error norm. These local contributions are given in figure 6.6. Again the pattern with more severe errors along the boundaries of the quarters is visible.

6.4 Relationship Hodge Laplacian and Dirac Operator

This chapter discusses the relationship $-\Delta + k^2 = (D - \imath k)(D + \imath k)$. From a analytical perspective we get

$$(-\Delta + k^2)\,\vec{u} = (D - \imath k)\,\vec{f} \tag{6.11}$$

$$\iff (D + \imath k)\,\vec{u} = \vec{f}. \tag{6.12}$$

With this we can solve the system in the two different ways (6.11, 6.12) and compare the two solutions. For this we first compare the convergence of the two solutions individually with the load function

$$\vec{f} \in H^1(\partial\mathbb{S}) \times \mathbf{H}(\mathrm{curl}_\Gamma, \partial\mathbb{S}) \times L^2(\partial\mathbb{S}),$$

given in equation (6.9). The computations can be found in testfunctionsHodgeAndDirac.nb. The two above equations (6.11, 6.12) can then be solved

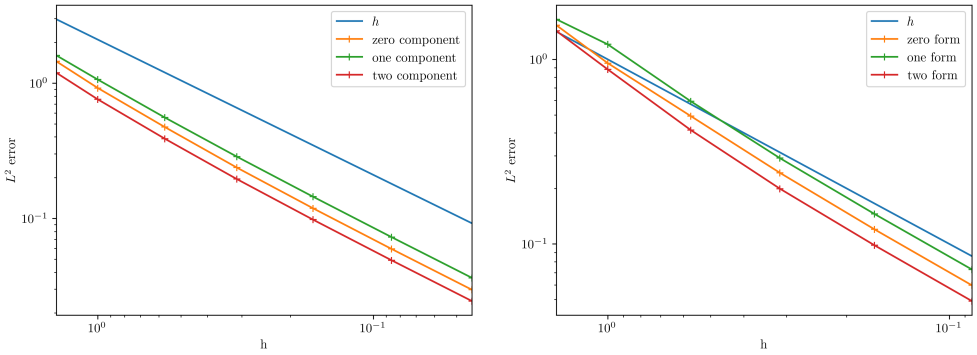

Figure 6.7: L^2-error of Hodge Laplacian (right) and Dirac operator (left) source problems with the same solution

with the finite element approach discussed in chapters 2 to 4. For the modified Hodge Laplace problem (6.11) we have the righthandside function

$$\vec{g} = \begin{pmatrix} g_0 \\ \mathbf{g}_1 \\ g_2 \end{pmatrix} := (D - \imath k)\, \vec{f} \tag{6.13}$$

of which, in the variational problems (2.25 - 2.33) the components $g_0, \mathbf{g}_1, g_2$ are used instead of $f_0, \mathbf{f}_1, f_2$. From this it also follows how the functions are used in the discretizations.

The code for these experiments is available in `HodgeAndDiracExperiment`. The individual convergence results are visualized in figure 6.7. The left figure is exactly same convergence study as in figure 6.5 and the right side shows a similar convergence to figure 6.5. This implies, both solutions converge with an algebraic rate of one. This is also what we expected when considering the h-convergence studies in the previous section.

In the next experiment we compared the solutions directly and plot the L^2-norm of the difference between the two solutions in figure 6.8. More formally let u_{dk} be the solution using the Dirac operator and u_{lk} the solution for the Hodge Laplacian, both for the k$^{\text{th}}$ refinement level. The measured quantity is $\|u_{dk} - u_{lk}\|_{L^2}$. This difference converges with an algebraic rate of two for all three components of the solution. This result shows that the Dirac operator source problem can be solved very well with the Hodge Laplace operator, by modifying the load function.

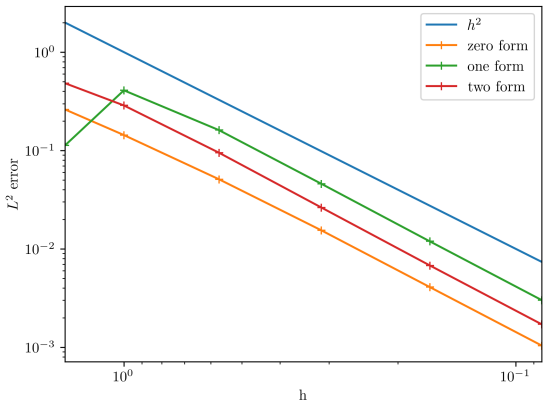

Figure 6.8: L^2-norm of difference function between the two solutions with Hodge Laplacian and Dirac operators

Chapter 7

Conclusions

The main goals of the thesis were the convergence studies of the Dirac Operator and the Hodge Laplacians. For all the operators, we examined source problems with stabilization terms. Moreover, we found that for our test functions defined in section 6.1, the discrete solution convergence with an empirical rate of one. By inspecting the cell-wise error contributions, we noticed that the structure of the sphere mesh, more precisely division into four quarters, becomes visible. The hypothesis for this behavior is that in the region of the separating lines of the quarters, the triangles become wider and hence have poorer regularity. Which then leads to an accentuation of errors in these cells. The relationship between the Dirac operator and the Hodge Laplacian then lead to the conclusion, that the two methods converge with the same rate, namley an algebraic rate of one. And the difference of the two solutions converges with an algebraic rate of two.

In order to conduct the experiments, we first had to implement the discretization of the source problems. Based on the Lehrfem++ [4] library, this involved providing the element providers for the bilinear forms in the variational forms of the source problems. Next, we combined these element providers in order to build linear systems of equations of which the solutions are the basis expansion coefficients for our discrete vector fields.

Here, we directly get the connection to the additional chapter about debugging, which became necessary due to a wrong computation leading to a sign error in the linear system for the Dirac source problem. In the debugging chapter, the test for the analytical computations in Mathematica [9] lead to no further insight. Then a test for convergence of the discretized solution showed that the solution converged but not to what function it converged. The last test, which revealed the bug, tested the bilinear form by comparing the analytical solution of a bilinear form to the discrete solutions multiplied with the Galerkin matrix.

The thesis has potential improvement in the triangulation of the sphere, for which one can build more regular meshes. Although this will probably not lead to a major improvement because the mesh is quite regular on most of the sphere except for the quarter boundaries. Further work might also consider higher-order implementations of the Whitney forms and convergence studies on other manifolds, such as a torus.

Appendix A

Derivation of Variational Problems

In this section we derive the variational problems in section 2.5 from the source problems in section 2.4

First we need greens theorem for the surface differential operators which can be derived form greens theorem [2, Section 13.3.3.3]

$$\int_\Omega \mathbf{j} \cdot \mathbf{grad}\, u \; d\mathbf{x} = -\int_\Omega u \operatorname{div} \mathbf{j} \; d\mathbf{x} + \int_{\partial\Omega} u\, \mathbf{j} \; dS \tag{A.1}$$

Then we show that the same formula holds for the differential operators $\mathbf{grad}_\Gamma$ and $\operatorname{div}_\Gamma$ first of all we make the assumption that $\partial\Omega = \emptyset$ which simplifies (A.1) to

$$\int_\Gamma \mathbf{j} \cdot \mathbf{grad}_\Gamma\, u \; d\mathbf{x} = -\int_\Gamma u \operatorname{div}_\Gamma \mathbf{j} \; d\mathbf{x}, \tag{A.2}$$

for a tangential $\mathbf{j}$.

With (A.2) we are then able to derive the variational problems. For the first source problem (2.20 we have with the fundamental lemma of variational calculus [7, Lemma 1.5.1.13]

$$-\Delta_0 u + k^2 u = f_0 \tag{A.3}$$

$$\iff -\int_\Gamma \Delta_0 u\, v \; d\mathbf{x} + k^2 \int_\Gamma u\, v \; d\mathbf{x} = \int_\Gamma f_0\, v \; d\mathbf{x} \quad \forall v \in H^1(\Gamma) \tag{A.4}$$

$$\overset{(A.2)}{\iff} \int_\Gamma \mathbf{grad}_\Gamma u \cdot \mathbf{grad}_\Gamma v \; d\mathbf{x} + k^2 \int_\Gamma u\, v \; d\mathbf{x} = \int_\Gamma f_0\, v \; d\mathbf{x} \quad \forall v \in H^1(\Gamma). \tag{A.5}$$

Then for the second source problem (2.21) we get in the same manner with $p = \operatorname{div} \mathbf{u} \in H^1(\Gamma)$

$$\forall \mathbf{v} \in H(\operatorname{curl}_\Gamma, \Gamma) \tag{A.6}$$

$$-\Delta_1 \mathbf{u} + k^2 \mathbf{u} = \mathbf{f}_1 \tag{A.7}$$

$$\iff -\int_\Gamma \Delta_1 \mathbf{u} \cdot \mathbf{v} \, d\mathbf{x} + k^2 \int_\Gamma \mathbf{u} \cdot \mathbf{v} \, d\mathbf{x} = \int_\Gamma \mathbf{f}_1 \cdot \mathbf{v} \, d\mathbf{x} \tag{A.8}$$

$$\begin{aligned} \overset{(A.2)}{\iff} & \int_\Gamma \operatorname{curl}_\Gamma \mathbf{u} \cdot \operatorname{curl}_\Gamma \mathbf{v} \, d\mathbf{x} + k^2 \int_\Gamma \mathbf{u} \cdot \mathbf{v} \, d\mathbf{x} \\ & + \int_\Gamma \mathbf{v} \cdot \mathbf{grad}_\Gamma \, p \, d\mathbf{x} = \int_\Gamma \mathbf{f}_1 \cdot \mathbf{v} \, d\mathbf{x}. \end{aligned} \tag{A.9}$$

Moreover we get the constraint (2.29) directly from (A.2)

Then for the third source problem (2.22) we have with $\mathbf{j} = \mathbf{grad}_\Gamma \, u \in H(\operatorname{div}_\Gamma)$

$$\int_\Gamma \mathbf{j} \cdot \mathbf{v} \, d\mathbf{x} \overset{(A.2)}{=} -\int_\Gamma u \operatorname{div}_\Gamma \mathbf{v} \, d\mathbf{x} \qquad \forall \mathbf{v} \in H(\operatorname{div}_\Gamma, \Gamma). \tag{A.10}$$

Then the second constraint can again be derived with the fundamental lemma of variational calculus [7, Lemma 1.5.1.13]

$$-\Delta_2 u + k^2 u = f_0 \tag{A.11}$$

$$\iff -\int_\Gamma \Delta_0 u \, q \, d\mathbf{x} + k^2 \int_\Gamma u \, q \, d\mathbf{x} = \int_\Gamma f_2 \, q \, d\mathbf{x} \qquad \forall q \in L^2(\Gamma) \tag{A.12}$$

$$\iff -\int_\Gamma \operatorname{div}_\Gamma \mathbf{j} \, q \, d\mathbf{x} + k^2 \int_\Gamma u \, q \, d\mathbf{x} = \int_\Gamma f_2 \, q \, d\mathbf{x} \qquad \forall q \in L^2(\Gamma). \tag{A.13}$$

Appendix B

Derivation of discretized biliner forms

This appendix contains derivations for formulas in chapter 4.

B.1 Derivation of (4.18)

$$\int_K \operatorname{curl}_\Gamma \mathbf{b}_i \cdot \operatorname{curl}_\Gamma \mathbf{b}_j \, d\mathbf{x} = s_i s_j \frac{1}{|K|}.$$

We remark, that the functions $\operatorname{curl} \mathbf{b}_{j_l}$ are constant on the triangle K_j. Then the $\operatorname{curl}_\Gamma$ is defined to be the surface curl. Hence with the restriction on the triangle K_j we can reduce the mathematical computations to arbitrary triangles in a two dimensional domain. In the following we use the index i modulo 3. This is $i+3 = i$ and $i = 1$ then $i+2 = 0$.

$$\mathbf{b}_i = \begin{pmatrix} b_{i\,x} \\ b_{i\,y} \end{pmatrix} \tag{B.1}$$

$$\operatorname{curl} \mathbf{b}_i = s_i \left(-d_x \, b_{i\,y} + d_y \, b_{i\,x}\right) \tag{B.2}$$

$$\begin{aligned} &= -s_i \left(d_x \left(\lambda_i \, \mathbf{grad} \, \lambda_{i+1} - \lambda_{i+1} \, \mathbf{grad} \, \lambda_i\right)_y \right. \\ &\quad \left. - d_y \left(\lambda_i \, \mathbf{grad} \, \lambda_{i+1} - \lambda_{i+1} \, \mathbf{grad} \, \lambda_i\right)_x\right) \end{aligned} \tag{B.3}$$

$$= -2 \, s_i \left(d_x \, \lambda_i \, d_y \, \lambda_{i+1} - d_y \, \lambda_{i+1} \, d_x \, \lambda_i\right), \tag{B.4}$$

with $K = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2)$ we get an alternative definition of λ_i from [7, section 2.4.5.1]

$$\mathbf{a}_i = \begin{pmatrix} a_{i\,x} \\ a_{i\,y} \end{pmatrix} \tag{B.5}$$

$$\lambda_i = \frac{1}{2\,|K|} (\mathbf{x} - \mathbf{a}_{i+1}) \cdot \begin{bmatrix} a_{i\,y} - a_{i+1\,y} \\ a_{i+1\,x} - a_{i\,x} \end{bmatrix} \tag{B.6}$$

$$\mathbf{grad} \, \lambda_i = \frac{1}{2\,|K|} \begin{bmatrix} a_{i\,y} - a_{i+1\,y} \\ a_{i+1\,x} - a_{i\,x} \end{bmatrix}, \tag{B.7}$$

with this we conclude

$$\operatorname{curl}_\Gamma \mathbf{b}_i = -\frac{s_i}{2\,|K|^2}\left((a_{i\,y}-a_{i+1\,y})(a_{i+2\,x}-a_{i+1\,x})-(a_{i+2\,y}-a_{i+1\,y})(a_{i\,x}-a_{i+1\,x})\right) \tag{B.8}$$

$$= -s_i\frac{1}{|K|}, \tag{B.9}$$

which leads to (4.18)

B.2 Derivation of (4.22)

$$c_{i\,j} := ((\mathbf{grad}\,\widehat{\lambda}_i)^T\,G_K^{-T}\,D\varphi_K^T\,D\varphi_K\,G_K^{-1}\,(\mathbf{grad}\,\widehat{\lambda}_j))\,\frac{|K|}{12}\,(1+\delta_{i\,j})$$

$$d_{i\,j} := ((\mathbf{grad}\,\widehat{\lambda}_i)^T\,G_K^{-T}\,D\varphi_K^T\,D\varphi_K\,G_K^{-1}\,(\mathbf{grad}\,\widehat{\lambda}_j))\,\frac{|K|}{12}\,(1+\delta_{i-1\,j+1})$$

$$\int_K \mathbf{b}_j\cdot\mathbf{b}_j\,d\mathbf{x} = s_i s_l\left(c_{i+1\,j+1}-d_{i+1\,j}-d_{j+1\,i}+c_{i\,j}\right).$$

For its derivation we use (3.19) in the last step

$$\int_K \mathbf{b}_i \cdot \mathbf{b}_l \,\mathrm{d}x = \int_K s_i \left(\lambda_i \,\mathbf{grad}_\Gamma\, \lambda_{i+1} - \lambda_{i+1}\,\mathbf{grad}_\Gamma\, \lambda_i\right) \cdot s_l \left(\lambda_l \,\mathbf{grad}_\Gamma\, \lambda_{l+1} - \lambda_{l+1}\,\mathbf{grad}_\Gamma\, \lambda_l\right) \mathrm{d}x \tag{B.10}$$

$$= s_i\, s_l \left(\int_K \lambda_i \,\mathbf{grad}_\Gamma\, \lambda_{i+1} \cdot \lambda_l \,\mathbf{grad}_\Gamma\, \lambda_{l+1} \,\mathrm{d}x - \int_K \lambda_i \,\mathbf{grad}_\Gamma\, \lambda_{i+1} \cdot \lambda_{l+1} \,\mathbf{grad}_\Gamma\, \lambda_l \,\mathrm{d}x \right.$$
$$\left. - \int_K \lambda_{i+1} \,\mathbf{grad}_\Gamma\, \lambda_i \mathbf{grad}_\Gamma(\lambda_{l+1})\lambda_l \,\mathrm{d}x + \int_K \lambda_{i+1} \,\mathbf{grad}_\Gamma\, \lambda_i \cdot \lambda_{l+1} \,\mathbf{grad}_\Gamma\, \lambda_l \,\mathrm{d}x \right) \tag{B.11}$$

$$= s_i s_l \left(\mathbf{grad}_\Gamma\, \lambda_{i+1} \cdot \mathbf{grad}_\Gamma\, \lambda_{l+1} \int_K \lambda_i\, \lambda_l \,\mathrm{d}x - \mathbf{grad}_\Gamma\, \lambda_{i+1} \cdot \mathbf{grad}_\Gamma\, \lambda_l \int_K \lambda_i\, \lambda_{l+1} \,\mathrm{d}x \right.$$
$$\left. - \mathbf{grad}_\Gamma\, \lambda_i \cdot \mathbf{grad}_\Gamma\, \lambda_{l+1} \int_K \lambda_{i+1}\, \lambda_l \,\mathrm{d}x + \mathbf{grad}_\Gamma\, \lambda_i \cdot \mathbf{grad}_\Gamma\, \lambda_l \int_K \lambda_{i+1}\, \lambda_{l+1} \,\mathrm{d}x \right) \tag{B.12}$$

$$= s_i s_l \left((\mathbf{grad}_\Gamma\, \widehat{\lambda}_{i+1})^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1} (\mathbf{grad}_\Gamma\, \widehat{\lambda}_{l+1}) \int_K \lambda_i\, \lambda_l \,\mathrm{d}x \right.$$
$$- (\mathbf{grad}_\Gamma\, \widehat{\lambda}_{i+1})^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1} (\mathbf{grad}_\Gamma\, \widehat{\lambda}_l) \int_K \lambda_i\, \lambda_{l+1} \,\mathrm{d}x$$
$$- (\mathbf{grad}_\Gamma\, \widehat{\lambda}_i)^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1} (\mathbf{grad}_\Gamma\, \widehat{\lambda}_{l+1}) \int_K \lambda_{i+1}\, \lambda_l \,\mathrm{d}x$$
$$\left. + (\mathbf{grad}_\Gamma\, \widehat{\lambda}_i)^T\, G_K^{-T}\, D\varphi_K^T\, D\varphi_K\, G_K^{-1} (\mathbf{grad}_\Gamma\, \widehat{\lambda}_l) \int_K \lambda_{i+1}\, \lambda_{l+1} \,\mathrm{d}x \right) \tag{B.13}$$

The formula (B.13) is then equivalent to (4.22) with use of [7, lemma 2.7.5.5].

B.3 Derivation of (4.24)

$$\int_K \mathbf{b}_i \, \mathbf{grad}_\Gamma \lambda_j \, \mathrm{d}x = \int_K s_i (\lambda_i \, \mathbf{grad}_\Gamma \lambda_{i+1} - \lambda_{i+1} \, \mathbf{grad}_\Gamma \lambda_i) \cdot \mathbf{grad}_\Gamma \lambda_j \, d\mathbf{x} \quad \text{(B.14)}$$

$$= s_i \left(\mathbf{grad}_\Gamma \lambda_{i+1} \cdot \mathbf{grad}_\Gamma \lambda_j \int_K \lambda_i d\mathbf{x} - \mathbf{grad}_\Gamma \lambda_i \cdot \mathbf{grad}_\Gamma \lambda_j \int_K \lambda_{i+1} \, d\mathbf{x} \right) \quad \text{(B.15)}$$

$$= s_i \left(\mathbf{grad}_\Gamma \lambda_{i+1} \cdot \mathbf{grad}_\Gamma \lambda_j - \mathbf{grad}_\Gamma \lambda_i \cdot \mathbf{grad}_\Gamma \lambda_j \right) \frac{|K|}{3} \quad \text{(B.16)}$$

$$= \frac{s_i \, |K|}{3} \Big(D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_{i+1} \right) \cdot D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_j \right) - D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_i \right) \cdot D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_j \right) \Big) \quad \text{(B.17)}$$

$$= \frac{s_i \, |K|}{3} \Big(\left(\mathbf{grad}_\Gamma \widehat{\lambda}_{i+1} \right)^T G_K^{-T} \, D\varphi_K^T \, D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_j \right) - \left(\mathbf{grad}_\Gamma \widehat{\lambda}_i \right)^T G_K^{-T} \, D\varphi_K^T \, D\varphi_K \, G_K^{-1} \left(\mathbf{grad}_\Gamma \widehat{\lambda}_j \right) \Big) . \quad \text{(B.18)}$$

In this derivation we again used the fact hat $\mathbf{grad}_\Gamma \lambda_i$ is constant, (3.19) and [7, lemma 2.7.5.5].

Appendix C

Number of entities in the Triangulation of the sphere

C.1 Number of Rings

In every refinement step we add a ring between two existing rings. As defined we then have three rings for refinement level zero, $|R_0| = 3$ using the notation R_j for the set of Rings in refinement level j. This leads to the formula

$$\begin{aligned} |R_k| &= |R_{k-1}| + |R_{k-1}| - 1 && \text{(C.1)} \\ &= 2\,|R_{k-1}| - 1 && \text{(C.2)} \\ &= 2^{k+1} + 1 && \text{(C.3)} \end{aligned}$$

Then we prove (C.3) by induction. We have that the equation holds for level zero by $2^1 + 1 = 3$. Then we get

$$\begin{aligned} |R_{k+1}| &= 2\,|R_k| - 1 && \text{(C.4)} \\ &= 2\,(2^{k+1} + 1) - 1 && \text{(C.5)} \\ &= 2^{k+2} + 1 && \text{(C.6)} \end{aligned}$$

which conludes the prove.

C.2 Number of Vertices

For the number of vertices we get by definition 2 for the first two rings, then 4 for the second and second last ring and incrementally 4 more vertices for every ring towards the middle. For refinement level k this gives the formula

$$|V_k| = 2 + \sum_{i=1}^{\lfloor |R_k|/2 \rfloor} 4i + \sum_{i=1}^{\lfloor |R_k|/2 - 1 \rfloor} 4i \qquad \text{(C.7)}$$

Further we know that $|R_k| = 2^{k+2} + 1$ is odd for every k. This then leads to

$$|V_k| = 2 + \sum_{i=1}^{\lfloor |R_k|/2 \rfloor} 4i + \sum_{i=1}^{(|R_k|-1)/2} 4i \tag{C.8}$$

$$= 2 + 4\left(\frac{(|R_k|+1)/2 \cdot (|R_k|-1)/2}{2}\right) + 4\left(\frac{(|R_k|-1)/2 \cdot (|R_k|-3)/2}{2}\right) \tag{C.9}$$

$$= 2 + \left(\frac{(|R_k|+1)(|R_k|-1)}{2}\right) + \left(\frac{(|R_k|-1)(|R_k|-3)}{2}\right) \tag{C.10}$$

$$= 2 + \left(2\,|R_k|^2 - 4\,|R_k| + 2\right) \tag{C.11}$$

$$= 2^{2k+2} + 2 \tag{C.12}$$

C.3 Number of Cells

By the construction of the triangulation we get two cells for every vertex except for the two cells in the first and last ring. This directly leads to

$$|C_k| = 2 \cdot |V_k| - 4 \tag{C.13}$$

$$= 2^{2k+3} \tag{C.14}$$

C.4 Number of Edges

The edges are constructed automatically but by the pattern of the cells and vertices we defined, we get that there are always six edges adjacent to one vertex except for the first six vertices to which only 4 edges are adjacent. This leads to

$$2\,|E_k| = 6\,|V_k| - 12 \tag{C.15}$$

$$\implies |E_k| = \frac{3}{2}\,|V_k| - 3 \tag{C.16}$$

$$= 3\,(2^{2k+2} + 1) - 3 \tag{C.17}$$

$$= 3 \cdot 2^{2k+2} \tag{C.18}$$

Bibliography

[1] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica*, 15:1–155, 2006.

[2] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Frankfurt am Main, 2020.

[3] Annalisa Buffa and Ralf Hiptmair. *Galerkin Boundary Element Methods for Electromagnetic Scattering*, pages 83–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[4] Raffael Casagrande and Ralf Hiptmair. LehrFem++. https://github.com/craffael/LehrFEMpp, 2022.

[5] Nico Graf. LehrFem++. https://github.com/ncograf/hldo-sphere, 2022.

[6] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[7] Ralf Hiptmair. *Numerical Methods for Partial Differential Equations*. 2022.

[8] Ralf Hiptmair and Jörg Ostrowski. Coupled boundary-element scheme for eddy-current computation. *Journal of Engineering Mathematics*, 51, 2005.

[9] Wolfram Research, Inc. Mathematica, Version 13.0.0. Champaign, IL, 2021.

[10] Paul Leopardi and Ari Stern. The abstract hodge–dirac operator and its stable discretization. *SIAM Journal on Numerical Analysis*, 54(6):3258–3279, jan 2016.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Hodge-Laplacians and Dirac Operators on the Surface of the 3-Sphere

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Graf	Nico

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 12.07.2022	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.