Delft University of Technology

Locally *hp*-Adaptive MSEM Method

Master Thesis

Author:
Jan Roth (5262054)

November 12, 2025

Preface

This thesis was born out of desire for challenge and a wish to contribute something meaningful to the field I was interested in. I can say that with regard with former my hopes were fulfilled, while with the latter, only time can tell. I was a bit unsure about the topic initially, as I feared that it might not offer too many opportunities to do things on my own and only use tools others have used to prove they could be used for a practical topic. Thankfully I have been very wrong on that point and I think that the amount of work I had to do on my own, both independently and with guidance of others, has truly helped me grow.

I suppose that I ought to first thank my mentors prof. Marc Gerritsma and prof. Steven Hulshoff. Both were an inspiration and their lectures fascinated me during both my Bachelor and Masters studies. I think one could hardly ask for better mentors when it comes knowledge of both topics I worked on as well as support during my thesis. I am incredibly grateful that most days I could knock on their door and arrange a short meeting whenever I had issues or doubts about the theory.

At the same time, I must also thank Suyash Shrestha and Andrea Bettini, the PhD students who helped me complete this thesis by helping me with learning and understanding the theory, as well as giving me new ideas and motivations in our conversations. I wish both of them the best of luck with their doctoral studies.

To even get to this point in my studies I must express incredible gratitude to my family - brother, parents, and grandparents. Being in touch often with the marvels of modern technology made the studies not only bearable but also enjoyable. Knowing I have those I consider the closest always rooting for me and talking with them always filled me with determination.

The work on this thesis would no doubt be much harder if I was on my own. For that I think that all those who also slaved day after day in the basement of the Aerodynamics lab also deserve a word of thanks. Thanks to all fellow basement-dwellers I felt like I am not alone and truly part of a group.

As the last note, I must thank all those I met and lifted with at the university gym. Whenever the challenges of mathematics tired me out mentally, lifting heavy things together always got me in high spirits and ready for the next day. For your camaraderie and friendship I thank you from the bottom of my heart.

Jan Roth

Abstract

Adaptive meshing is important to many practical problems, though it presents many challenges. This work focuses primarily on mesh refinement within the context of Mimetic Spectral Element Method (MSEM). This is a method that approaches formulating the discrete system of equations in the style of Finite Element Method, but through application of differential geometry and exterior calculus. This results in the method satisfying the inf-sup condition by construction, allowing it to correctly solve any saddle point problem, such as the incompressible Navier-Stokes equations. This work also examines the applicability of using results given by Variational Multi-Scale (VMS) theory as an error estimator. The solver that was written based on the theory in this work and then later used to produce the results can be found at the remote repository under the GNU Public License 3.

MSEM in this work is formulated in a hybridized way, where each element is considered to have separate degrees of freedom, with continuity being enforced through Lagrange multipliers. This allows for neighboring elements to have different polynomial orders and even levels of refinement. Local refinement is achieved using raising polynomial order of elements (p-refinement) and hierarchically dividing elements (h-refinement). To this end, the theory and procedures to implement combined hp-refinement within the existing MSEM framework are formulated and verified.

From there, a refinement criterion for deciding between h-refinement or p-refinement for each element involved in a refinement sweep is introduced and validated. This refinement criterion is based on estimating the change in the L^2 error norm of the solution and is computed by using the Legendre coefficients of the error estimate for each element. This is in contrast to many other similar criteria present in literature, which consider only the solution and the current mesh state as the basis of the decision to apply either h-refinement or p-refinement.

To obtain an error estimate to use for the refinement criterion, several different estimators are proposed, notably including VMS. The main appeal of using VMS is that a good error estimate is offered by the unresolved scales, which are obtained by the method normally. These error estimates were tested on steady, two-dimensional test problems of increasing complexity, from mixed formulation Poisson equation, to linear advection-diffusion, and lastly incompressible Navier-Stokes equations. Based on these tests, VMS appears second only to knowing the exact error, though the computational cost associated with each refinement criterion was not compared.

Contents

Chapter 1

Introduction

With computational fluid dynamics, one of the most important topics for any problem is mesh generation. For any viscous flow, it is very important to properly resolve near-surface flow in order to have any idea of what the forces acting on it will be and for supersonic flows, strong gradients caused by the presence of shock waves or expansion fans must also be properly captured. At the same time, the remainder of the domain will likely not contain any important physics to resolve, so uniform mesh refinement is out of the question.

It can be difficult to a priori guess where and how much refinement should be applied to what parts of the mesh. As such, adaptive meshing, where mesh is refined based on results as they are computing, can be used to solve this issue. The topic of adaptive mesh refinement is definitely not new, with hundreds of papers and articles already tackling this problem. There exist a wide range of methods pertaining to it. For industry solvers, the most common approach is to use solution properties directly, such as with Ansys Fluent for example [1]. Another common approach is an adjoint based approach, where a goal variable, such as lift or some other force, is chosen and an adjoint problem is then solved [2, 3]. There are of course many other approaches, each with their own trade-off between cost and accuracy.

All of these methods attempt to gauge the error in the computed solution in one way or another. As such, a different approach, based on the Variational Multi-Scale theory, ought to be considered. While VMS theory has been around for quite some time, its modern resurgence began around 2007, when one of the original authors of the methods published a paper, which added much more formalism, and formulated the method in terms of projections [4], which layed a more firm theoretical foundation. The approach in that work was generalized as a by Shrestha et al. [5], where it was applied in the context of the mimetic spectral element method (MSEM). MSEM also lends itself very well to adaptive mesh refinement when formulated with a hybridized approach.

First work on locally adaptive MESM was done in 2012 by Kuystermans [6]. At the time the theoretical framework of MSEM was not as developed as it is now. That thesis also only explored h-refinement and p-refinement separately, not as a combined hp-refinement scheme. In its section on recommendations for further research it suggests that this ought to be investigated. This work was also followed up with the work of [7] written on the topic of combining the VMS theory with MSEM for advection-dominated problems.

To cover the current developments in the topics of mesh refinement, the thesis first contains a literature study in chapter 2. There the current developments in adaptive meshing, VMS, and MSEM are covered. This is aimed at giving a clear motivation for the research done in this thesis and very clearly gives rise to research questions that will be answered in this thesis.

Following that, the theory for the work in the thesis is presented. The approach taken in this thesis is heavily inspired by the two thesis mentioned thus far [6, 7], as additional theoretical background laid out by Jain et al. [8]. As such, this thesis covers the underlying theory in multiple steps. First, basic differential geometry is discussed in chapter 3, to the extent needed to understand MSEM. Following this, the formulation of MSEM is provided in chapter 4. Lastly, the theory of VMS used in this thesis is explained in chapter 5. This then concludes the theory required to understand the work done in this thesis.

With the prerequisites out of the way, chapter 6 discusses the theory and some implementation behind the hp-refinement in the context of MSEM. It also discusses the choice of h- or p-refinement, depending on the error distribution, and formulating a refinement criterion as a function of error or its estimate. It is then concluded by discussing different error estimators to be used in the numerical tests, which include VMS.

The outcomes of numerical tests are presented in chapter 7. First the criterion used is presented clearly, followed by the list of error estimators. This is followed by the overview of test problems used in the section. Following that, results for each of these test cases using each of the error estimators are presented. From there suitability of each of the error estimators can be judged. The section is concluded with a brief summary of how each of the error estimators behaves.

The thesis is then concluded by chapter 8, where the theory from chapter 6 and numerical results from chapter 7 are discussed. This discussion is formulated in the way where it can be clearly seen how the work done in the thesis answers research questions posed in the beginning. It is then followed by discussion of new research questions which arose during the thesis work, suggestions on how they could be investigated, and a reflection regarding what shortcomings were present in this thesis.

The theoretical results were tested using a solver which was written during the course of this thesis. The source code can be found in the remote repository, with the tag `thesis-results` marking the state of repository which was used to produce the following results. The code is freely available under the GPL3 license [9]. The code was originally written in Python, with performance-critical parts rewritten with C when profiling indicated significant time being spent there. At the time of writing this work, the documentation of the code is not yet finished, as it was deemed task that could be done afterwards.

While work on the solver took very significant part of time taken for this thesis, an in-depth explanation of code is not in the thesis, as the primary focus of the thesis is to show the development of MSEM, not software development. Still, there are occasional reference to implementation where it might be of some interest or where it may not be obvious that the implementation might be done in the way it was. If the code itself is of interest, the reader is invited to view it on the remote repository.

Chapter 2

Literature Study

This section presents current state of research on the three topics discussed in this thesis: adaptive meshing, variational multi-scale theory, and structure preserving methods, which mimetic spectral elements method is a member of. Each of these is discussed in more detail in each of their sections, followed by a section which raises research questions based on knowledge gaps in these sections.

2.1 Adaptive Meshing

The idea of adaptive meshing is not new by any account. A great amount of work on this topic was already done by Ivo Babuška [10, 11], who worked on the *p*- and *hp*-versions of FEM. While this work very thoroughly showed the *how* of adaptive meshing, the questions of *where* where and *how much* still remains unanswered to this day. The reason for it is quite simple - if the error distribution for the solution were to be known exactly, then the exact solution would be known either way and no mesh refinement would be needed at all!

Since for most problems of any interest the exact solution is not known, it thus becomes important to find some error estimate. To be useful, the estimate should necessarily be easier to obtain than the exact solution, as well as approximate the real error.

A very common approach is to directly use the solution given by the solver and use some quantity directly extracted from it as an indicator. A common approach for low speed aerodynamics is to try and use information about the pressure solution, such as using its Laplacian, or to use some other quantity of the flow, such as TKE [12, 13]. Another commonly taken approach taken for flow related problems is to consider different solution gradients, such as velocity, concentration, and porosity gradients [14].

While all these approaches are very appealing in terms of having a relatively very low overhead, they all share the same underlying problem. There is no real link between how large a gradient (or any other property, such as the Laplacian) of a solution is and how well it is resolved on different meshes. If a very sharp gradient is well resolved, there is no reason to keep on refining there, especially once that part becomes much more accurately resolved than the surrounding parts with lower gradients. This is a glaring issue which all of these methods based just on the solution suffer from.

These approaches were considered as ”explicit error estimators” in work by Ainsworth and Oden [15], where the division between explicit methods, which employ the obtained solution and other information about the problem, like mesh, forcing, and boundary conditions, to explicitly compute an error estimate. The same work also discusses methods for implicit error estimation, where the equation governing the error would be solved instead. A new method for implicit error estimation is also presented, based on localizing the residual problem.

In comparison to more commonly used explicit methods, adjoint-based methods offer a much more robust approach [2, 3]. For these methods, the problem is first solved for the solution of interest. From there and adjoint problem regarding some goal quantity is solved, which describes the error. This is accurate and usefully when trying to improve accuracy of a specific predicted quantity, but there is a significant downside. For unsteady problems, especially non-linear unsteady problems, there is an enormous overhead introduced. First, solving the adjoint problem involves backwards in time. If the problem is non-linear, this means that the quantities in the non-linear terms must be stored for each and all time steps in order to allow for this backwards time march. This makes this adjoint optimization very costly in terms of storage and work needed to obtain them. It is also problematic that the longer the simulation goes on, the more work will be needed.

Another method, which can be seen as the compromise between the direct and adjoint methods can be used. This is solving the problem on two different grids - a coarse and a fine one [16, 17, 18]. When both solutions are computed, the refinement can be done based on the difference between the two of them. This avoids having to solve an expensive backwards time marching adjoint problem to obtain solution information. At the same time, it does not suffer from the problems of using solution quantities alone for mesh refinement, as it is entirely clear where the solution becomes better resolved and where it was already resolved.

This method is however also not without its flaws. The most obvious one is that the only way to know that the mesh the solution was solved on is refined enough to not need any further refinement is by solving it on another finer than that. While that is true, the method is still very appealing, since it at least allows to deal with unsteady problems one time step at a time.

2.1.1 Mesh Refinement Criterion

Besides having an error estimate, which indicates where the refinement should occur, it is also important to have a refinement criterion to decide between applying h-refinement or p-refinement. A very nice overview of the state of these methods by the year 2003 was done by Houston et al. [19]. The approaches presented in that work are:

- A priori information about the problem, like singularities or discontinuities in coefficients, using h-refinement near then while p-refinement is used elsewhere.
- Type parameter, which is defined as a function of the solution, polynomial order, and mesh size. The ratio of this parameter for a solution at the same h-refinement and one p-refinement level higher is then used to decide between h-refinement or p-refinement.
- ”Texas 3-step”, where h-refinement is applied until first error tolerance is achieved, followed by p-refinement until second error tolerance is met.
- Mesh optimization for a reference known solution, where for each element, the projection problem is solved for some known reference solution. The refinement is then chosen in a way which gives the highest error reduction for the number of unknowns added. This is actually also based on work by Demkowicz et al. [16] that combined it with the error estimator itself.
- Estimating Legendre coefficient decay in the solution, where a least-squares fit of a decaying exponential is used to model values of Legendre coefficients. Based on their rate of decay, the local degree of smoothness is estimated, based on which the choice between h-refinement and p-refinement can be taken,
- Local regularity estimation by approximating local Sobolev index for each element, describing the local regularity. With this information, p-refinement is used until the order of the element is the same as the regularity of the Sobolev space, or h-refinement if the order is already high enough.

These approaches are all well motivated and can no doubt be implemented effectively in practice. However, all of the refinement criteria above, none use the error estimate itself, with most using the computed solution itself. While these can still be used in tandem with it by first locating element which must be refined by the estimators, they do not re-use the obtained information. This seems a shame in the context of using VMS, since it very naturally offers an accurate error estimator, which could be reused for a low cost.

2.2 Variational Multi-Scale Theory

VMS theory began as a way to formulate problems where all scales would not be feasible to compute [20]. These methods are applicable in many differing fields, such as material science [21, 22] or combustion [23]. In these fields, the reason for incorporating these methods is to capture effect that micro- or mesoscopic scales have on the macroscopic scales. There VMS is more commonly referred to as *homogenisation* [24]. However, that type of VMS application is not of interest for aerodynamics, as the main assumption underlying it is that different scales are separate enough to not influence one another. This is most certainly not the case for aerodynamics.

Instead, the approach that was formulated by Hughes and Sangalli [4] was to formulate the multi-scale as stemming from using a projection of the solution from the infinite-dimensional function space to a discrete polynomial function space. As such, the exact solution would very naturally be split in the part which can be represented on the finite set of basis of that space and the part which would not. Based on that, if the effect of the unresolved scales on the equation for the resolved scales could be resolved, the exact projection of the solution on the basis could be obtained.

It was recently shown by Shrestha et al. [25] that if the projector of interest is defined by the symmetric part of the differential equation operator, it is possible to entirely resolve the effect of unresolved scales on the full system exactly, knowing only the Green's function of the symmetric part, or even just using its approximation. This has been shown to work remarkably well for advection-dominated problems too.

Main drawback of using approach taken by Shrestha et al. [25] is that for the approximation of the Green's function for the symmetric operator, one needs to set up the system matrix for the problem on a finer mesh, thus sharing some hurdles with the refinement criterion based on coarse-fine mesh solution differences. However, VMS does present an advantage in that it allows for the remaining non-symmetric part of the PDE to only be solved on the coarse mesh. The system that has to be solved to obtain the unresolved scales is also quite complicated and must be solved on the fine mesh.

2.3 Mimetic Spectral Element Methods

Spectral element methods are evolution of Finite Element Exterior Calculus (FEEC). The idea behind the FEEC was to make use of differential geometry and algebraic topology to formulate consistent, stable, and convergent FEM discretizations. A large bulk of work on initially developing it came from Arnold et al. [26]. These methods were especially welcome to the field of fluid dynamics, electromagnetism, and elasticity, for providing a way to solve the saddle point problem arising in equations of all these problems. The saddle point problem in these cases involves the vector Laplacian.

The class of the methods that succeeded and built upon this framework are also know as structure-preserving methods. These methods find their uses in many other fields as well, not just in FEM [27]. The key underlying idea behind all of them is that properties which characterize the infinite-dimensional solution, such as incompressibility, conservation of energy, conservation of momentum, are also exactly conserved by the discrete solution. For example, structure preserving methods allow for discretizing incompressible Navier-Stokes equations in a way in which the obtained solution is *exactly* divergence free (incompressible). By comparison, Finite Volumes Method by definition only guarantees the incompressibility, or any other equation/constraint in the system, will only be satisfied on average for each cell.

Returning back to FEEC, as the natural evolution of previously mentioned foundation[26], came works by Kreeft et al. [28], Gerritsma [29], Palha [30] with the formulation of a mimetic framework for arbitrary order quadrilaterals. This became known as the Mimetic Spectral Element Method (MSEM) by Gerritsma [31]. These methods are have since evolved and have been successfully applied to more and more complicated problems [5, 32, 33, 34].

In its earlier days some work was also done on local p- and local h-refinement [6]. Work was done to show that local refinement is indeed possible for these methods for the both different types. However a full hp-refinement scheme was not formulated.

2.4 Knowledge Gap and Research Questions

Given the current state of research in the fields presented in section 2.1, section 2.2, and section 2.3, some knowledge gaps can be identified. From there some research questions can be formulated, which are to be answered in this thesis.

Since adaptive refinement will all be done within the context of MSEM, it first has to be investigated how that can be done. The work in this field so far looked at separate p-refinement and one level of h-refinement between the neighboring elements. Ideally, this should really be expanded to include combined hp-refinement. On top of that, it would be very useful if this refinement could be done without the need to only have a maximum of one level of h-refinement between neighboring elements.

Pertaining to the adaptive meshing and VMS, there is a question of how well does VMS perform in as a guide for adaptive mesh refinement. Since VMS recovers the effect of unresolved scales on the discrete system, it might offer a very good way to determine where the refinement should be done first. This also makes it more favourable to opt for using VMS, since it won't only improve the accuracy of the solution, but also provide a refinement criterion.

Compared to the specific method of comparing a coarse with the fine solution, VMS might also offer a more efficient approach due to the fact that it only requires the symmetrical part of the PDE to be solved on the finer mesh. As a concrete example, for the Navier-Stokes equations, that would mean only solving the Stokes flow part and only solving the non-linear advection term on the coarse mesh.

Based on that, a question of how well VMS works in case the approximation of the symmetric part is used instead of the full operator. This would greatly help with performance, since solving the full symmetric part is inherently a global operation, making it relatively expensive. If instead a locally obtainable approximation could be used, it would be even more computationally appealing, acting as a compromise between the methods which use the solution directly and those that compute it on two different meshes.

With the aforementioned research gaps, the following research questions are formulated in order to answer them in the thesis. Note that these can not cover the entirety of all knowledge gaps presented in this section, as the scope would become entirely too large.

1. How can *hp*-refinement be formulated in the MSEM context,
2. How to choose between *h*-refinement and *p*-refinement, given an esimate of the error,
3. How do different error estimators compare with VMS.

In order to find answers to these questions, about the simplest problems that should be examined would be linear advection-diffusion equations in 2D. There are quite a few good reasons as to why. First, the lowest dimension where *h*-refinement is non-trivial is the 2D case. Next, the simplest equation to solve would be linear advection-diffusion, since it is about the simplest non-symmetric PDE, with its symmetric part being the Poisson equation. Otherwise, using a PDE which is based on a fully symmetric operator, VMS would not return any additional information, since the solution would already be as good as VMS can achieve on the fine mesh, meaning it is exact in the energy norm. It is also not too dissimilar from the Navier-Stokes equation, with the two differences being that the advection term would change from linear to a non-linear and that there would be an added incompressibility constraint. Of course these two differences are sure to have very important effects, however those should be investigated only after feasibility is shown for simpler problems.

Chapter 3

Differential Geometry

First prerequisite for understanding this thesis is differential geometry. This field of mathematics encompasses many different areas and objects which are key for formulating the procedures used in this work.

An advantage of using differential geometry is that it offers a way to describe and solve differential equations in a way which is irrespective of the number of dimensions of the space they are being defined in. Another advantage is that it generalizes and unifies different differential operators from vector calculus (gradient, curl, and divergence), as well as products (multiplication of scalar and vector, cross product, dot product). While this section covers quite a number of concepts, it is by no means a complete and rigorous introduction to differential geometry. A very good introduction to this topic is given by Tu [35].

3.1 Manifolds

First concept which has to be covered is the concept of a manifold. An n-dimensional $\mathcal{M}$ manifold is defined to be an n-dimensional topological space, which locally resembles Euclidean space. It is an union of local coordinate neighbourhoods, with a mapping between each of the neighbourhoods and $\mathbb{R}^n$. Since a manifold $\mathcal{M}$ is a topological object, there is no natural distance, angle, or orientation measure. In more formal language, this means that there is no metric. To be able to define such things even locally, it is necessary to introduce tangent and co-tangent spaces.

3.2 Tangent and Cotangent Spaces

Since the manifold has no natural orientation, one must be constructed. A tangent vector space is such a concept. Consider a a point P on an n-dimensional manifold $\mathcal{M}$. For a real-valued smooth function $f : \mathbb{R}^n \to \mathbb{R}$, the derivative of f at the point P in the direction of a vector $\vec{x}$ is then defined as:

$$X_p(f) = \sum_{i=1}^{n} x_i \left.\frac{\partial f}{\partial x_i}\right|_P . \tag{3.1}$$

This can be generalized as an operator for an arbitrary function as:

$$X_p = \sum_{i=1}^{n} x_i \left.\frac{\partial}{\partial x_i}\right|_P . \tag{3.2}$$

By setting $\vec{x}$ to be the j-th basis vector, meaning that $\frac{\partial x_i}{\partial x_j} = \delta_{i,j}$, the expression from Equation 3.2 becomes:

$$X_p^j = \left.\frac{\partial}{\partial x_j}\right|_P . \tag{3.3}$$

Using these, the basis of the tangent space at the point P can thus be defined as $\left\{ \left.\frac{\partial}{\partial x_1}\right|_P, \dots, \left.\frac{\partial}{\partial x_n}\right|_P \right\}$, denoted by $\mathcal{T}_\mathcal{M}$. Using the basis of the tangent space $\mathcal{T}_\mathcal{M}$, it is possible to define a vector $\vec{v}$ in a vector space $\mathcal{V}$ as:

$$\vec{v} = \sum_{i=1}^{n} v_i \frac{\partial}{\partial x_i}. \tag{3.4}$$

The tangent space $\mathcal{T}_\mathcal{M}$ also has its own dual space. By definition, each element of the dual space maps elements of the primal space to a single value. If the dual space to the tangent space $\mathcal{T}_\mathcal{M}$ is denoted by $\mathcal{T}^\star_\mathcal{M}$, then an element from it $\alpha \in \mathcal{T}^\star_\mathcal{M}$ is defined as per Equation 3.5:

$$\alpha : \mathcal{T}_\mathcal{M} \to \mathbb{R}, \qquad \alpha \in \mathcal{T}^\star_\mathcal{M}. \tag{3.5}$$

Canonical dual basis σ_i of the dual space $\mathcal{T}^\star_\mathcal{M}$ are defined with respect to the basis e_j of the primal space $\mathcal{T}_\mathcal{M}$, such that when dual basis i is applied to the primal basis j, the result is one when $i = j$ and otherwise zero, as per Equation 3.6:

$$\sigma_i(e_j) = \delta_{i,j} \tag{3.6}$$

Based on these definitions, the dual space of the tangent space $\mathcal{T}_\mathcal{M}$ is the space $\mathcal{T}^\star_\mathcal{M}$, known as the cotangent space, spanned by the basis $\{\mathrm{d}x_{1\,P}, \dots, \mathrm{d}x_{n\,P}\}$. Its elements are known as covectors and when applied to any vector in the tangent space $\mathcal{T}_\mathcal{M}$ assign it a scalar value.

Similarly to how vectors could be written using the basis of the tangent space $\mathcal{T}_\mathcal{M}$, using the basis of the cotangent vector space $\mathcal{T}^\star_\mathcal{M}$, a linear functional $\alpha : \mathcal{V} \to \mathbb{R}$ can be defined:

$$\alpha = \sum_{i=1}^{n} \alpha_i \mathrm{d}x_i. \tag{3.7}$$

Applying α to $\vec{v}$, as expected, results in a real number:

$$\alpha(\vec{v}) = \sum_{i=0}^{n} v_i \frac{\partial}{\partial x_i} \left(\sum_{j=0}^{n} \alpha_j \mathrm{d}x_j \right) = \sum_{i=0}^{n} \sum_{j=0}^{n} v_i \alpha_j \frac{\partial x_j}{\partial x_i} = \sum_{i=0}^{n} v_i \alpha_i. \tag{3.8}$$

Having defined the tangent and cotangent spaces for a manifold $\mathcal{M}$. Using these definitions, different objects and operations can be defined.

3.3 Differential Forms

A differential form of k-th order $\alpha^{(k)}$ defined on a manifold $\mathcal{M}$, has a value expressed k-th order covector basis. This means that each k-form, maps a bundle of k vectors to a value in $\mathbb{R}$. As such, these can be seen as a quantity, which can be integrated over k-dimensional objects in order to produce a real value. This leads to some key properties, such as allowing operations like differentiation and integration, independent of the choice of the coordinate system, which will be shown later.

The simplest are the 0-forms, which assign a scalar value to each 0-dimensional object, meaning a point. As such, these are the most familiar to most people. If $f : \mathbb{R}^n \to \mathbb{R}$, then a zero form can be written as:

$$f^{(0)} = f(x_1, \dots, x_n). \tag{3.9}$$

A 0-form thus associates each point on a manifold with a value. This is illustrated in Figure 3.1.

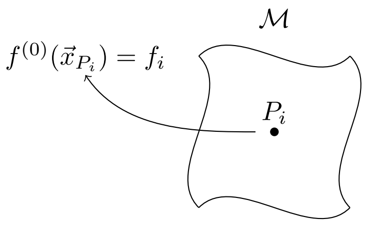


Figure 3.1: Point P_i being mapped to a value by a 0-form $f^{(0)}$.

One order higher than 0-forms are 1-forms. These by definition give a real value when integrated over a 1-dimensional object on the manifold, meaning lines. As such, they can in general be written as:

$$\alpha^{(1)} = \alpha_1(x_1, \dots, x_n)\mathrm{d}x_1 + \dots + \alpha_n(x_1, \dots, x_n)\mathrm{d}x_n. \tag{3.10}$$

As per definition, this gives a real number when integrated over a curve $\mathcal{C} \in \mathcal{M}$:

$$\int_{\mathcal{C}} \alpha^{(1)} = \int_{\mathcal{C}} \alpha_1(x_1, \ldots, x_n)\mathrm{d}x_1 + \cdots + \alpha_n(x_1, \ldots, x_n)\mathrm{d}x_n = \alpha_{\mathcal{C}} \in \mathbb{R}. \tag{3.11}$$

As such, analogous to how 0-forms map each point (0-dimensional geometrical object) to a real value, 1-forms map each curve (1-dimensional geometrical object) to a real value. An example of this is shown in Figure 3.2, where a curve C_i is mapped to a real value.

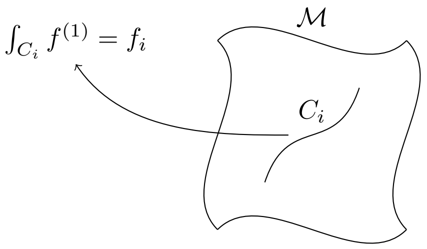


Figure 3.2: Curve C_i being mapped to a real value by a 1-form $f^{(1)}$.

This is then further extended to differential 2-forms:

$$\beta^{(2)} = \beta_{1,2}(x_1, \ldots, x_n)\mathrm{d}x_1 \wedge \mathrm{d}x_2 + \cdots + \beta_{n-1,n}(x_1, \ldots, x_n)\mathrm{d}x_{n-1} \wedge \mathrm{d}x_n. \tag{3.12}$$

These map 2-dimensional geometrical objects (surfaces) to real values, just as 1-forms and 0-forms did this for 1-dimensional and 0-dimensional objects. This is illustrated by diagram in Figure 3.3.

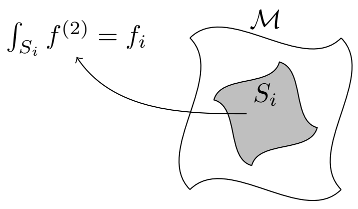


Figure 3.3: Surface S_i being mapped to a real value by a 2-form $f^{(2)}$.

This is then further generalized for higher order forms, where the underlying principle is that a k-form associates k-dimensional objects with values. As the notation can get quite cumbersome with k basis for each components, it is common to write them simply as $\gamma^{(k)} = \sum\limits_{I} \gamma_I dx^I$. To denote the set of all k-th order differential forms on a manifold $\mathcal{M}$, the symbol $\Lambda^{(k)}(\mathcal{M})$ is used. Note that Equation 3.12, wedge product $\wedge$ was used. This is an operation, which is a mapping defined as: $\wedge : \Lambda^{(k)}(\mathcal{M}) \times \Lambda^{(l)}(\mathcal{M}) \to \Lambda^{(k+l)}(\mathcal{M})$. Given $\alpha^{(k)} \in \Lambda^{(k)}(\mathcal{M})$, $\beta^{(l)} \in \Lambda^{(l)}(\mathcal{M})$, $\gamma^{(m)} \in \Lambda^{(m)}(\mathcal{M})$, and $f \in \Lambda^{(0)}(\mathcal{M})$, the following relations are satisfied:

$$\alpha^{(k)} \wedge \beta^{(l)} = (-1)^{k \cdot l} \beta^{(l)} \wedge \alpha^{(k)}, \tag{3.13}$$

$$\left(\alpha^{(k)} + \beta^{(l)}\right) \wedge \gamma^{(m)} = \alpha^{(k)} \wedge \gamma^{(m)} + \beta^{(l)} \wedge \gamma^{(m)}, \tag{3.14}$$

$$\left(\alpha^{(k)} \wedge \beta^{(l)}\right) \wedge \gamma^{(m)} = \alpha^{(k)} \wedge \left(\beta^{(l)} \wedge \gamma^{(m)}\right), \tag{3.15}$$

$$f(\alpha^{(k)} \wedge \beta^{(l)}) = f(\alpha^{(k)}) \wedge \beta^{(l)} = \alpha^{(k)} \wedge f(\beta^{(l)}). \tag{3.16}$$

3.3.1 Differential Forms in 2 and 3 Dimensions

Since 2D and 3D are of particular importance for this work, they will be discussed in a bit more detail. In particular, the aim of this subsection is to illustrate what kind of a quantities can be represented with what kind of a differential form. In traditional vector calculus, there are only scalar functions on a vector field V denoted by $U(V)$, such that $u : V \to \mathbb{R}, \forall u \in U(V)$. There are also vector-valued functions denoted by $X(V)$, where $\vec{x} : V \to \mathbb{R}^n, \forall \vec{x} \in X(V)$, and every component of $\vec{x}$ is a function on the vector field as $x_i \in U(V)$.

With differential geometry, 0-forms and n-forms act as scalar functions, as they only have a single component. While these can be augmented to represent vectors or tensors using bundle-valued forms, these will not be considered in this work, so for the purposes of what is discussed here, this is always the case. For the case of 0-forms, this is the point value with no basis, while for an n-form this is the "volume" component $dx_1 \wedge \dots \wedge dx_n$. As such, it is possible to represent scalar functions from $U(V)$ as being either 0-forms or n-forms. Which one is more appropriate will be discussed in section 3.5.

On the other hand, 1-forms (and 2-forms in 3D) have n-distinct components. With 1-forms these components have their respective basis as $\mathrm{d}x_1, \dots, \mathrm{d}x_n$. These can be thought of as a value along the direction of $\frac{\partial}{\partial x_i}$. For 2-forms in 3D, the basis are $\mathrm{d}x_2 \wedge \mathrm{d}x_3$, $\mathrm{d}x_3 \wedge \mathrm{d}x_1$, and $\mathrm{d}x_1 \wedge \mathrm{d}x_2$, which can be though of as being normal to the x_1, x_2, and x_3 directions, since that's the only direction where applying the primal basis will result in no contribution. This can clearly be seen as resulting from the duality pairing of the cotangent space $\mathcal{T}^{\star}_{\mathcal{M}}$ in which the dual basis, used by differential forms are defined, with the tangent vector space.

In summary, for scalar valued functions 0-forms and n-forms can be used for 2D and 3D, while differential forms of different orders represent vector fields. The specifics of which ones are used for what quantities will be covered in other sections.

3.4 Hodge

As mentioned in the duality pairing between inner- and outer-oriented geometrical objects on the manifold. Differential forms may be transferred between them using the *Hodge* operators, denoted by $\star$. This continuous operator is defined as a mapping between k-forms and $(n-k)$-forms $\star : \Lambda^{(k)}(\mathcal{M}) \to \Lambda^{(n-k)}(\mathcal{M})$. The mapping of basis is defined as per Equation 3.17. This not only changes the basis, but can also result in scaling the components, since covector basis are covarying.

$$\mathrm{d}x^I \to \mathrm{d}x^J, \quad \mathrm{d}x^I \wedge \mathrm{d}x^J = \mathrm{d}x_1 \wedge \dots \wedge \mathrm{d}x_n \tag{3.17}$$

Applying the Hodge operator to a k-form twice results in a k-form again, but with a potential sign change as per Equation 3.18. The exact sign of the result depends on the order of the k-form and the dimension of the manifold n.

$$\star\star\,\alpha^{(k)} = (-1)^{k(n-k)}\alpha^{(k)} \tag{3.18}$$

Overall, the application of the Hodge can be seen as rewriting a differential k-form with one orientation, as a $(n-k)$-form with the other orientation. This can also be used for a k-form from a primal space to a dual space, which will be used to define inner product in subsection 3.4.2.

Another important thing to note is the fact that the Hodge is *metric dependent*. This can be seen from the fact that basis of components are transformed per Equation 3.17. This can also be intuitively observed from the if for example an outer oriented 1-form is mapped to an inner oriented 1-form on a two dimensional manifold, the exact process of mapping must depend on how the primal and dual lines are arranged in space.

3.4.1 Orientation

A geometrical object on an n-dimensional manifold of dimension k must be given an orientation to be able to discuss integration in any meaningful way. There are two possible way to define an orientation of k-forms and their associated k-dimensional objects on a manifold: an inner and an outer orientation. An inner orientation of a manifold can be defined rigorously for any dimension, through invoking coordinate charts and transformations, with an outer orientation then having basis which are perpendicular to the basis of the inner oriented objects.

Since the focus of this work is not differential geometry, just the practical results for 2D and 3D cases are shown. In 2D, the inner-outer pairs of points, lines, and surfaces are presented in Figure 3.4. Note that there is a correspondence between the k-dimensional inner-and $(n-k)$-dimensional outer-oriented objects. This property is what the Hodge is based on, as an inner oriented k-dimensional object is mapped to a $(n-k)$-dimensional object.

3.4.2 Inner Product

With the Hodge $\star$ and wedge product $\wedge$, it is possible to now define an inner product between two differential k-forms. The definition of inner product is given per Equation 3.19. The second k-form is mapped to the dual

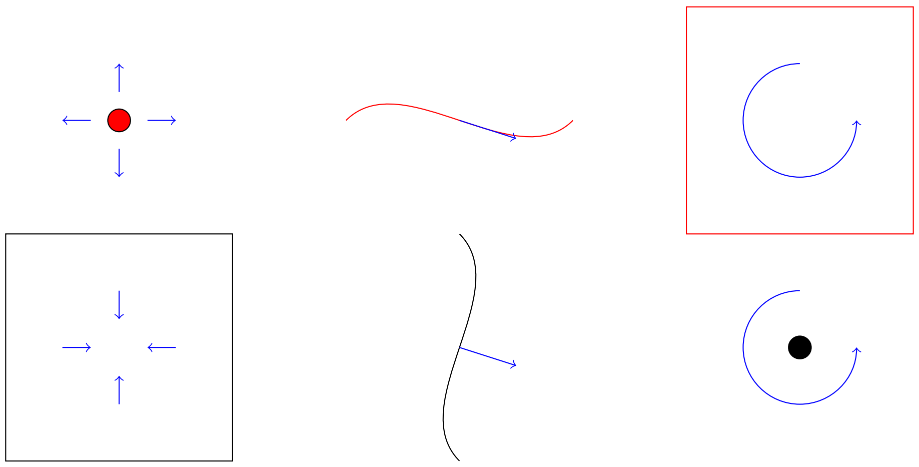


Figure 3.4: Inner orientation (in red) and outer orientation (in black) for 2D

space with the Hodge, making it a $(n-k)$-form. From there, the wedge product of the k-form and $(n-k)$-form results in a n-form, as can be seen from the definition of the Hodge's mapping of the basis in Equation 3.17.

$$\left(\alpha^{(k)}, \beta^{(k)}\right)_{\Omega^n} = \int_{\Omega^n} \alpha^{(k)} \wedge \star\beta^{(k)} \tag{3.19}$$

By attaching this inner product to the function spaces of differential k-forms $\Lambda\left(\mathcal{M}\right)^{(k)}$, it becomes a normed space. These spaces are all complete, meaning they are Hilbert spaces with the natural norm Equation 3.20.

$$\left|\left|a^{(k)}\right|\right|^2 = \left(a^{(k)}, a^{(k)}\right)_{\Omega^n} = \int_{\Omega^n} a^{(k)} \wedge \star a^{(k)} \tag{3.20}$$

3.5 Exterior Derivatives

On an n-dimensional manifold $\mathcal{M}$, the derivative of a k-form $\alpha^{(k)} \in \Lambda^{(k)}(\mathcal{M})$ is defined as per Equation 3.21 for 0-forms and Equation 3.22 for higher order differential forms. It can be seen from Equation 3.22, that the term $\mathrm{d}x^I \wedge \mathrm{d}x_i$ is zero for all I and i when $k = n$, meaning that the exterior derivative d of an n-form on an n-dimensional manifold is always zero. More importantly, it can be shown that always applying an exterior derivative to $(k+1)$-form which is a result of applying an exterior derivative to a k-form, will always map it to zero, as per Equation 3.23.

$$\mathrm{d}\alpha^{(0)} = \sum_{i=1}^{n} \frac{\partial \alpha^{(0)}}{\partial x_i} \mathrm{d}x_i \in \Lambda^{(1)}(\mathcal{M}) \tag{3.21}$$

$$\mathrm{d}\alpha^{(k)} = \sum_I \mathrm{d}\alpha_I \wedge \mathrm{d}x_i = \sum_I \left(\sum_{i=1}^{n} \frac{\partial \alpha_I}{\partial x_i} \mathrm{d}x^I \wedge \mathrm{d}x_i \right) \in \Lambda^{(k+1)}(\mathcal{M}), k \geq 1 \tag{3.22}$$

$$\mathrm{d}\left(\mathrm{d}\alpha^{(k)}\right) = 0, \qquad \alpha^{(k)} \in \Lambda^{(k)}\left(\mathcal{M}\right) \tag{3.23}$$

Using exterior derivative d, derivative operations from vector calculus in 2D or 3D can be expressed as a single operation. In 3D these are gradient of a scalar field ∇f, curl of a vector field $\nabla \times \vec{u}$, and a divergence of a vector field $\nabla \cdot \vec{w}$, while in 2D these are the curl of a scalar $\nabla \times f$, divergence of a vector field $\nabla \cdot \vec{v}$, curl of a vector field $\nabla \times \vec{v}$, and the gradient of a scalar field ∇f. These all correspond to exterior derivatives of different order differential forms on the 2-dimensional and 3-dimensional manifolds respectively.

Another important property of the exterior derivative d is how it is distributed over the wedge product $\wedge$, as shown in Equation 3.24, where $\alpha^{(k)} \in \Lambda^{(k)}(\mathcal{M})$ and $\beta^{(l)} \in \Lambda^{(l)}(\mathcal{M})$. An important thing to notice is that when $\alpha^{(k)}$ is a differential form of an even order, the rule takes a form which resembles the traditional product rule.

$$\mathrm{d}\left(\alpha^{(k)} \wedge \beta^{(l)}\right) = \mathrm{d}\alpha^{(k)} \wedge \beta^{(l)} + (-1)^k \alpha^{(k)} \wedge \mathrm{d}\beta^{(l)} \tag{3.24}$$

Lastly, it is very important to note that at no point during the definition of the exterior product d was there any need to involve the metric. As such, the exterior derivative is a *metric-free* operation and does not depend on the geometry of the manifold on which the differential forms involved are defined.

3.5.1 Generalized Stokes Theorem

Using differential geometry, the Gauss theorem for integration can be written to work for an arbitrary dimension. This is written in Equation 3.25. There is no need to involve the normal vectors or similar objects, as the basis of the differential forms already play such a role.

$$\int_{\Omega^k} \mathrm{d}\alpha^{(k-1)} = \oint_{\partial\Omega^k} \alpha^{(k-1)} \tag{3.25}$$

This is again a *metric-free* relation and a quite important one at that. Specifically, it states that if integral of a $(k-1)$ form along a boundary of a k-dimensional object is known, the integral of the derivative of that $(k-1)$-form on that k-dimensional object is *exactly equal* to the integral on the boundary. This relation is later used in section 4.2 to define exact discrete derivative.

3.5.2 Exterior Derivative in 2D

The two ways in which exterior derivative appears depend on the orientation, as discussed in subsection 3.4.1. If the resulting 1-form is written in inner-orientation, then the exterior derivative is the gradient:

$$\mathrm{d}f^{(0)} = \frac{\partial f}{\partial x_1}\mathrm{d}x_1 + \frac{\partial f}{\partial x_2}\mathrm{d}x_2. \tag{3.26}$$

Applying the exterior derivative to an inner oriented 1-form results in a curl of a vector:

$$\mathrm{d}v^{(1)} = \mathrm{d}v_1 \wedge dx_1 + \mathrm{d}v_2 \wedge \mathrm{d}x_2 = \left(\frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{x_2}\right)\mathrm{d}x_1 \wedge \mathrm{d}x_2. \tag{3.27}$$

Applying the exterior derivative to an outer oriented 1-form on the other hand results in the divergence:

$$\mathrm{d}v^{(1)} = -\mathrm{d}v_1 \wedge \mathrm{d}x_2 + \mathrm{d}v_2 \wedge \mathrm{d}x_1 = \left(\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2}\right)\mathrm{d}x_1 \wedge \mathrm{d}x_2. \tag{3.28}$$

This allows the setup of the continuous De Rham sequence which these operators form as shown in Figure 3.5. The important fact to see from here is that whilst it is true that if a k-form is an exterior derivative of a $(k-1)$-form (also know as being an "exact differential form"), its exterior derivative is zero (also know as being a "closed differential form"), it does not mean that if a k-form has a zero exterior derivative, it is a result of applying an exterior derivative to a $(k-1)$-form. In short, an all exact k-forms have zero exterior derivative, but not all k-forms with zero exterior derivative are exact.

$$\begin{tikzcd}
\Lambda^{(0)}(\mathcal{M}) \arrow[r, "\mathrm{d}"] & \Lambda^{(1)}(\mathcal{M}) \arrow[r, "\mathrm{d}"] & \Lambda^{(2)}(\mathcal{M}) \arrow[r, "\mathrm{d}"] & 0
\end{tikzcd}$$

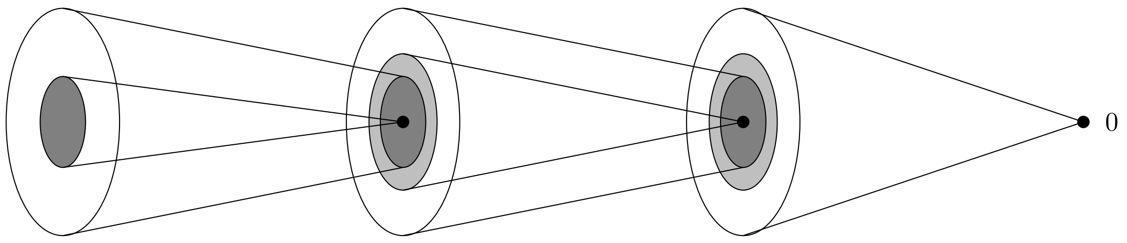


Figure 3.5: Continuous De Rham sequence for a 2D manifold.

If the Hodge operator is included in this, as the means of transferring between inner- and outer-oriented differential forms, then a double De Rham sequence can be constructed as per Figure 3.6. Here the outer-oriented differential forms are the top part of the sequence, while the bottom half is the inner-oriented differential forms. This is a choice which can be made freely. With this diagram, it is now possible to identify what differential forms to ascribe to what quantities, based on what operations are to be done to them.

For example, a vector field, for which divergence will be computed, ought to be an outer oriented 1-form, while a scalar field for which the gradient will be taken should be an inner oriented 0-form, or an outer oriented 2-form.

$$\begin{tikzcd}
\Lambda^{(0)}(\mathcal{M}) \arrow[r, "\text{d - curl}"] \arrow[d, shift left] \arrow[d, shift right, leftarrow, "\star"'] & \Lambda^{(1)}(\mathcal{M}) \arrow[r, "\text{d - div}"] \arrow[d, shift left] \arrow[d, shift right, leftarrow, "\star"'] & \Lambda^{(2)}(\mathcal{M}) \arrow[d, shift left] \arrow[d, shift right, leftarrow, "\star"'] \\
\tilde{\Lambda}^{(2)}(\mathcal{M}) & \tilde{\Lambda}^{(1)}(\mathcal{M}) \arrow[l, "\text{d - curl}"'] & \tilde{\Lambda}^{(0)}(\mathcal{M}) \arrow[l, "\text{d - grad}"']
\end{tikzcd}$$

Figure 3.6: Continuous double De Rham sequence for a 2D manifold, with outer-oriented differential forms being on top and inner-oriented being on the bottom.

3.6 Interior Product

Since an differential forms have co-vectors as their basis, they can be applied to tangent vector field. This operation is called the *interior product*. It is a linear mapping that reduces the order of the differential form by one $i_{\vec{v}} : \Lambda^{(k)}(\mathcal{M}) \to \Lambda^{(k-1)}(\mathcal{M})$. For an unit vector $\vec{v} = \frac{\partial}{\partial x_i}$, interior product with a component with a basis vector bundle $\mathrm{d}x^I = \mathrm{d}x^a \wedge \cdots \wedge \mathrm{d}x^b$ is zero if $\mathrm{d}x^i$ is not in it, otherwise the result is $\mathrm{d}x^I$ with $\mathrm{d}x^i$ removed, which is basis for a differential form one order lower. The sign of the basis is positive if the basis co-vector $\mathrm{d}x^i$ is in an odd position and negative if in an even one. Based on this definition, the following properties can be highlighted:

$$i_{\vec{v}}(f^{(0)}) = 0 \qquad \vec{v} \in \mathcal{T}(\mathcal{M}), \qquad f^{(0)} \in \Lambda^{(0)}(\mathcal{M})$$

$$i_{\vec{v}}(\alpha^{(1)}) = \alpha^{(1)}(\vec{v}) \in \mathbb{R} \qquad \vec{v} \in \mathcal{T}(\mathcal{M}), \qquad \alpha^{(1)} \in \Lambda^{(1)}(\mathcal{M})$$

$$i_{a\vec{v}+b\vec{u}}(\beta^{(k)}) = a i_{\vec{v}}(\beta^{(k)}) + b i_{\vec{u}}(\beta^{(k)}) \qquad \vec{v}, \vec{u} \in \mathcal{T}(\mathcal{M}), \qquad \beta^{(k)} \in \Lambda^{(k)}(\mathcal{M}), \qquad a, b \in \mathbb{R}$$

$$i_{\vec{v}}(a\beta^{(k)} + b\gamma^{(l)}) = a i_{\vec{v}}(\beta^{(k)}) + b i_{\vec{v}}(\gamma^{(l)}) \qquad \vec{v} \in \mathcal{T}(\mathcal{M}), \qquad \beta^{(k)} \in \Lambda^{(k)}(\mathcal{M}), \qquad \gamma^{(l)} \in \Lambda^{(l)}(\mathcal{M}), \qquad a, b \in \mathbb{R}$$

$$i_{\vec{v}}\left(\beta^{(k)} \wedge \gamma^{(l)}\right) = i_{\vec{v}}\left(\beta^{(k)}\right) \wedge \gamma^{(l)} + (-1)^k \beta^{(k)} \wedge i_{\vec{v}}\left(\gamma^{(l)}\right) \qquad \beta^{(k)} \in \Lambda^{(k)}(\mathcal{M}), \qquad \gamma^{(l)} \in \Lambda^{(l)}(\mathcal{M}) \tag{3.29}$$

By assuming primal variables use outer orientation, the interior product represents the operations shown in Figure 3.7. While in differential geometry they are all simply represented by the interior product $i_{\vec{v}}$, there are different vector calculus analogues depending on what k-form they are applied to. For the 1-forms, the outer oriented case becomes equivalent to a cross product for the vectors with only 2D components, as shown in Equation 3.30. On the other hand, it becomes a dot product for inner oriented primal forms, as per Equation 3.31. For the 2-forms, the primal case is a 1-form resulting from components of the tangent vector field $\vec{v}$ being multiplied by the 2-form, shown in Equation 3.32. For the dual case, the result is a cross-product-like operation in Equation 3.33.

$$i_{\vec{v}}(u_x\mathrm{d}y - u_y\mathrm{d}x) = u_x v_y - u_y v_x \in \Lambda^{(0)}(\mathcal{M}) \tag{3.30}$$

$$i_{\vec{v}}(u_x\mathrm{d}x + u_y\mathrm{d}y) = u_x v_x + u_x v_x \in \tilde{\Lambda}^{(0)}(\mathcal{M}) \tag{3.31}$$

$$i_{\vec{v}}(\omega\mathrm{d}x \wedge \mathrm{d}y) = v_x\omega\mathrm{d}y - v_y\omega\mathrm{d}x \in \Lambda^{(1)}(\mathcal{M}) \tag{3.32}$$

$$i_{\vec{v}}(\omega\mathrm{d}x \wedge \mathrm{d}y) = (-v_y\omega)\mathrm{d}x + (v_x\omega)\mathrm{d}y \in \tilde{\Lambda}^{(1)}(\mathcal{M}) \tag{3.33}$$

$$\begin{tikzcd}
\Lambda^{(0)}(\mathcal{M}) & \Lambda^{(1)}(\mathcal{M}) \arrow[l, "i_{\vec{v}} \text{ - cross}"'] & \Lambda^{(2)}(\mathcal{M}) \arrow[l, "i_{\vec{v}} \text{ - mul}"'] \\
\tilde{\Lambda}^{(2)}(\mathcal{M}) \arrow[r, "i_{\vec{v}} \text{ - cross}"'] & \tilde{\Lambda}^{(1)}(\mathcal{M}) \arrow[r, "i_{\vec{v}} \text{ - dot}"'] & \tilde{\Lambda}^{(0)}(\mathcal{M})
\end{tikzcd}$$

Figure 3.7: Interior product operations on a continuous double De Rham sequence for a 2D manifold.

3.6.1 Note on Dual Interior Product

Consider the case where an interior product of a k-form with an interior product of a $(k-1)$-form, which is Hodged to a dual $(n+1-k)$-form, then interior product with $\vec{v}$ applied to it, and lastly brought back as a primal k-form. Based on Equation 3.29 it is possible to rewrite it as an interior product with a primal interior product moved to the weight function:

$$\begin{aligned}
\left(u^{(k)}, \star i_{\vec{v}}\left(\star v^{(k-1)}\right)\right)_\Omega &= \int_\Omega u^{(k)} \wedge \star\star\, i_{\vec{v}}\left(\star v^{(k-1)}\right) = \int_\Omega u^{(k)} \wedge (-1)^{k(n-k)} i_{\vec{v}}\left(\star v^{(k-1)}\right) = \\
&= \cancel{(-1)^{k(n-k)} \int_\Omega i_{\vec{v}}\left(u^{(k)} \wedge \star v^{(k-1)}\right)} + (-1)^{k(n-k)+k+1} \int_\Omega i_{\vec{v}}\left(u^{(k)}\right) \wedge \star v^{(k-1)} = \\
&= (-1)^{k(n+1-k)+1}\left(i_{\vec{v}}\left(u^{(k)}\right), v^{(k-1)}\right)_\Omega .
\end{aligned}$$

This is somewhat similar to how a derivative of a dual form can be moved to the weight function, although in that case there is a boundary term which emerges due to integration by parts. This allows to work with interior products without using dual derivatives.

Chapter 4

Mimetic Spectral Elements Method

This chapter introduces the core of the Mimetic Spectral Elements Method (MSEM), since this work is based on building on existing MSEM framework.

4.1 Discretization of Differential Forms

Differential forms discussed in chapter 3 are continuous mathematical objects, residing in infinitely dimensional Hilbert spaces. Unfortunately, due to limitations beyond human control, it is only possible to numerically deal with these by restricting them to a be a linear combination of a finite number of basis functions, each with a corresponding degree of freedom. As such, a k-form $\alpha^k \in \Lambda^{(k)}(\mathcal{M})$ is represented as per Equation 4.1, where $\mathcal{N}_i^{(k)}(\alpha^{(k)})$ is the i-th degree of freedom of $\alpha^{(k)}$ and $\psi_i^{(k)}$ is the associated i-th basis k-form. Note that in this work basis are arranged in a *column* vector, rather than a row vector, which other works use [8]. It can also be seen that $\mathcal{N}_i^{(k)}$ is a functional, mapping a k-form to a single value. Given that all $\mathcal{N}_i^{(k)}$ are linear, it can be shown that there is a pairing between the basis and degrees of freedom as $\mathcal{N}_i^{(k)}\left(\psi_j^{(k)}\right) = \delta_{i,j}$. This can be recovered from Equation 4.1 by applying $\mathcal{N}_i^{(k)}$ to it sides.

$$\alpha^{(k)} = \sum_{i=1}^{N} \mathcal{N}_i^{(k)}(\alpha^{(k)})\psi_i^{(k)} = \begin{bmatrix} \psi_1^{(k)} & \cdots & \psi_N^{(k)} \end{bmatrix} \begin{bmatrix} \mathcal{N}_1^{(k)}(\alpha^{(k)}) \\ \vdots \\ \mathcal{N}_N^{(k)}(\alpha^{(k)}) \end{bmatrix} = \left(\vec{\psi}^{(k)}\right)^T \vec{\mathcal{N}}^{(k)}(\alpha^{(k)}) \tag{4.1}$$

Basis functions for different k-forms must thus be determined and chosen from correct functional spaces to allow for representation of the operations which have to be supported by these forms. It should also be noted that these basis are in themselves differential k-forms, meaning that they all contain basis co-vectors, such as the example in Equation 4.2, which shows basis for 2-forms on a 3 dimensional manifold for a case where each component of the 2-form has only one basis.

$$\psi_1^{(2)} = \psi_1 \mathrm{d}x_2 \wedge \mathrm{d}x_3 \qquad \psi_2^{(2)} = \psi_2 \mathrm{d}x_3 \wedge \mathrm{d}x_1 \qquad \psi_3^{(2)} = \psi_3 \mathrm{d}x_1 \wedge \mathrm{d}x_2 \tag{4.2}$$

These degrees of freedom $\mathcal{N}_i^{(k)}(\alpha^{(k)})$ for MSEM are based trying to discretely represent different k-forms. As such, degrees of freedom for a k-form are values of geometrical k-dimensional objects that this k-form maps them to. For a 0-form this means values at the mesh points, for 1-forms these are values of line integrals over lines of the mesh, for 2-forms they are surface integrals, and so on.

An example of how these degrees of freedom represent a function discretely, consider an example function in two dimensions presented in Figure 4.1. Its representation as discrete degrees of freedom can be seen in Figure 4.2 for all different k-form that it could be in two dimensions. When this taken to be a 0-form, it is represented by degrees of freedom as shown in Figure 4.2a. This function could also represent either the first or the second component of a 1-form, which are shown in Figure 4.2b and Figure 4.2c respectively. In those two cases, it is represented by line integrals over the lines in the mesh. Lastly, if that function was to represent a 2-form, its degrees of freedom would be represented as surface integrals shown in Figure 4.2d.

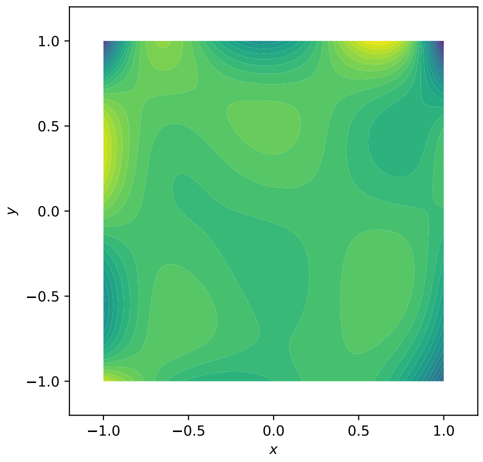


Figure 4.1: Example function in 2D

4.2 Discrete Exterior Derivative

Since there can not be a differential equation without a derivative operation, MSEM introduces the discrete version of the exterior derivative. Consider the k-form $\alpha^{(k)}$, written as a combination of a discrete number of degrees of freedom and basis functions, as per Equation 4.1. In that case, applying the exterior derivative to it results in Equation 4.3. Note that since each basis function contains co-vector basis, its exterior derivative, as shown in Equation 4.4, becomes represented by different basis of the forms of the higher order.

$$\mathrm{d}\alpha^{(k)} = \begin{bmatrix} \mathrm{d}\psi_1^{(k)} & \cdots & \mathrm{d}\psi_N^{(k)} \end{bmatrix} \vec{\mathcal{N}}^{(k)}(\alpha^{(k)}) \tag{4.3}$$

$$\mathrm{d}\psi_i^{(k)} = \sum_{j=1}^{n} \frac{\partial \psi_i}{\partial x_j} \mathrm{d}x^i \wedge \mathrm{d}x_j \tag{4.4}$$

Note that based on what co-vector basis dx^i the basis k-form contains, it will contribute to different basis of the $(k+1)$-form with its exterior derivative. As an example, the derivatives of basis for 1-forms on 3-dimensional manifold yield Equation 4.5, Equation 4.6, and Equation 4.7.

$$\mathrm{d}\psi_1^{(1)} = \mathrm{d}\left(\psi_1 \mathrm{d}x_1\right) = \frac{\partial \psi_1}{\partial x_3} \mathrm{d}x_3 \wedge \mathrm{d}x_1 - \frac{\partial \psi_1}{\partial x_2} \mathrm{d}x_1 \wedge \mathrm{d}x_2 \tag{4.5}$$

$$\mathrm{d}\psi_2^{(1)} = \mathrm{d}\left(\psi_2 \mathrm{d}x_2\right) = \frac{\partial \psi_2}{\partial x_1} \mathrm{d}x_1 \wedge \mathrm{d}x_2 - \frac{\partial \psi_2}{\partial x_3} \mathrm{d}x_2 \wedge \mathrm{d}x_3 \tag{4.6}$$

$$\mathrm{d}\psi_3^{(1)} = \mathrm{d}\left(\psi_3 \mathrm{d}x_3\right) = \frac{\partial \psi_3}{\partial x_2} \mathrm{d}x_2 \wedge \mathrm{d}x_3 - \frac{\partial \psi_3}{\partial x_1} \mathrm{d}x_3 \wedge \mathrm{d}x_1 \tag{4.7}$$

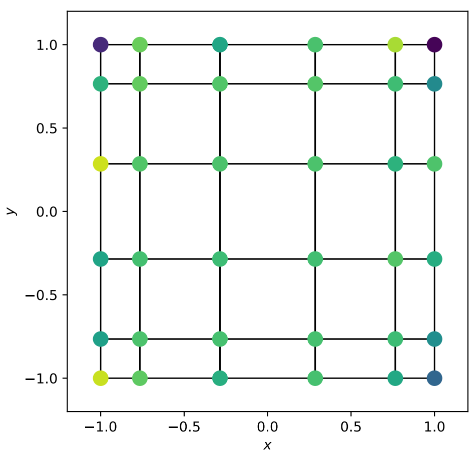
MSEM exploits this relation between the basis of derivatives and derivatives of basis, which allows this operation to be performed exactly, irrespective of the mesh geometry. This is done by first constructing basis of the 0-form in arbitrary dimension as a product of 1D nodal basis $h(\xi)$. For MSEM a very common choice is Lagrange polynomials associated with Gauss-Lobatto-Legendre (GLL) nodes. This thus fixes degrees of freedom of 0-form to be nodal values at these GLL nodes. While this choice common (see for example [8]), there is no reason why a different choice could not be made.

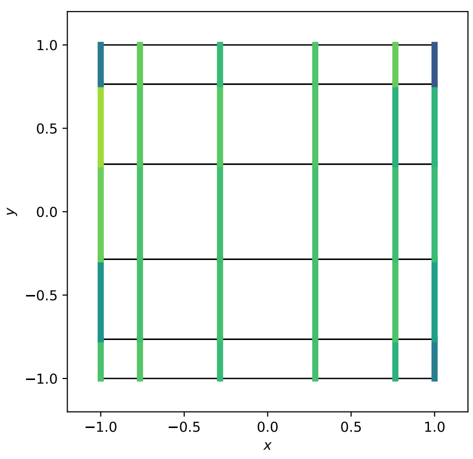
Consider a set of p-th order basis $H = \{h_1(\xi), \ldots, h_{p+1}(\xi)\}$. The nodal basis in n-dimensions would be written as Equation 4.8. In 2D, these basis would then be given by Equation 4.9.

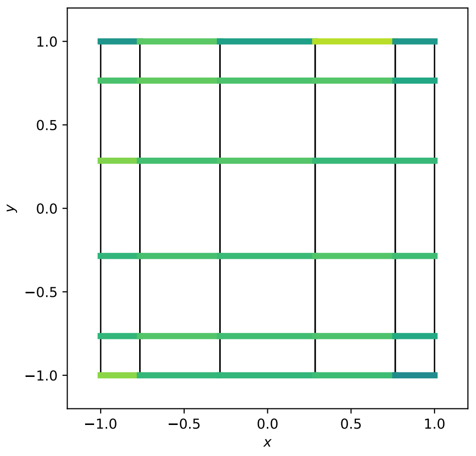
$$\psi_{i_1,\ldots,i_n}^{(0)}(x_1, \ldots, x_n) = \prod_{k=1}^{n} h_{i_k}(x_k), \quad i_1, \ldots, i_n \in [1, p+1] \tag{4.8}$$

$$\psi_{i,j}^{(0)}(x_1, x_2) = h_i(x_1) \cdot h_j(x_2), \quad i, j \in [1, p+1] \tag{4.9}$$

Since the exterior derivative is an entirely topological operation, being independent of metric (no Hodge) at a continuous level, MSEM mimics this property at the discrete level as well. With the choice of 0-form basis being


(a) Example function DoFs for a 0-form


(b) Example function DoFs for the first component of a 1-form


(c) Example function DoFs for the second component of a 1-form

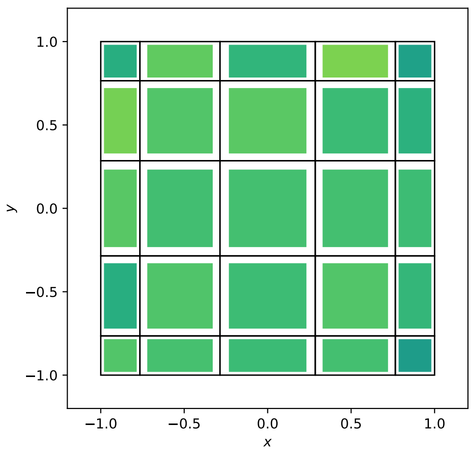

(d) Example function DoFs for a 2-form

Figure 4.2: Degrees of freedom for the example function in Figure 4.1 if taken as any of the different k-forms in 2D.

associated with values at mesh nodes, the generalized Stokes theorem, which was discussed in subsection 3.5.1 can be used.

For 0-forms this theorem is formulated as Equation 4.10 and illustrated in Figure 4.3. This means that an integral of the an external derivative of a 0-form $u^{(0)}$ is equal to the difference of its values at the end and beginning points of the curve C_i.

$$\int_{C_i} \mathrm{d}u^{(0)} = u^{(0)}\left(p^i_{\text{end}}\right) - u^{(0)}\left(p^i_{\text{begin}}\right) \tag{4.10}$$

The degrees of freedom for 1-forms in MSEM are line integrals of tangential or normal components of 1-forms along mesh lines, so using Equation 4.10, it is possible to express these degrees of freedom for a derivative of a 0-form in terms of degrees of freedom of the 0-form and orientation only. This is relation is given as Equation 4.11.

$$\mathcal{N}^{(1)}_i\left(\mathrm{d}u^{(0)}\right) = \int_{C_i} \mathrm{d}u^{(0)} = u^{(0)}\left(p^i_{\text{end}}\right) - u^{(0)}\left(p^i_{\text{begin}}\right) = \mathcal{N}^{(0)}_{\text{end}}\left(u^{(0)}\right) - \mathcal{N}^{(0)}_{\text{begin}}\left(u^{(0)}\right) \tag{4.11}$$

When this is done for each 1-form degree of freedom and organized in a matrix which can be applied to a vector of degrees of freedom of the 0-form $\vec{\mathcal{N}}^{(0)}\left(u^{(0)}\right)$, an incidence matrix $\mathbb{E}^{(1,0)}$ is created. This is a very sparse

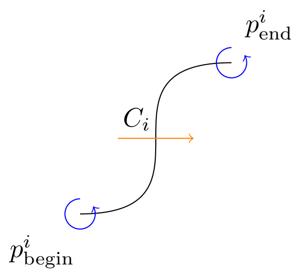


Figure 4.3: Application of Stokes theorem to link an exterior derivative of a 0-form with the form it comes from.

matrix, since each row only contains two non-zero entries, the sign of which depends on the orientation of the mesh points and lines. This matrix depends only on the orientation of element degrees of freedom and the order of the element, being completely independent of the metric.

Similarly to how the Stokes theorem is used to link a 0-form with its exterior derivative, an exterior derivative of a 1-form can be linked with the 1-form it comes from. For a quadrilateral surface, this is done as illustrated in Figure 4.4 and derived in Equation 4.12. What should be noted is the orientation of the lines and surface in these are assumed to match, meaning that the integrals in Equation 4.12 may be negative if the lines are oriented to consider positive normal integral to be in the different direction.

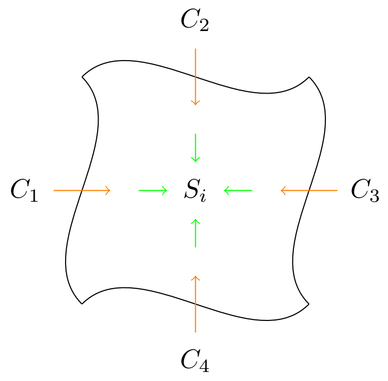


Figure 4.4: Application of Stokes theorem to link an exterior derivative of a 1-form with the form it comes from.

$$\int_{S_i} \mathrm{d}v^{(1)} = \int_{\partial S_i} v^{(1)} = \int_{C_1} v^{(1)} + \int_{C_2} v^{(1)} + \int_{C_3} v^{(1)} + \int_{C_4} v^{(1)} \tag{4.12}$$

Just as for 0-forms there is an incidence matrix $\mathbb{E}^{(1,0)}$, which maps degrees of freedom of a 0-form to those of its 1-form exterior derivative, the incidence matrix $\mathbb{E}^{(2,1)}$ can be assembled in the same fashion. For quadrilateral surfaces, each row has four entries, with the signs of the entries once again depending only on the mesh orientation, not on the metric.

What is interesting is that these matrices are also related to weak derivatives. Applying integration by parts to obtain a weak givens the term in Equation 4.13. Since the weak derivative term on the left side of Equation 4.13 is the result of integration by parts, it is actually the negative weak gradient. As such, the weak gradient for a k-form (which is a $(k-1)$-form) is applied by the negative transpose of the matrix $\mathbb{E}^{(k,k-1)}$. This operation is performed on dual forms, so the k-form in question must first be Hodged to a dual form. As such, the weak derivative is not a metric-free operation. Note that the boundary term that should also be present for a weak derivative is omitted from Equation 4.13.

$$\left(\mathrm{d}v^{(k)}, u^{(k+1)}\right)_\Omega = \left(\vec{\mathcal{N}}^{(k)}\left(v^{(k)}\right)\right)^T \left(\mathbb{E}^{(k+1,k)}\right)^T \mathbb{M}^{(k+1)} \vec{\mathcal{N}}^{(k+1)}\left(u^{(k+1)}\right) \tag{4.13}$$

A key property which is also satisfied by the exterior derivative operation is that applying a derivative twice to a k-form maps it to zero. This therefore means that applying an incidence matrix $\mathbb{E}^{(k+2,k+1)}$ to the incidence matrix $\mathbb{E}^{(k+1,k)}$ must necessarily result in a matrix of all zeros, since that is the only way that Equation 4.14 can be satisfied for any k-form $\alpha^{(k)}$.

$$\mathrm{d}\mathrm{d}\alpha^{(k)} = \left(\vec{\psi}^{(k+2)}\right)^T \mathbb{E}^{(k+2,k+1)}\mathbb{E}^{(k+1,k)}\vec{\mathcal{N}}^{(k)}\left(\alpha^{(k)}\right) = 0 \tag{4.14}$$

4.3 Discrete Hodge

While exterior derivative operation can be exact in the discrete setting, the same does not hold for the Hodge. The Hodge transfers an inner-oriented k-form into an outer-oriented one, and vice versa. For differential k-forms of a specific orientation, it is required to have the same number of degrees of freedom for the differential $(n-k)$-forms of the opposite orientation to have have an inevitable mapping. In the case where degrees of freedom of k-forms are associated with the k-dimensional geometrical objects, it implies that given a specific orientation is chosen for a so called *primal* mesh, it is required that the quantities of opposite orientation reside on a *dual* mesh, where each k-dimensional geometrical object on the primal mesh corresponds to a $(n-k)$-dimensional object on the dual mesh to have a bijective mapping between the two.

There are two main ways to approach this problem of transferring between quantities on the primal mesh defined in first orientation and the dual quantities defined on the dual mesh with the opposite orientation:

1. Construct the dual mesh geometry.
2. Construct the mapping between the two.

If the first approach is followed, the mapping depends on the choice of dual mesh geometry, assuming the primal mesh is the same between the two approaches. The mapping then simply becomes interpolation of degrees of freedom from one mesh to another, which is written as applying an interpolation matrix to the primal degrees of freedom.

On the other hand, it is possible to work the other way around and start by first defining the mapping. The dual mesh degrees of freedom might then lose the meaning they would otherwise have with the first approach, which means only quantities on the primal mesh have any sensible interpretation.

While MSEM allows for both, here the second approach is followed. By setting the mapping from the primal mesh to the dual mesh for k-forms to be defined as simply the basis mass matrix $\mathbb{M}^{(k)}$ as defined in Equation 4.15, it allows for re-use of matrices which arise due to the FEM formulation on its own.

$$\left(\mathbb{M}^{(k)}\right)_{i,j} = (\psi_i^{(k)}, \psi_j^{(k)})_{\Omega^n} = \int_{\Omega^n} \psi_i^{(k)} \wedge \star\psi_j^{(k)} \tag{4.15}$$

4.4 Discrete Interior Product

To evaluate interior products in the numerical setting, degrees of freedom of resulting form need to be computed. This is done by computing the exact interior product and projecting it on the basis of the resulting form. Due to being more expedient, this is done by computing the degrees of freedom of the Hodge resulting form, meaning that dual degrees of freedom are computed. The reason for this is that it is cheap to evaluate the interior product point wise, so when faced with a choice to either compute degrees of freedom of the form or its Hodge, it takes many less evaluations to compute the dual degrees of freedom instead, especially for 1-forms.

High cost of evaluation for 1-form degrees of freedom is due to the fact that in an element of order N a total of $2N(N+1)$ degrees of freedom must be evaluated. This means evaluating that $2N(N+1)$ numerical line integrals to compute the non-Hodged degrees of freedom. When an integration rule with M points is used, the number of times both the tangent vector field $\vec{v}$ and the 1-form in question must be evaluated is the product of the two.

When instead dual degrees of freedom are evaluated instead, what needs to be computed is the integral given in Equation 4.16. This requires evaluation of the interior product only at the 2D integration rule points, which can be reused between the different degrees of freedom. The only things changing between the integrals are the basis functions. These Hodged degrees of freedom can then get mapped to primal degrees of freedom by applying the inverse mass matrix.

$$\left(\psi_i^{(k-1)}, i_{\vec{v}}\left(\alpha^{(k)}\right)\right)_\Omega = \int\limits_\Omega \psi_i^{(k-1)} \wedge \star i_{\vec{v}}\left(\alpha^{(k)}\right) \tag{4.16}$$

When the tangent vector field $\vec{v}$ is specified in advance, the mapping from degrees of freedom of the original k-form into the degrees of the $(k-1)$-form that is the result of the interior product can be expressed as a matrix. This matrix is of course not square, since k-forms and $(k-1)$-forms do not have the same number of degrees of freedom. This matrix is again computed by using Equation 4.16, then applying the Hodge to return them back to primal degrees of freedom.

For dual k-forms, interior product can be computed by re-writing them in terms of primal basis functions, then applying the matrix that computes the dual degrees of freedom of the inner product from Equation 4.16. That way, dual basis functions need never be evaluated.

There is another option for computing the interior products dual k-forms, based on the fact that as was shown in subsection 3.6.1, these can be rewritten to have the interior product applied on the weight form. When the interior product operator for primal forms are written as matrices multiplying degree of freedom vectors, it then becomes very clear that their transposes become matrices used for interior products of dual forms. Either way, both methods of deriving these operators will yield the same result.

For non-linear problems, such as the Navier-Stokes equations, the tangent vector field involved in the interior product is the result of "lowering" a 1-form (co-vector basis) into a tangent vector field (contra-vector basis). In such cases, this mapping is non-linear. If fixed-point iteration is used, to solve the problem, then a single matrix can be used to apply the interior product to the degrees of freedom and assemble the linearized system.

On the other hand, if Newton's method is used to solve a linear system with a bi-linear term, then two matrices will result from the interior product term, with the newly added matrix appearing from partial derivatives of the interior product term with respect to the degrees of freedom from which the tangent vector field is being computed. This can be seen that given degrees of freedom for differential forms $\vec{x}$ and $\vec{y}$, the bi-linear form can be written as Equation 4.17, with $\mathbb{M}_{\vec{x}}$ and $\mathbb{M}_{\vec{y}}$ representing linearized system around the input state of $\vec{x}$ and $\vec{y}$ respectively.

$$b(\vec{x}, \vec{y}) = \mathbb{M}_{\vec{x}}\vec{y} = \mathbb{M}_{\vec{y}}\vec{x} \tag{4.17}$$

With the solver that was written along with this thesis, using the Newton's method was found to be very cumbersome when compared to fixed-point iteration. Profiling revealed the cause to be the need to recompute the non-linear blocks. While it is true that fewer iterations were needed, fixed-point iteration ended up being faster simply due to low iteration cost. With a lower cost and larger region of stability, non-linear problems ended up being solved using fixed-point iteration.

To put all this together, to compute primal degrees of freedom of an interior product of a primal form, Equation 4.16 can be used, then the resulting dual degrees of freedom are Hodged back to primal. To compute the dual degrees of freedom of interior product of dual forms, they need to first be Hodged to primal forms, at which point Equation 4.16 can be used.

These mappings are shown on double de Rham sequence for primal interior products in Figure 4.5, where $\mathbb{M}_{\vec{v}}^{(1,2)}$ and $\mathbb{M}_{\vec{v}}^{(0,1)}$ represent matrices which apply Equation 4.16 to a primal 2-from and 1-form respectively. The top half shows the primal form spaces and their interior products, while the bottom shows the dual forms and their interior products. The matrices that perform these operations can be changed in case where the tangent vector field $\vec{v}$ is the result of lowering of a 1-form, meaning that $\vec{v} = \left(v^{(1)}\right)^\flat$. The mappings have the same meaning as their continuous counterparts:

- Primal 1-form to 0-form: cross product of vectors,
- Primal 2-form to 1-form: scalar multiplied by vector,
- Dual 1-form to a 2-form: dot product of vectors,
- Dual 0-form to a 1-form: cross multiplication by vector.

$$\begin{tikzcd}
H(\Omega,\mathrm{curl}) & & H(\Omega,\mathrm{div}) \arrow[ll, bend right, "\mathbb{I}_{\vec{v}}^{(0,1)} = \left(\mathbb{M}^{(0)}\right)^{-1}\mathbb{M}_{\vec{v}}^{(0,1)}"'] & & L^2(\Omega) \arrow[ll, bend right, "\mathbb{I}_{\vec{v}}^{(1,2)} = \left(\mathbb{M}^{(1)}\right)^{-1}\mathbb{M}_{\vec{v}}^{(1,2)}"'] \\
\widetilde{L}^2(\Omega) \arrow[rr, bend right, "\mathbb{I}_{\vec{v}}^{(\widetilde{1},\widetilde{2})} = -\left(\mathbb{M}_{\vec{v}}^{(0,1)}\right)^T\left(\mathbb{M}^{(0)}\right)^{-1}"'] & & \widetilde{H}(\Omega,\mathrm{curl}) \arrow[rr, bend right, "\mathbb{I}_{\vec{v}}^{(\widetilde{1},\widetilde{0})} = -\left(\mathbb{M}_{\vec{v}}^{(1,2)}\right)^T\left(\mathbb{M}^{(1)}\right)^{-1}"'] & & \widetilde{H}(\Omega,\mathrm{grad})
\end{tikzcd}$$

Figure 4.5: Discrete De Rham sequence for a 2D manifold with interior product operators.

4.5 Discrete De Rham Sequence

Using these discrete k-forms and operators, it is possible to now construct a discrete De Rham sequence and operators which transfer between them as per Figure 4.6, which shows the inner-oriented differential forms on the bottom, while the outer-oriented are on top. The function spaces for all differential forms have also been restricted to Sobolev space, which are more typical for FEM formulations. Another interesting property that can be observed is that all operations on the bottom part of the diagram, which correspond to operations on the dual forms, can be written as transpose of the primal operators, which in the FEM context means a primal operation is applied on the weight. For the derivatives this means that a weak derivative is used, while for interior product both primal and dual are equally "strong" and neither involves boundary terms.

$$\begin{tikzcd}
H(\Omega,\mathrm{curl}) \arrow[rr, bend left, "\mathbb{E}^{(1,0)}"] \arrow[d, bend left, "\mathbb{M}^{(0)}"] & & H(\Omega,\mathrm{div}) \arrow[ll, bend left, "\mathbb{I}_{\vec{v}}^{(0,1)}"] \arrow[rr, bend left, "\mathbb{E}^{(2,1)}"] \arrow[d, bend left, "\mathbb{M}^{(1)}"] & & L^2(\Omega) \arrow[ll, bend left, "\mathbb{I}_{\vec{v}}^{(1,2)}"] \arrow[d, bend left, "\mathbb{M}^{(2)}"] \\
\widetilde{L}^2(\Omega) \arrow[u, bend left, "\left(\mathbb{M}^{(0)}\right)^{-1}"] \arrow[rr, bend right, "-\left(\mathbb{I}_{\vec{v}}^{(0,1)}\right)^T"'] & & \widetilde{H}(\Omega,\mathrm{curl}) \arrow[u, bend left, "-\left(\mathbb{M}^{(1)}\right)^{-1}"] \arrow[ll, bend right, "-\left(\mathbb{E}^{(1,0)}\right)^T"'] \arrow[rr, bend right, "-\left(\mathbb{I}_{\vec{v}}^{(1,2)}\right)^T"'] & & \widetilde{H}(\Omega,\mathrm{grad}) \arrow[u, bend left, "\left(\mathbb{M}^{(2)}\right)^{-1}"] \arrow[ll, bend right, "-\left(\mathbb{E}^{(2,1)}\right)^T"']
\end{tikzcd}$$

Figure 4.6: Discrete double De Rham sequence for a 2D manifold, with outer-oriented differential forms being on top and inner-oriented being on the bottom.

For the practical implementation of the solver written, the Hodge was not explicitly an operator. Since one can perform all the operations in the dual space of Figure 4.6 by taking an inner product with a primal weight function, then apply primal operations on the weight form, that is how it is handled. There is also no real worry about redundant operations, such as applying a mass matrix just to apply its inverse right after. For this exact reason, the solver breaks interior product into the dual-to-dual and inverse mass matrix step and then try to simplify.

4.6 Reference Element

At the core of MSEM is the single reference element. All equations are formulated on the physical domain, then transferred to a reference element. From there, the solution on the mesh is then obtained by introducing Lagrange multipliers to ensure that elements which are adjacent to each other agree on the shared degrees of freedom. The reference element as shown in Figure 4.8 and the degrees of freedom which are shown on it are always defined in the *reference domain*, such that the element is a square $(\xi, \eta) \in [-1, +1] \times [-1, +1]$. This reference element is then mapped to a physical element in the domain with the use of a mapping $\phi : (\xi, \eta) \rightarrow (x, y)$, which must be invertable, meaning that there exists a reverse mapping $\phi^{-1} : (x, y) \rightarrow (\xi, \eta)$. This is illustrated in Figure 4.7.

The reason for having this setup is because in order to solve equations on a physical element e_i different operations must be computed. These become very complicated if not impossible on a deformed domain. To this end, physical quantities are transferred to the reference domain using the mapping ϕ, where these calculations can be very easily performed. From there, the results can once again be transfered back to the physical domain with the inverse mapping ϕ^{-1}.

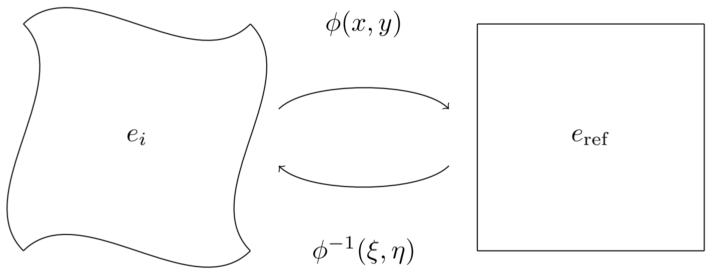


Figure 4.7: Mapping between the reference domain and the physical domain visualized

Consider a reference 2D element as shown in Figure 4.8, where degrees of freedom for nodes (corresponding to 0-forms), edges (corresponding to 1-forms), and surfaces (corresponding to 2-forms) are visualized. Note that the outer orientation is used for these. As mentioned before, there are also dual degrees of freedom (with inner orientation), associated with the dual nodes, edges, and surfaces, however, due to the way that the Hodge is defined in section 4.3, these aren't possible to visualize, since they don't correspond to any "real" mesh.
With the element, basis and their degrees of freedom, and the discrete De Rham sequence defined, it is now possible to discretize the differential equations that are to be solved. Consider for example the very basic Laplace equation in Equation 4.18. With differential geometry in 2D, this can be written in two different ways: either as a curl of a scalar followed by a curl of a vector field, or as a gradient of a scalar field, followed by the divergence of the resulting vector field. These give rise to the two formulations of the scalar Laplacian based on 0-forms and 2-forms respectively, as written in Equation 4.19 and Equation 4.20. These equations can then be solved using MSEM.

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{4.18}$$

$$\star d \star d\phi^{(0)} = 0 \tag{4.19}$$

$$d \star d \star \phi^{(2)} = 0 \tag{4.20}$$

When Equation 4.19 is written in variational formulation, Equation 4.21 is obtained. Using the double Hodge property from Equation 3.18 and generalized Stokes theorem Equation 3.25, it is then possible to simplify it further into Equation 4.22 and Equation 4.23.

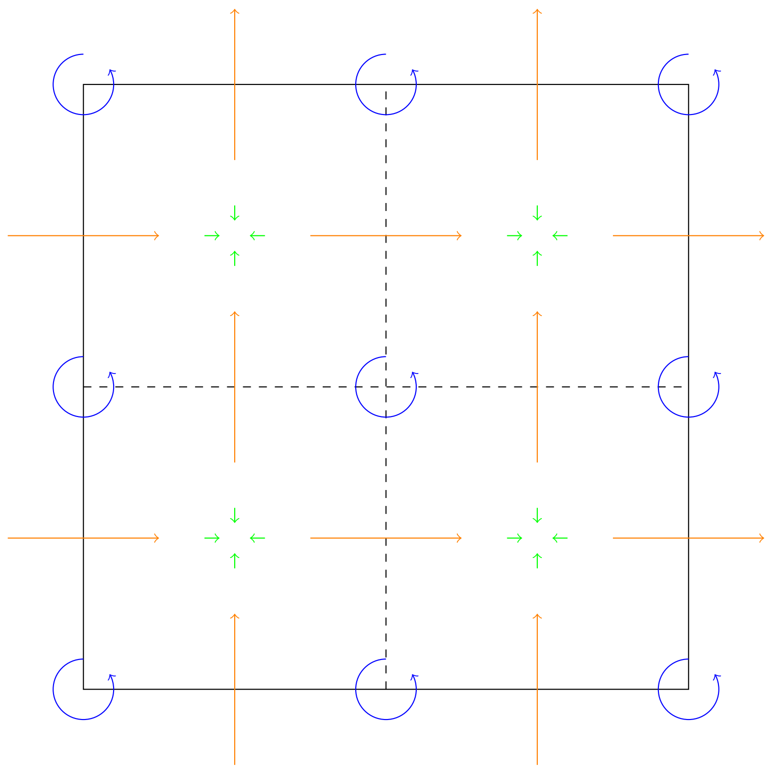


Figure 4.8: Degrees of freedom for nodes (blue), edges (orange), and surfaces (green) on a reference element of the polynomial order $p = 2$.

$$\left(v^{(0)}, \star d \star d\phi^{(0)}\right)_{\Omega} = \int_{\Omega} v^{(0)} \wedge \star\star d \star d\phi^{(0)} = 0 \qquad \forall v^{(0)} \in \Lambda^{(k)}\left(\Omega\right) \tag{4.21}$$

$$\left(v^{(0)}, \star d \star d\phi^{(0)}\right)_{\Omega} = \int_{\Omega} v^{0} \wedge d \star d\phi^{(0)} = \int_{\Omega} d\left(v^{0} \wedge \star d\phi^{(0)}\right) - \int_{\Omega} dv^{(0)} \wedge \star d\phi^{(0)} \qquad \forall v^{(0)} \in \Lambda^{(k)}\left(\Omega\right) \tag{4.22}$$

$$\left(v^{(0)}, \star d \star d\phi^{(0)}\right)_{\Omega} = \int_{\Omega} v^{0} \wedge d \star d\phi^{(0)} = \int_{\partial\Omega} v^{(0)} \wedge \star d\phi - \left(dv^{(0)}, d\phi^{(0)}\right)_{\Omega} \qquad \forall v^{(0)} \in \Lambda^{(k)}\left(\Omega\right) \tag{4.23}$$

The Equation 4.23 ultimately leads to the final formulation for the so called "direct" Laplacian, as written in Equation 4.24. There $g^{(1)}$ is used as a shorthand for the derivative of $\phi^{(0)}$. This can be discretized using MSEM to obtain the form in Equation 4.25. By requiring it holds for any admissible $v^{(0)}$, it becomes a very straight-forward system, as per Equation 4.26

$$\left(dv^{(0)}, d\phi^{(0)}\right)_{\Omega} = \int_{\partial\Omega} v^{(0)} \wedge \star d\phi^{(0)} = \int_{\partial\Omega} v^{(0)} \wedge \star g^{(1)} \qquad \forall v^{(0)} \in \Lambda^{(0)}\left(\Omega\right) \tag{4.24}$$

$$\left(\mathbb{E}^{(1,0)}\vec{v}\right)^{T} \mathbb{M}^{(1)} \left(\mathbb{E}^{(1,0)}\vec{\phi}\right) = \vec{v}^{T}\vec{g} \tag{4.25}$$

$$\left(\left(\mathbb{E}^{(1,0)}\right)^{T} \mathbb{M}^{(1)}\mathbb{E}^{(1,0)}\right)\vec{\phi} = \vec{g} \tag{4.26}$$

The formulation for the 2-form Laplacian is similar, however, for the reasons discussed in section 4.7, it is necessary to write the problem in the so-called "mixed" formulation. In this case, it means introducing the derivative of the solution $u^{(1)}$. Note that this is a "weak" derivative, since in two dimensional space, the "strong" exterior derivative of a 2-form is zero. As such, the weak derivative is taken, which would in continuous setting correspond to taking the Hodge of the 2-form, taking its derivative, then applying the Hodge again to make it a primal form again. This means that $u^{(1)} = \star d \star \phi^{(2)}$. When written in the variational form it results in Equation 4.27.

$$0 = \left(\omega^{(1)}, u^{(1)} - \star d \star \phi^{(2)}\right)_{\Omega} = \left(\omega^{(1)}, u^{(1)}\right)_{\Omega} + \left(d\omega^{(1)}, \phi^{(2)}\right)_{\Omega} - \int_{\partial\Omega} \omega^{(1)} \wedge \star\phi^{(2)} \qquad \forall \omega^{(1)} \in \Lambda^{(1)}\left(\Omega\right) \tag{4.27}$$

Using this, it is possible to then write the 2-form Laplacian as Equation 4.28 and Equation 4.29. When discretized in the way Equation 4.24 was, Equation 4.30 is obtained.

$$\begin{aligned}\left(p^{(1)}, q^{(1)}\right)_\Omega + \left(\mathrm{d}p^{(1)}, u^{(2)}\right)_\Omega &= \int_{\partial\Omega} p^{(1)} \wedge \star u^{(2)} && \forall p^{(1)} \in \Lambda^{(1)}(\Omega) && (4.28)\\ \left(v^{(2)}, \mathrm{d}q^{(1)}\right)_\Omega &= 0 && \forall v^{(2)} \in \Lambda^{(2)}(\Omega) && (4.29)\end{aligned}$$

$$\begin{bmatrix} \mathbb{M}^{(1)} & \left(\mathbb{M}^{(2)}\mathbb{E}^{(2,1)}\right)^T \\ \mathbb{M}^{(2)}\mathbb{E}^{(2,1)} & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \vec{\phi} \end{bmatrix} = \begin{bmatrix} \vec{g} \\ 0 \end{bmatrix} \tag{4.30}$$

Using the discretizations in Equation 4.26 and Equation 4.30, there is now a formulation of the PDE in the discrete setting on the reference element. All that is required now, is to connect multiple elements to gether, which is discussed in section 4.7.

4.7 Connectivity

With the system formulated on the reference element in section 4.6, it is now necessary to connect multiple elements together. This section first considers only the case where all neighboring elements have the exact same polynomial order and no subdivisions, as those are discussed later in chapter 6. The basic principle behind connectivity in that case is to impose constraints between elements e_1 and e_2 which share a boundary, such that the two agree on the solution on that boundary. For the 2D case, this inadvertently means that 2-forms can not have any form of continuity imposed upon them. This is also the reason why the 2-form Laplacian in section 4.6 can not be solved with MSEM without the mixed formulation, as there would be nothing connecting element solutions with each other if not for a non-2-form variable.

There are two main options for ensuring continuity of the elements. First one is that a global list of degrees of freedom is maintained and that when local element matrices are assembled, these are transferred to the global matrix via local-to-global index lookup. While there's an advantage to this method in that it introduces no additional degrees of freedom, it has its drawbacks. First, it requires a considerable amount of book-keeping, which can be especially tedious in the context of distributed computing. It also makes the bandwidth of the resulting global matrix very large, depending on the mesh topology.

The second option is to use Lagrange multipliers. These introduce new degrees of freedom and add new equations in the form of $\lambda\left(u_{e_1} - u_{e_2}\right)$, as well as extra terms to the equations involving the unknowns they constrain. On the other hand, it significantly reduces the bandwidth of the global matrix, as it then always retains the form shown in Equation 4.31. It thus allows for a much nicer sparsity pattern, as well as having an option to solve the system using the Schur's compliment, assuming that element matrices are inevitable. It should be noted that with MSEM, it is always possible to write the system in such a way by changing the formulation to its dual.

Ultimately, between the two options, the second method of using Lagrange multipliers was chosen. This was not only done for the sake of making distributed computations easier, but what is even more important, is the fact that in this thesis elements with mismatching number of degrees of freedom will be used. This would mean that the book-keeping arising from the first approach would become even more cumbersome, and the entries of element matrices pertaining to the boundary degrees of freedom would need to be transformed.

$$\mathbf{A}_g = \begin{bmatrix} \begin{bmatrix} \mathbf{A}_1 & & & & \\ & \mathbf{A}_2 & & & \\ & & \mathbf{A}_3 & & \\ & & & \mathbf{A}_4 & \\ & & & & \mathbf{A}_5 \end{bmatrix} & \begin{bmatrix} \\ \mathbf{N}^T \\ \\ \end{bmatrix} \\ \begin{bmatrix} & & \mathbf{N} & & \end{bmatrix} & 0 \end{bmatrix} \tag{4.31}$$

In two dimensional case the Lagrange multipliers are thus used to enforce that points of neighboring elements have matching values for 0-form degrees of freedom, as shown in Figure 4.9. Similarly, for 1-forms, the multipliers enforce that degrees of freedom representing the line integrals over the same lines match, just as shown in Figure 4.10.
As mentioned before, the process of generating these Lagrange multipliers for the case where neighboring elements do not have matching orders and/or number of subdivisions, is described in detail later in section 6.3.

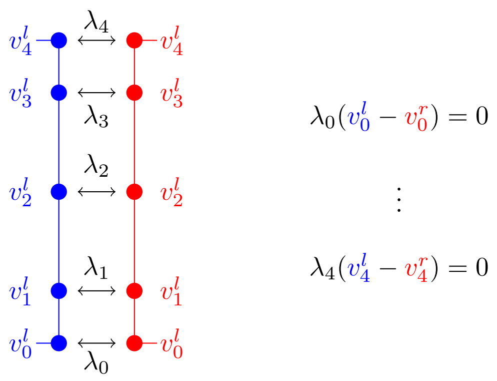


Figure 4.9: Lagrange multipliers enforcing continuity between the left and right element boundary for a 0-form.

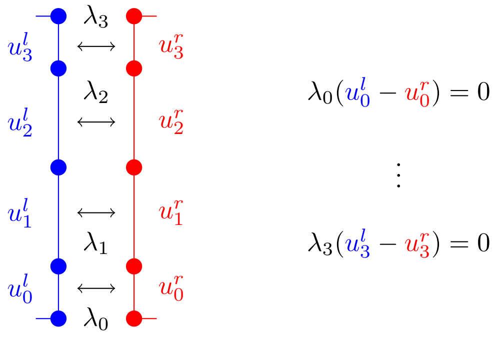


Figure 4.10: Lagrange multipliers enforcing continuity between the left and right element boundary for a 1-form.

The reason for delaying this is that it deeply connected with how the subdivisions are performed and how *hp*-refinement is handled.

Another important note to make about the hybridized approach is that Lagrange multipliers tend to carry physical meaning and are not just some mathematical trick. In fact, when weak derivatives are used, it can be shown that these Lagrange multipliers are in fact boundary degrees of freedom corresponding to fluxes between the elements or the value of the solution on the boundary [8].

As a quick demonstration of how this can manifest, consider a part of a system of equations, involving a k-form $q^{(k)}$ and a $(k+1)$-form $u^{(k+1)}$, where $q^{(k)}$ is the weak derivative of $u^{(k+1)}$. This results in Equation 4.32 being added to the system. The boundary term $\int_{\partial\Omega} p^{(k)} \wedge \star u^{(k+1)}$ provides natural/weak boundary conditions for $u^{(k+1)}$, which are added explicitly for boundaries of elements which are on the boundary of the domain. However, in the hybridized approach these are provided by Lagrange multipliers in the interior of the solution.

$$\left(p^{(k)}, q^{(k)}\right)_\Omega + \left(\mathrm{d}p^{(k)}, u^{(k+1)}\right)_\Omega = \int_{\partial\Omega} p^{(k)} \wedge \star u^{(k+1)} \qquad \forall p^{(k)} \in \Lambda^{(k)} \tag{4.32}$$

4.8 Solution Process

As the consequence of enforcing connectivity through the method described in section 4.7, three distinct possibilities appear for computing the solution to the global system. These are as follows:

- Direct solution using a frontal solver (pivoted sparse LU),
- Direct solution using Schur's compliment when element matrices are non-singular,
- Iterative solution using pre-conditioned BiCGStab (or CG if the matrix is symmetric).

Each of these methods has advantages and drawbacks. In terms of requirements and accuracy, the direct solve with a frontal solver seems the most appealing, as the bandwidth of the global matrix, such as the one shown in Equation 4.31 is low on the block-diagonal part created by element matrices, with the only part with larger bandwidth coming from the blocks that come as a result of Lagrange multipliers.

On the other hand such a solution method very serial, which makes it harder to scale to very large problems. This is not a very pressing concern for this work, as all problems are solved on 2D meshes only, which makes the solution time reasonable for fairly sizable problems.

In comparison, using Schur's compliment involves rewriting the system of equations involving the degrees of freedom and Lagrange multipliers as Equation 4.33 and Equation 4.34. These can be rewritten to eliminate terms with $\vec{u}$ to obtain an equation for the Lagrange multipliers from $\mathbf{N}^T\mathbf{A}^{-1}\mathbf{N}\vec{\lambda} = \mathbf{N}\mathbf{A}^{-1}\vec{v} + \vec{\phi}$. Once these are obtained, the solution for the degrees of freedom can be obtained from Equation 4.33 after adding the terms from Lagrange multipliers.

$$\mathbf{A}\vec{u} + \mathbf{N}^T\vec{\lambda} = \vec{v} \tag{4.33}$$

$$\mathbf{N}\vec{u} = \vec{\phi} \tag{4.34}$$

This might not seem significantly better on its own, but given the structure given in Equation 4.31, if the element matrices are invertible on their own, it means that it can be done on per-element basis. This makes these direct inverses relatively cheap, as well as possible to do it in parallel. The only bottleneck is the need to invert the matrix $\mathbf{N}^T\mathbf{A}^{-1}\mathbf{N}$, which has the dimensions of the Lagrange multiplier count. As mentioned before, the problem must be expressed in the form where the matrix $\mathbf{A}$ is invertible. This is for example not the case with the direct formulation of the Poisson equation, instead requiring the mixed formulation.

Another note on this topic is that while it may be expensive to directly compute the inverse of $\mathbf{N}^T\mathbf{A}^{-1}\mathbf{N}$, the system for $\vec{\lambda}$ can be solved iteratively in a matrix-free fashion. This requires only evaluations of $\mathbf{N}^T\mathbf{A}^{-1}\mathbf{N}$. When the matrix $\mathbf{A}$ is also SPD, it is possible to then use conjugate gradient method to solve the system.

The third option, which was mentioned before, is the use of iterative solvers. First of all, with a large system such as this, preconditioning is a no-brainer, as for this matrix structure block Jacobi for block $\mathbf{A}$ is relatively cheap and identity can be used for the bottom part. As such, it can be combined with BiCGStab iterative solver (or CG if $\mathbf{A}$ is SPD), which requires very low memory overhead for general matrices.

For the solver that was used in this work, all three options were attempted. For the direct solve, SciPy's `sparse.spsolve` was used [36], which under the hood uses a frontal solver from UMFPACK [37]. The solver is written in ANSI C and besides the small overhead to bind it to Python is about as efficient as one could ask for. Given the way the system is structured for two dimensional problems, it was quite clear that this method would be very efficient.

The other two solvers were also implemented to get an idea of how each of these would fare. Both Schur and PCG solvers were implemented in C and integrated into Python, such that the overhead of Python was for the almost entirely negligible. Since Schur and PCG were solved in an iterative way, the stopping criterion was set to allow for the ratio of residual L^2 norm to forcing L^2 norm of $1 \cdot 10^{-15}$, which resulted in the actual solution having the max norm error of around $1 \cdot 10^{-13}$.

To give an idea of performance of each of these for the two dimensional mixed Poisson problem on a square mesh of n by n elements with order p. For each test case a total of 10 runs were performed to try and reduce non-deterministic variations, such as delay when Python's garbage collector might run during the solve. The test was first run by varying the number of elements n, then repeated by varying the order of elements p.

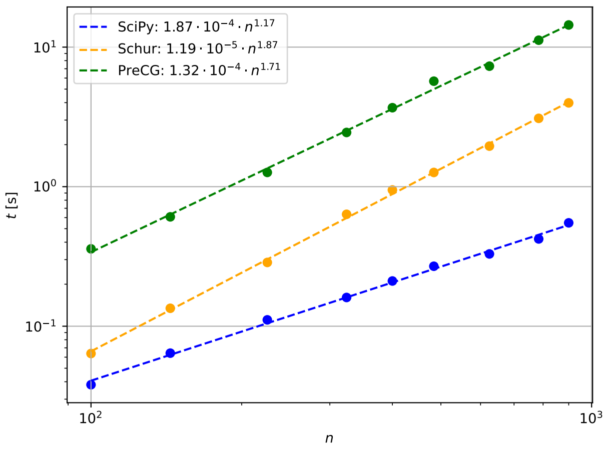
The result of these two tests can be seen in Figure 4.11. The data points are presented along with least squares function fits of log-log or semi-log functions to allow for comparison of scaling with the parameters n and p. The total number of degrees of freedom is roughly Equation 4.35, as number of degrees per element scales roughly with p^2 and with number of elements scaling with n^2. There are additional scaling terms since continuity and boundary degrees of freedom depend on what k-forms are in the system, but the overall scaling still follows that in Equation 4.35.

$$n_{\text{dof}} = \mathcal{O}(p^2)\mathcal{O}(n^2) \tag{4.35}$$

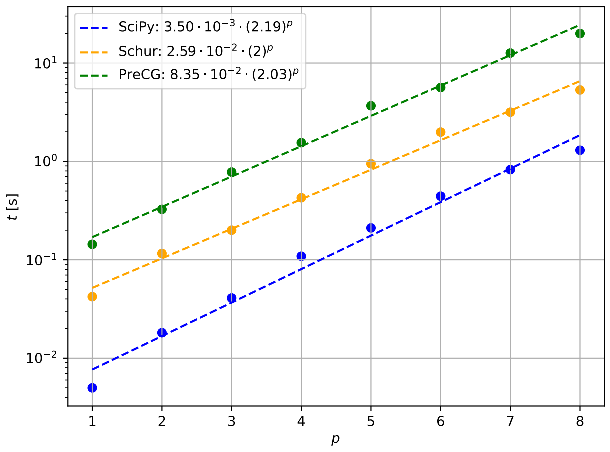
What is quite clear is that time required for the solve scales in the same manner for all solvers depending on whether p or n are varied. This is to be expected, along with the nature of their scaling. The scaling with increasing n is Figure 4.11a is linear in the log-log plot, where scaling with increasing p is linear in the semi-log plot in Figure 4.11b. The reason this difference is the fact that by increasing the number of elements with n the sparsity of the matrix remains about the same, while when p is increased, the resulting matrix is much less sparse.

As for the differences between the solvers themselves, which is the more important topic, it is quite clear based on these graphs, that for the systems of the tested size, SciPy's solver using UMFPACK [37] was the fastest, with the preconditioned conjugate gradient solver taking the most time. An interesting observation, however is how the scaling between these solvers vary. For example, when it comes to scaling with the polynomial order p, SciPy took the least time but had the worst scaling, with time being proportional to about 2.2^p, while Schur having the best scaling. On the other hand, this order was completely reversed when looking at scaling with increasing the number of elements n. There the scaling of the Schur solver was by far the worst, with it being proportional to $n^{1.9}$ and SciPy's scaling being $n^{1.2}$.

While this benchmark is quite decisive, what should also be considered the conditions this test was performed under. Most importantly, all these solvers were run serially on a single core, with the entire system being able to fit in the process' memory. If instead an architecture with distributed memory had to be used, which is likely what it would be the case if a problem in 3D or with a significantly higher number of elements was considered, these solvers would need to be modified. At that point, the scaling could drastically change. For those cases, another similar investigation into scaling and performance of these solvers ought to be considered.



(a) Time needed to solve as a function of element count (in one direction) n for $p = 5$.



(b) Time needed to solve as a function of element order p for $n = 20$.

Figure 4.11: Time needed by different solvers to obtain the solution to the system with comparable accuracy.

Chapter 5

Variational Multi-Scale

Variational multi-scale is class of numerical methods, which have been introduced back in the 1980s, with developments up to the year 2007 formalized and expressed in terms of projection by Hughes and Sangalli [4]. To show how this approach works, consider the variational formulation of the following problem. Consider a vector space V with the dual space $V^{\star}$ and an duality pairing $\langle \cdot, \cdot \rangle : V^{\star} \times V \to \mathbb{R}$. This pairing is defined similarly to inner product for differential k-forms as per Equation 5.1, where the primal space is $\Lambda^{(k)}$ with the dual space $\widetilde{\Lambda}^{(n-k)}$.

$$\left\langle u^{(n-k)}, v^{(k)} \right\rangle = \int_{\Omega^n} u^{(n-k)} \wedge v^{(k)} \qquad u^{(n-k)} \in \widetilde{\Lambda}^{(n-k)} \qquad v^{(k)} \in \Lambda^{(k)} \tag{5.1}$$

Inner product Equation 3.19 can actually be written in terms of this duality pairing as per Equation 5.2. This also highlights a key difference between the duality pairing and the inner product: while the duality pairing is metric-free, the inner product is metric-dependent. This is clearly shown in Equation 5.2, when in order to write inner product in terms of the duality pairing, the Hodge $\star$ must be added.

$$\left\langle u^{(k)}, \star v^{(k)} \right\rangle = \left(u^{(k)}, v^{(k)} \right)_{\Omega^n} \qquad u^{(k)} \in \Lambda^{(k)} \qquad v^{(k)} \in \Lambda^{(k)} \tag{5.2}$$

Now consider the solution of a problem in Equation 5.3, denoted by u. There $\mathcal{L}$ and $\mathcal{C}$ are differential operators, with $\mathcal{L}$ being symmetric positive-definite (SPD) and $\mathcal{C}$ being the remainder.

$$\mathcal{L}u + \mathcal{C}u = f \tag{5.3}$$

This problem can be rewritten in the variational formulation, with $u \in V$, $\mathcal{L} : V \to V^{\star}$, $\mathcal{C} : V \to V^{\star}$, and $f \in V^{\star}$ satisfying Equation 5.4, potentially with some extra boundary conditions.

$$\langle \mathcal{L}u, v \rangle + \langle \mathcal{C}u, v \rangle = \langle f, v \rangle \qquad \forall v \in V \tag{5.4}$$

While Equation 5.4 is exact, it can not be solved numerically, as V is an infinite dimensional vector space. Because of that, it must be divided into $\overline{V}$ and V', such that $V = \overline{V} \oplus V'$ and $\overline{V}$ is finite-dimensional. The solution for $\overline{v} \in \overline{V}$ can then be solved for, since being in a vector space with a finite number of dimensions means it is possible to represent it with a finite number of degrees of freedom and formulate a finite number of equations to solve for these.

While it was initially intended to implement VMS according to work by Shrestha et al. [25] and the theory presented in section 5.1 and section 5.2, it was not possible to finish the implementation due to time constraints. As such, results of VMS are imitated according to section 5.3.

5.1 Projectors

This split of Equation 5.4 is requires a projector $\mathcal{P} : V \to \overline{V}$, for which it is required that must be impotent $\mathcal{P}\overline{v} = \overline{v}$, $\forall \overline{v} \in \overline{V}$ and bounded, meaning that $||\mathcal{P}u|| \leq ||u||$. This then gives a split of the space V into the resolved space $\mathrm{Range}(\mathcal{P}) = \overline{V}$ and the unresolved space $V' = \mathrm{Ker}(\mathcal{P})$. This is a unique split, such that $V = \overline{V} \oplus V'$.

The result of this is that Equation 5.4 can be split into two equations - Equation 5.5 with the test space of $\overline{V}$, and Equation 5.6) for the test space V'. Since Equation 5.5 has $\overline{v} \in \overline{V}$, which is a finite function space, it gives a finite system of equations, which can be solved.

$$\langle \mathcal{L}u, \overline{v} \rangle + \langle \mathcal{C}u, \overline{v} \rangle = \langle f, \overline{v} \rangle \qquad \forall \overline{v} \in \overline{V} \tag{5.5}$$

$$\langle \mathcal{L}u, v' \rangle + \langle \mathcal{C}u, v' \rangle = \langle f, v' \rangle \qquad \forall v' \in V' \tag{5.6}$$

There are many ways to define such a projector, and in turn different splits of function spaces, but for the purposes of this work, the projectors of interest are based on differential operators. Consider now the symmetric positive operator $\mathcal{L}$. Due to symmetry, it can be shown that it is the result of minimization of some energy function [26]. From there, it is possible to define a norm associated with this energy functional's inner product $||\cdot||_{\mathcal{L}}$. With this norm, a projector $\mathcal{P}_{\mathcal{L}}$ can be defined as finding the function $\overline{u} \in \overline{V}$, which minimizes the error to the real solution u in the energy norm, as per Equation 5.7.

$$\mathcal{P}_{\mathcal{L}} u = \underset{\overline{u} \in \overline{V}}{\arg\min} \, \frac{1}{2} \, ||u - \overline{u}||_{\mathcal{L}}^2 \tag{5.7}$$

Note that it is key that the operator $\mathcal{L}$ and its discrete version $\overline{\mathcal{L}}$ are both invertable, with inverse operators $\mathcal{L}^{-1}$ and $\overline{\mathcal{L}}^{-1}$ respectively. With these, a function can be mapped from V to $V^\star$ and back, and the same for their discrete counterparts. This is illustrated in Figure 5.1.

$$\begin{tikzcd}
V \arrow[d, bend right, "\mathcal{L}"'] & \overline{V} \arrow[d, bend left, "\overline{\mathcal{L}}"] \\
V^\star \arrow[u, bend right, "\mathcal{L}^{-1}"'] & \overline{V}^\star \arrow[u, bend left, "\overline{\mathcal{L}}^{-1}"]
\end{tikzcd}$$

Figure 5.1: Differential operators and the spaces they map between

Before going any further with fully defining the resulting projector, some other important concepts must be addressed. Namely, these are the formal and Hermitian adjoint of a projector, since these are crucial for being able to properly define an operator-based projector $\mathcal{P}_{\mathcal{L}}$.

5.1.1 Adjoints of Projectors

There are two adjoints of a projectors that are relevant in this work. First is the formal adjoint and the second is the Hermitian adjoint. The formal adjoint of projector $\mathcal{P}$, which is denoted by $\mathcal{P}^T$, is defined with respect to the duality pairing $\langle \cdot, \cdot \rangle$ as Equation 5.8. This makes it a map from $\overline{V}^\star$ to $V^\star$.

$$\langle \mathcal{P}u, \overline{v} \rangle = \left\langle u, \mathcal{P}^T \overline{v} \right\rangle \qquad \forall \overline{v} \in \overline{V}^\star \tag{5.8}$$

On the other hand, the Hermitian adjoint $\widehat{\mathcal{P}}$ is defined with respect to the inner product $(\cdot, \cdot)_\Omega$ as per Equation 5.9. This makes the Hermitian adjoint instead a mapping from $\overline{V}$ to V.

$$(\mathcal{P}u, \overline{v})_\Omega = \left(u, \widehat{\mathcal{P}} \overline{v}\right)_\Omega \qquad \forall \overline{v} \in \overline{V} \tag{5.9}$$

Of course, one could also construct a Hermitian adjoint of the formal adjoint $\widehat{\mathcal{P}}^T$, which would make it a mapping from $V^\star$ to $\overline{V}^\star$. These four allow for operations that connect the infinite-dimensional spaces V and $V^\star$ with their discrete counterparts $\overline{V}$ and $\overline{V}^\star$. This is shown in Figure 5.2.

$$\begin{tikzcd}
V \arrow[r, bend left, "\mathcal{P}"] & \overline{V} \arrow[l, bend left, "\widehat{\mathcal{P}}"]
\end{tikzcd}$$

$$\begin{tikzcd}
V^{\star} \arrow[r, bend right, "\widehat{\mathcal{P}}^T"'] & \overline{V}^{\star} \arrow[l, bend right, "\mathcal{P}^T"']
\end{tikzcd}$$

Figure 5.2: Projector operators and the spaces they map between

Given that when working with Hilbert spaces, the L^2 norm is always available and defined, the L^2 projectors can be defined and used as basis for defining other operator-based projectors. If the L^2 norm is used with Equation 5.7, the resulting definition of the L^2 projector as the projector which minimizes the error between u and $\overline{u}$ in the L^2 norm is given by Equation 5.10.

$$(\mathcal{P}_{L^2} u, \overline{v})_\Omega = (\overline{u}, \overline{v})_\Omega = (u, \overline{v})_\Omega \qquad \forall \overline{v} \in \overline{V} \tag{5.10}$$

Implementation of the L^2 Projectors

Consider projection of k-form $\widetilde{u}^{(k)}$ from space $\widetilde{V}$, such that $\overline{V} \subset \widetilde{V}$. In this case, the L^2 projection problem Equation 5.10 can be rewritten by expanding the original k-form $\widetilde{u}^{(k)}$ and the projection $\overline{u}^{(k)}$ in terms of basis functions and degrees of freedom. Expanding the projection changes Equation 5.10 into Equation 5.11, which after also expanding the original form $\widetilde{u}$ in terms of basis Equation 5.12.

$$\begin{bmatrix} \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \overline{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \overline{\psi}_N^{(k)} \\ \vdots & \ddots & \vdots \\ \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \overline{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \overline{\psi}_N^{(k)} \end{bmatrix} \begin{bmatrix} \overline{\mathcal{N}}_1^{(k)}(u^{(k)}) \\ \vdots \\ \overline{\mathcal{N}}_N^{(k)}(u^{(k)}) \end{bmatrix} = \begin{bmatrix} \int_\Omega \widetilde{u}^{(k)} \wedge \star \overline{\psi}_1^{(k)} \\ \vdots \\ \int_\Omega \widetilde{u}^{(k)} \wedge \star \overline{\psi}_N^{(k)} \end{bmatrix} \tag{5.11}$$

$$\begin{bmatrix} \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \overline{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \overline{\psi}_N^{(k)} \\ \vdots & \ddots & \vdots \\ \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \overline{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \overline{\psi}_N^{(k)} \end{bmatrix} \begin{bmatrix} \overline{\mathcal{N}}_1^{(k)}(u^{(k)}) \\ \vdots \\ \overline{\mathcal{N}}_N^{(k)}(u^{(k)}) \end{bmatrix} = \begin{bmatrix} \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \widetilde{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_1^{(k)} \wedge \star \widetilde{\psi}_M^{(k)} \\ \vdots & \ddots & \vdots \\ \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \widetilde{\psi}_1^{(k)} & \cdots & \int_\Omega \overline{\psi}_N^{(k)} \wedge \star \widetilde{\psi}_M^{(k)} \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{N}}_1^{(k)}(u^{(k)}) \\ \vdots \\ \widetilde{\mathcal{N}}_N^{(k)}(u^{(k)}) \end{bmatrix} \tag{5.12}$$

If $\widetilde{V}$ is actually the full infinite-dimensional space V, then this $\widetilde{u}$ is represented by the infinite number of degrees of freedom and and infinite number of basis, however when $\widetilde{V}$ is also discrete, though finer than V, this makes the entire system discrete. It can be written in the matrix notation as Equation 5.13. This system can be rearranged to write the projector as a matrix which transforms degrees of freedom of k-forms from $\widetilde{V}$ to k-forms from $\overline{V}$, as per Equation 5.14.

$$\overline{\mathbf{M}}^{(k)} \vec{\overline{\mathcal{N}}}^{(k)} \left(u^{(k)}\right) = \mathbf{N}^{(k)} \vec{\widetilde{\mathcal{N}}}^{(k)} \left(u^{(k)}\right) \tag{5.13}$$

$$\mathbf{P}^{(k)} = \left(\overline{\mathbf{M}}^{(k)}\right)^{-1} \mathbf{N}^{(k)} \tag{5.14}$$

The formal adjoint of the L^2 projector $\mathcal{P}^T$ is actually given by just transposing the matrix $\mathbf{P}^{(k)}$ from Equation 5.14. This can be shown by using the definition of the formal adjoint projector in Equation 5.8. This gives the matrix which transforms the of k-form from $\overline{V}^\star$ to $V^\star$ as Equation 5.15. Note that the star $\star$ symbol is used to denote the adjoint projector as to not confuse the matrix transpose.

$$\left(\mathbf{P}^{(k)}\right)^\star = \left(\mathbf{N}^{(k)}\right)^T \left(\overline{\mathbf{M}}^{(k)}\right)^{-1} = \left(\left(\overline{\mathbf{M}}^{(k)}\right)^{-1} \mathbf{N}^{(k)}\right)^T = \left(\mathbf{P}^{(k)}\right)^T \tag{5.15}$$

The Hermitian adjoint projector and the formal adjoint of the Hermitian adjoint projector can be derived from the definition of the formal adjoing in Equation 5.8 and using the relation between the inner product and the duality pairing in Equation 5.2. Consider a primal k-form $u \in V$ and a dual k-form $\overline{v} \in \overline{V}^\star$. By inserting theim into the definition of the formal adjoint in Equation 5.8, it is possible to reformulate the relation as an inner product and the original projector $\mathcal{P}$. At that point the Hermitian transpose can be substituted in from Equation 5.9, which results in Equation 5.16.

$$\left\langle u, \mathcal{P}^T \overline{v} \right\rangle = \left\langle \mathcal{P} u, \overline{v} \right\rangle = \left(\mathcal{P} u, \star \overline{v}\right)_\Omega = \left(u, \widehat{\mathcal{P}} \star \overline{v}\right)_\Omega = \left\langle u, \star \widehat{\mathcal{P}} \star \overline{v} \right\rangle \tag{5.16}$$

As such it can be seen that the Hodge adjoint of a projector is pre- and post-Hodged formal adjoint, meaning that $\star \widehat{\mathcal{P}} \star = \mathcal{P}^T$. Hodge operator which takes degrees of freedom from primal discrete to dual discrete space is a mass matrix or its inverse, this results in the matrix which applies this to the degrees of freedom as defined by Equation 5.17.

$$\widehat{\mathbf{P}}^{(k)} = \left(\widetilde{\mathbf{M}}^{(k)}\right)^{-1} \left(\mathbf{P}^{(k)}\right)^\star \overline{\mathbf{M}}^{(k)} = \left(\widetilde{\mathbf{M}}^{(k)}\right)^{-1} \left(\overline{\mathbf{N}}^{(k)}\right)^T = \left(\widetilde{\mathbf{M}}^{(k)}\right)^{-1} \left(\mathbf{N}^{(k)}\right)^T \tag{5.17}$$

The last to be defined in this section is the formal adjoint of the Hermitian adjoint, denoted by $\widehat{\mathcal{P}}^T$. Similar to how the formal adjoint of the adjoint is defined using the duality pairing, this is also defined using the duality pairing, but involving the Hermitian adjoint $\widehat{\mathcal{P}}$, as per Equation 5.18.

$$\left\langle \overline{u}, \widehat{\mathcal{P}} v \right\rangle = \left\langle \widehat{\mathcal{P}}^T \overline{u}, v \right\rangle \qquad \forall \overline{u} \in \overline{V} \qquad \forall v \in V^\star \tag{5.18}$$

When written in terms of matrices it is no surprise that the matrix $\left(\widehat{\mathbf{P}}^{(k)}\right)^\star$ which applies it is the transpose of the one which applies the Hermitian transpose $\widehat{\mathbf{P}}^{(k)}$, just as how the one which applies the formal transpose of the base projector is applied by the transpose of the matrix which applies the projector.

$$\left(\widehat{\mathbf{P}}^{(k)}\right)^\star = \overline{\mathbf{N}}^{(k)} \left(\widetilde{\mathbf{M}}^{(k)}\right)^{-1} = \left(\left(\widetilde{\mathbf{M}}^{(k)}\right)^{-1} \left(\overline{\mathbf{N}}^{(k)}\right)^T\right) = \left(\widehat{\mathbf{P}}^{(k)}\right)^T \tag{5.19}$$

With this, all the projectors from Figure 5.2 have been implemented for L^2 norm based projector. This allows construction of arbitrary SPD operator based projectors in subsection 5.1.2.

5.1.2 Definition of Operator-Based Projectors

With the L^2 projectors taken care of, $\mathcal{L}$ operator based projectors can be defined. Consider the case when some u is given, which can be inserted into Equation 5.7 to obtain the projection $\mathcal{P}_\mathcal{L} u = \overline{u}$ via variation formulation. It is defined by Equation 5.7 with using the L^2 norm. In that case the differential operator is $\mathcal{L} = \star$, which just maps the k-form u from primal to dual space.

Since $\mathcal{L}$ arises from the variational formulation of minimization of the energy functional used to define the norm $||\cdot||_\mathcal{L}$, it is quite clear that this gives rise to Equation 5.20.

$$\left\langle \overline{\mathcal{L}} \mathcal{P}_\mathcal{L} u, \overline{v} \right\rangle = \left\langle \overline{\mathcal{L}} \overline{u}, \overline{v} \right\rangle = \left\langle \widehat{\mathcal{P}}^T_{L^2} \mathcal{L} u, \overline{v} \right\rangle \qquad \forall \overline{v} \in \overline{V} \tag{5.20}$$

In Equation 5.20, $\overline{\mathcal{L}}$ is the discrete differential operator based on restricting $\mathcal{L}$ to the discrete space, defined as $\overline{\mathcal{L}} : \overline{V} \to \overline{V}^\star$. This can then be reformulated in terms of operators as per Equation 5.21, with $\overline{\mathcal{L}}^{-1} : \overline{V}^\star \to \overline{V}$ being the inverse operator of $\overline{\mathcal{L}}$. The action of this projector broken down into its constituent operations is illustrated in Figure 5.3.

$$\begin{tikzcd}
V \arrow[r, bend left, "\mathcal{P}_{\mathcal{L}}"] \arrow[d, bend right, "\mathcal{L}"'] & \overline{V} \\
V^{\star} \arrow[r, bend right, "\widehat{\mathcal{P}}_{L^2}^T"'] & \overline{V}^{\star} \arrow[u, bend right, "\overline{\mathcal{L}}^{-1}"']
\end{tikzcd}
\qquad \mathcal{P}_{\mathcal{L}} = \overline{\mathcal{L}}^{-1}\widehat{\mathcal{P}}_{L^2}^T\mathcal{L}$$

Figure 5.3: Projector $\mathcal{P}_{\mathcal{L}}$ with its building blocks.

$$\mathcal{P}_{\mathcal{L}} = \overline{\mathcal{L}}^{-1}\widehat{\mathcal{P}}_{L^2}^T\mathcal{L} \tag{5.21}$$

Similar to how the projector based on the operator $\mathcal{L}$ can be written in terms of operations of differential operators based on $\mathcal{L}$ as shown in Figure 5.1 and operations based on the L^2 projector $\mathcal{P}_{L^2}$ in Figure 5.2, it is possible to write all other $\mathcal{L}$-based projector operations in terms of those. For the sake of completeness these are listed here. The base $\mathcal{L}$ based projector, which was already defined in Equation 5.21 is repeated in Equation 5.22, with its formal adjoint in Equation 5.23, the Hermitian adjoint in Equation 5.24, and lastly the formal adjoint of its Hermitian adjoint in Equation 5.25.

$$\mathcal{P}_{\mathcal{L}} = \overline{\mathcal{L}}^{-1}\widehat{\mathcal{P}}_{L^2}^T\mathcal{L} \tag{5.22}$$
$$\mathcal{P}_{\mathcal{L}}^T = \mathcal{L}\widehat{\mathcal{P}}_{L^2}\overline{\mathcal{L}}^{-1} \tag{5.23}$$
$$\widehat{\mathcal{P}}_{\mathcal{L}} = \mathcal{L}^{-1}\mathcal{P}_{L^2}^T\overline{\mathcal{L}} \tag{5.24}$$
$$\widehat{\mathcal{P}}_{\mathcal{L}}^T = \overline{\mathcal{L}}\mathcal{P}_{L^2}\mathcal{L}^{-1} \tag{5.25}$$

5.2 Fine-Scale Green's Function

This split of Equation 5.4 is requires a projector $\mathcal{P} : V \to \overline{V}$, which uniquely splits the space V into $\overline{V}$ and V'. Based on the projector $\mathcal{P}$, these are defined as $\text{Range}(\mathcal{P}) = \overline{V}$ and $V' = \text{Ker}(\mathcal{P})$. From there, the solution u can be split into the coarse-scales/resolved part $\overline{u}$ and the fine-scales/unresolved part u'. Since the split $V = \overline{V} \oplus V'$ is unique, so is this split of the solution $u = \overline{u} + u'$. If $\mathcal{L}$ and $\mathcal{C}$ are linear or are linearized, this allows for Equation 5.4 to be split up into Equation 5.26.

$$\langle \mathcal{L}\overline{u}, v\rangle + \langle \mathcal{L}u', v\rangle + \langle \mathcal{C}\overline{u}, v\rangle + \langle \mathcal{C}u', v\rangle = \langle f, v\rangle \qquad \forall v \in V \tag{5.26}$$

To be able to solve this problem, the test space for v must also be taken to be finite dimensional. If the split of V into $\overline{V}$ and V' is also used here, this yields the problem that must be solved as in Equation 5.27.

$$\langle \mathcal{L}\overline{u}, \overline{v}\rangle + \langle \mathcal{L}u', \overline{v}\rangle + \langle \mathcal{C}\overline{u}, \overline{v}\rangle + \langle \mathcal{C}u', \overline{v}\rangle = \langle f, \overline{v}\rangle \qquad \forall \overline{v} \in \overline{V} \tag{5.27}$$

The traditional Galerkin formulation would then proceed to ignore the terms resulting from the contributions of unresolved part of the solution u'. With VMS, contributions of u' are instead accounted for in a way which allows for some quite strong guarantees on the properties of $\overline{u}$. It was shown by Shrestha et al. [25] that the coarse scale solution $\overline{u}$ can be made to be exact in the norm induced by the SPD operator $\mathcal{L}$, by including the unresolved scale contributions.

These are obtained by applying the Suyash-Green operator to the residual of the fine scales on the coarse equations through the remaining operator $\mathcal{C}$, as given by Equation 5.28. The fine-scales Green operator $\mathcal{G}'$ is given by Equation 5.29.

$$\mathcal{C}u' = \sigma_{\text{SG}}^{\mathcal{C}}\mathcal{R}\overline{u} = \left(1 + \mathcal{C}\mathcal{G}'\right)^{-1}\mathcal{C}\mathcal{G}'\left(f - \mathcal{L}\overline{u}\right) \tag{5.28}$$

$$\mathcal{G}' = \mathcal{L}^{-1} - \widehat{\mathcal{P}}_{L^2}\overline{\mathcal{L}}^{-1}\widehat{\mathcal{P}}_{L^2}^{T} \tag{5.29}$$

Due to the resolved solution being exact in the norm induced by $\mathcal{L}$, contribution of $\mathcal{L}u'$ in $\overline{V}$ is zero, so Equation 5.27 can be rewritten as Equation 5.30.

$$\langle\mathcal{L}\overline{u}, \overline{v}\rangle = \langle f, \overline{v}\rangle + \langle\mathcal{C}\overline{u}, \overline{v}\rangle + \left\langle\sigma_{\text{SG}}^{\mathcal{C}}\overline{u}, \overline{v}\right\rangle \qquad \forall\overline{v} \in \overline{V} \tag{5.30}$$

Theoretically u' live in the infinite-dimensional space V', such that $V = \overline{V} \oplus V'$, with operators involved in Equation 5.29 dealing with this infinite-dimensional space (or its infinite-dimensional dual space), it is again not possible to compute anything there. As such, a discrete approximation of it $\widetilde{V}$ must be taken, such that $\overline{V} \subset \widetilde{V} \subset V$. As such, at least u' residing in $\widetilde{V}$ can be resolved. The relation between these spaces is illustrated by the diagram Figure 5.4.

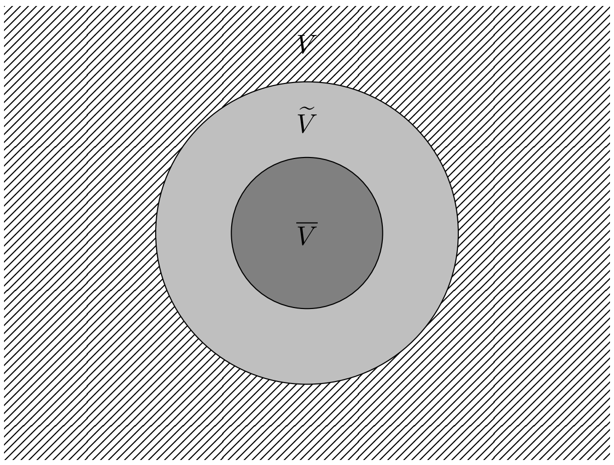


Figure 5.4: Illustration of relationship between the different function space $\overline{V}$, $\widetilde{V}$, and V.

The simplest way to generate this space $\widetilde{V}$ is to increment the polynomial order of each element by some integer Δp. This value need not be too high, as Shrestha et al. [25] showed that at $\Delta p = 3$ and $\Delta p = 4$ the results were incredibly close to theoretical results if $\widetilde{V} = V$.

It is also possible to use the fine-scale Green's function in Equation 5.29 to recover unresolved scales from Equation 5.31. In the case where infinite-dimensional space is used for operators and V', this would mean that by adding u' it is possible to obtain the exact solution since $u = \overline{u} + u'$. However, in the case where discrete $\widetilde{V}$ is used instead, the highest accuracy of u that can be obtained is the solution that would be obtained by using the test space $\widetilde{V}$.

$$u' = \mathcal{G}'\left(f - \mathcal{C}\overline{u} - \mathcal{C}u'\right) \tag{5.31}$$

5.3 Imitating VMS

Due to time constraints on the thesis, VMS was not implemented in the solver. To still be able to compare its effectiveness when using it as an error refinement criterion, it was instead imitated, by obtaining its results in post-processing. It was shown in work by Shrestha et al. [25], that VMS results computed with finite approximations to the infinite-dimensional space converge exponentially to the analytical results, as order of approximation of such space is increased. Based on those results, the order difference of $\Delta p = 3$ was assumed to be sufficient.

As for the results themselves, the optimally projected solution $\overline{u}$ on the coarse mesh is taken for all test cases as what is computed by solving the projection problem associated with the PDE, instead of the PDE itself. For the unresolved scales u', the values used are the difference between the projection of order p and order $p + \Delta p$, since unresolved scales may not be of higher order than the highest space used for calculations.

As such, the numerical tests that used VMS used $\overline{u}$ to be computed to be the solution of equation Equation 5.32, meaning that an $\mathcal{L}$-based projector was used to exactly compute $\overline{u}$.

$$\langle \mathcal{L}\overline{u}, \overline{v} \rangle = \left\langle \widehat{\mathcal{P}}_{L^2}^{T} \mathcal{L}u, \overline{v} \right\rangle \qquad \forall \overline{v} \in \overline{V} \tag{5.32}$$

For the fine scale equation, the difference of the coarse solution $\overline{u}$ from Equation 5.32 and the fine-scale projection was used. When written in terms of projectors, this is written as Equation 5.33, where $\widetilde{u}$ is the solution in the fine space $\widetilde{V}$.

$$u' = \left(1 - \widehat{\mathcal{P}}_{\mathcal{L}} \mathcal{P}_{\mathcal{L}}\right) \widetilde{u} \tag{5.33}$$

Chapter 6

Refinement

Local refinement can be used within the context of MSEM in order to try and improve how resolved the solution is with a smaller increase in computational cost compared to just increasing the number of cells in the mesh uniformly or increasing the polynomial order of the solution everywhere. Doing this on an individual element is not particularly difficult, however, doing so in a way which preserves the continuity of the solution and keeps the problem well posed is another issue.

MSEM provides boundary conditions from one spectral element to the other by adding constraints to ensure degrees of freedom on one element's boundary match those on the other element's boundary. This is almost trivial if elements which border each other are of the same order and share the entire boundary, with degrees of freedom being exactly equal between the two, however in order to be able to locally refine either the mesh (h-refinement) or the polynomial order (p-refinement), it must also be able to deal with cases where that may not hold.

Refinement of an element, as illustrated in Figure 6.1, requires three main components, which are all covered by research questions to be answered in this section. These are to first add theoretical understanding of how local refinement is done within MSEM, which will be covered by section 6.1 for the case of p-refinement and by section 6.2 for the case of h-refinement. With that, combined hp-refinement is addressed in section 6.3.

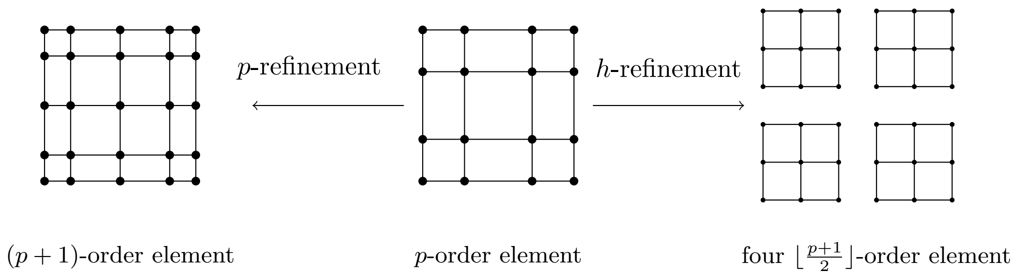


Figure 6.1: Example of an element being h-refined and p-refined

With the knowledge of which refinement option to take for each element that will be refined. With assumption that an adequately accurate error estimate is given, section 6.4 motivates, derives, and validates a refinement criterion which allows for a choice between performing h-refinement or p-refinement.

The last of the three research questions to be answered is to finally find how well different error estimators are suited to be used in this setup. This is examined in section 6.5, where different error estimators are presented and clearly defined.

6.1 p-refinement

The easiest way to perform local refinement is to simply increase the polynomial order of the solution in an element. This is known as the p-refinement. This inevitably causes a mismatch of the number of degrees of freedom at the boundaries with neighbouring elements which do not have the same polynomial order. This section will cover what is done in the 2D case, where a boundary of a cell is a line, however, everything done here can easily be extended to higher dimensions, though it introduces even more book-keeping.

The underlying principle for continuity k-forms is exactly the same: both elements must agree on degrees of freedom on that boundary. If the element with the higher order has order p and the one with the lower (or equal) order has order q, then the highest number of degrees of freedom, which can be matched is that of order q. In theory, the boundary interface could be taken to even be 0-th order, however in this work the highest possible order is chosen, which means that order q is used. Using the highest possible order for the boundary interface gives the highest possible accuracy.

Based on this, the constraint equations take the form of Equation 6.1. The order of the k-form and degrees of freedom is omitted, with superscripts l and r indicating the solution from the left and right element respectively. In the case where the order of the two elements is the same (meaning $p = q$), this simplifies in the case discussed in section 4.7, where degrees of freedom from the two boundaries are matched. Enforcing these constraints effectively lowers the order of the higher element's boundary to that of the lower order element.

$$\lambda_i \left(\mathcal{N}_i(u^l) - \mathcal{N}_i(u^r)\right) = 0 \tag{6.1}$$

6.1.1 Prerequisites

Before diving deeper into p-refinement, this subsection reviews relevant properties of basis used in MSEM, which are important for implementation and understanding the other subsections. First, the choice of basis in MSEM is based on the fact that a basis of a k-form should have duality pairing with the degrees of freedom in the form $\mathcal{N}_i^{(k)}\left(\psi_j^{(k)}\right) = \delta_{i,j}$. Since degrees of freedom for a 0-form are nodal values at nodes $\Xi = \{\xi_1, \ldots, \xi_N\}$, their basis are Lagrange polynomials.

Next, the edge basis for the same element $\Psi^{(1)} = \left\{\psi_1^{(1)}(\xi), \ldots, \psi_{N-1}^{(1)}(\xi)\right\}$ instead have a the relation with pairs nodes in Ξ by definition:

$$\int_{\xi_j}^{\xi_{j+1}} \psi_i^{(1)}(\xi) d\xi = \delta_{i,j}. \tag{6.2}$$

These can be rewritten in terms of nodal basis $\Psi^{(0)}$, due to construction:

$$\psi_i^{(1)} = -\sum_{k=1}^{i} \frac{d\psi_k^{(0)}}{d\xi}. \tag{6.3}$$

Another way to put this is that their integrals can be written as difference of sums of nodal basis $\Psi^{(0)}$ at the two integration nodes:

$$\int_{\xi_A}^{\xi_B} \psi_i^{(1)}(\xi) d\xi = \sum_{k=1}^{i} \left(\psi_k^{(0)}(\xi_A) - \psi_k^{(0)}(\xi_B)\right). \tag{6.4}$$

6.1.2 Continuity of 0-forms

With the mathematical preliminaries out of the way, the continuity of 0-forms can now dealt with. First remark is the fact that corner nodes will always coincide, thus the relations for continuity is a simple equality. The situation is a bit more complicated for the nodes on the interior of the boundary.

Consider the two elements (1) and (2), which are of orders N_1 and N_2, where $N_1 > N_2$. In the case of total continuity, we would have the following relation:

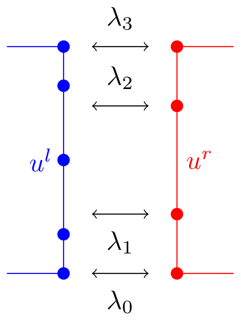
$$\sum_{i=1}^{N_1} u_i^1 \psi_i^{(0),1} = \sum_{i=1}^{N_2} u_i^2 \psi_i^{(0),2}. \tag{6.5}$$

This relation however can not hold for all ξ, except for the trivial case where $N_1 = N_2$, where $u_i^1 = u_i^2$. Instead, we can constrain the higher order element to match the solution on the lower order element. Continuity would not work the other way around, as it can not be guaranteed that the solution on the higher order element could be exactly represented on the lower order element.

If we choose to do so at the nodes of the higher order element Ξ^1, the expression Equation 6.5 becomes much simpler:

$$u_j^1 = \sum_{i=1}^{N_2} u_i^2 \psi_i^{(0),2}(\xi_j^1). \tag{6.6}$$

This allows for enough equations to provide boundary conditions and continuity for 0-forms on the fine element by constraining it to be the same as the lower order solution at its nodes. The diagram of this is shown in Figure 6.2.



$$0 = \lambda_i \left(\mathcal{N}_i^r(u^r) - \mathcal{N}_i^r(u^l)\right) = \lambda_i \left(u_i^r - \sum_{j=0}^{p^r} u_j^l \mathcal{N}_i^r(\psi_j^l)\right)$$

Figure 6.2: The degrees of freedom on the higher order element are constrained to match the values of lower order element at element's nodes.

6.1.3 Continuity of 1-forms

For continuity of 1-forms, the approach follows similar logic as 0-forms. Once again, let elements (1) and (2) have orders N_1 and N_2 where $N_1 \leq N_2$. The continuity requirement for complete continuity would once again be:

$$\sum_{i=1}^{N_1-1} v_i^1 \psi_i^{(1),1} = \sum_{i=1}^{N_2-1} v_i^2 \psi_i^{(1),2}. \tag{6.7}$$

To follow in the steps done for the continuity of 0-forms, the approach taken here is to obtain equations which constrain the degrees of freedom on the higher order element (1) to match the solution on the lower order element (2). In this case, this can be done by integrating over intervals between nodes on the finer element Ξ^1:

$$v_j^1 = \int_{\xi_j^1}^{\xi_{j+1}^1} \sum_{i=1}^{N_2-1} v_i^2 \psi_i^{(1),2} d\xi = \sum_{i=1}^{N_2-1} v_i^2 \int_{\xi_j^1}^{\xi_{j+1}^1} \psi_i^{(1),2} d\xi. \tag{6.8}$$

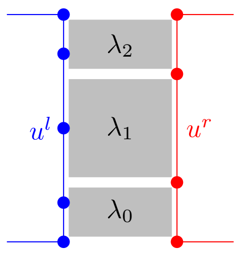
While evaluation these integrals on parts of the boundary might seem like tedious work, expression Equation 6.4 can be substituted in Equation 6.8 to express everything just in terms of nodal basis Ψ^2:

$$v_j^1 = \sum_{i=1}^{N_2-1} v_i^2 \left(\sum_{k=1}^{i} \left(\psi_k^{(0),2}(\xi_j) - \psi_k^{(0),2}(\xi_{j+1})\right)\right) = \sum_{i=1}^{N_2-1} v_i^2 c_i^{(1,2)}. \tag{6.9}$$

This expression is simple to compute, especially since there is a simple recurrence relation for coefficients $c_i^{(1,2)}$:

$$c_i^{(1,2)} = c_{i-1}^{(1,2)} + \left(\psi_i^{(0),2}(\xi_j) - \psi_i^{(0),2}(\xi_{j+1})\right). \tag{6.10}$$

These equations then enforce continuity by forcing the 1-form reconstruction on the higher order element (1) match that on the lower order element (2), similar to what was done for the 0-forms. The diagram of this is shown in Figure 6.3, where the left element has a higher order than the right element.



$$0 = \lambda_i \left(\mathcal{N}_i^r(u^r) - \mathcal{N}_i^r(u^l)\right) = \lambda_i \left(u_i^r - \sum_{j=0}^{p^r} u_j^l \mathcal{N}_i^r(\psi_j^l)\right)$$

Figure 6.3: The degrees of freedom on the higher order element are constrained to match the values of lower order element as integrals over edges.

6.2 *h*-refinement

Performing *h*-refinement means subdividing elements into smaller ones in order to improve the mesh resolution. This can be done in many different ways, some better than others. For this thesis, only one of these will be examined, though it is possible to use some other variation of it.

At the core of this method is the idea of taking a parent element E_p which is subdivided further into four child elements $E_c = \{E_{1,1}, E_{1,2}, E_{2,1}, E_{2,2}\}$ of equal size, as per Figure 6.4. Such a configuration allows for storing the elements as a quad-tree, which in 3D would be an oct-tree. Technically, if unidirectional refinement was a highly desirable property, the method could be done with only uni-directional divisions, which would result in a binary tree.

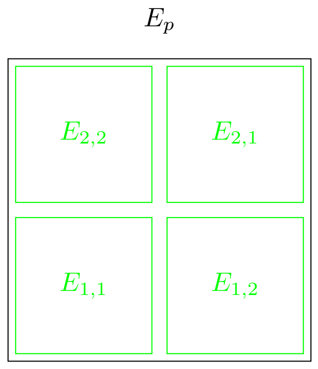


Figure 6.4: Parent element E_p and its children, which are drawn a bit smaller to not overlap.

The main problem which has to be overcome for *h*-refinement is to deal with the case where two boundaries with subdivisions must be connected. This can be dealt with in a few different ways. It should also be kept in mind that because this should ultimately also be compatible with *p*-refinement, it would be ideal if there was no requirement on what orders of the boundaries must be.

Based on this, consider the case as illustrated by diagram in Figure 6.5. In that example, the boundary of one top level element is first split once, then the right side is split again, with the left part of the newly created quarter divided again. The boundary of the bottom element is split only once for contrast.

The first observation that can be made is that the left part of both boundaries actually match in that there's no further subdivisions on those child elements. As such, it makes sense to treat this as a regular boundary resulting from *p*-refined, since orders of these might differ. Next, the second half has to be dealt with. Now the situation arises where in this example the bottom element has no more subdivisions, while the top one still has one or more. This brings the motivation for what will henceforth be referred to as *boundary merging*.

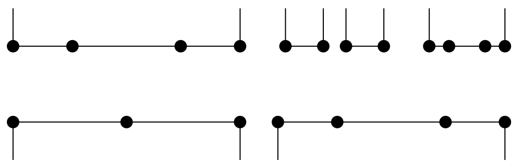


Figure 6.5: Example of two boundaries with varying levels of subdivisions.

6.2.1 Boundary Merging

Consider an even more extreme difference in subdivisions than in Figure 6.5, presented in Figure 6.6. The problem that needs to be overcome here is how can the boundary with different levels of refinement be connected to the one with no subdivision. This is solved by merging the divided boundary into a single boundary, since that allows to then just use the relations used for p-refinement to merge the two boundaries with (potentially) differing number of degrees of freedom.

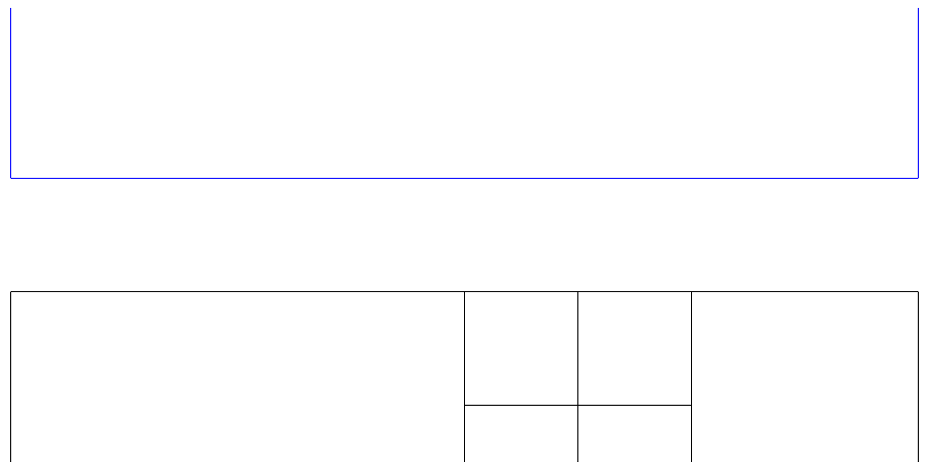


Figure 6.6: Boundaries with different refinement levels.

To perform this merge, first step is introduction of a new boundary with the order of basis being the sum of basis order along the boundaries of the subdivisions, which is illustrated in Figure 6.7. These boundaries have the same number of (unique) degrees of freedom for 0-forms and 1-forms.

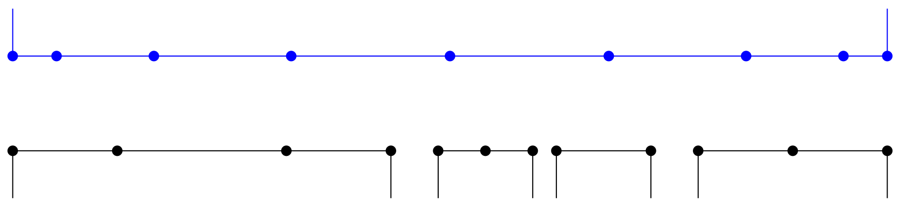


Figure 6.7: Boundaries with different refinement levels and example degrees of freedoms on them.

6.2.2 Continuity of 0-forms

For 0-forms, a mapping matrix $\mathbb{T}^{(0)}$ is constructed such that $\mathbb{T}_{i,j} = \psi_j^{(0)}(\xi_i)$, where $\psi^{(0)}$ are 0-form basis defined on the nodes of the merged boundary and ξ_i are positions of the individual degrees of freedom on each of the subdivisions' boundaries. Note that since corner degrees of freedom overlap, only one of them should be considered, with the other being constrained with another Lagrange multiplier, which makes it equal to the other. With the mapping matrix $\mathbb{T}^{(0)}$, the degrees of freedom on the merged boundary can be written as per Equation 6.11, where $\vec{u}_i$ denote degrees of freedom on the boundaries of subdivisions and $\vec{u}_m$ are those on the merged boundary. The representation of this is shown in Figure 6.8.

$$\begin{bmatrix} \vec{u}_0 \\ \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \mathbb{T}^{(0)} \vec{u}_m \tag{6.11}$$

Since the merged boundary was defined to have the same order as the sum of subdivisions' boundaries, this system is inevitable. As such, if now continuity constraints need to be made between the merged boundary with some other element's boundary, the merged boundary degrees of freedom can be instead just written in terms of the degrees of freedom on the subdivisions' boundaries.

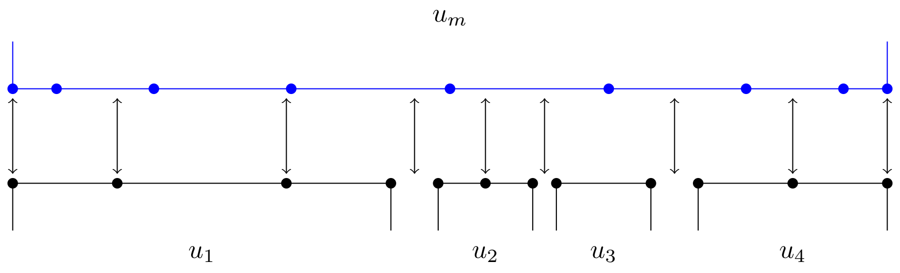


Figure 6.8: Constraint system enforced by Equation 6.11.

For the example in Figure 6.8, the system in Equation 6.11 would be assembled by first assembling the positions of nodes associated with 0-form degrees of freedom for all four elements $\Xi^1 = \left\{\xi_1^1, \xi_2^1, \xi_3^1, \xi_4^1\right\}$, $\Xi^2 = \left\{\xi_1^2, \xi_2^2, \xi_3^2\right\}$, $\Xi^3 = \left\{\xi_1^3, \xi_2^3\right\}$, and $\Xi^4 = \left\{\xi_1^4, \xi_2^4, \xi_3^4\right\}$. From there, these degrees of freedom would be linked to degrees

$$\mathcal{N}_i^1\left(u^1\right) = \mathcal{N}_i^1\left(u^m\right) = u_m\left(\xi_i^1\right) = \sum_{k=1}^{p^m+1} \psi_k^m\left(\xi_i^1\right)\mathcal{N}_k^m\left(u^m\right)$$
$$\mathcal{N}_i^2\left(u^2\right) = \mathcal{N}_i^2\left(u^m\right) = u_m\left(\xi_i^2\right) = \sum_{k=1}^{p^m+1} \psi_k^m\left(\xi_i^2\right)\mathcal{N}_k^m\left(u^m\right)$$
$$\mathcal{N}_i^3\left(u^3\right) = \mathcal{N}_i^3\left(u^m\right) = u_m\left(\xi_i^3\right) = \sum_{k=1}^{p^m+1} \psi_k^m\left(\xi_i^3\right)\mathcal{N}_k^m\left(u^m\right)$$
$$\mathcal{N}_i^4\left(u^4\right) = \mathcal{N}_i^4\left(u^m\right) = u_m\left(\xi_i^4\right) = \sum_{k=1}^{p^m+1} \psi_k^m\left(\xi_i^4\right)\mathcal{N}_k^m\left(u^m\right)$$

To give a numerical example, consider the case where two elements, one with order $p^1 = 1$ and one with order $p^2 = 2$ are merged into the boundary with order $p^m = 3$. This case is illustrated in Figure 6.9. In this case, the system which connects them would be Equation 6.12. Note that the unknown u_1^2 is not present in the system due to the fact that it is determined by the value of u_2^1, thus it would result in an over-constrained system. If the basis in Equation 6.12 are evaluated, the result is Equation 6.13. This system can be solved to obtain expressions for degrees of freedom u_1^m, u_2^m, u_3^m, and u_4^m in terms of u_1^1, u_2^1, u_2^2, and u_3^2.

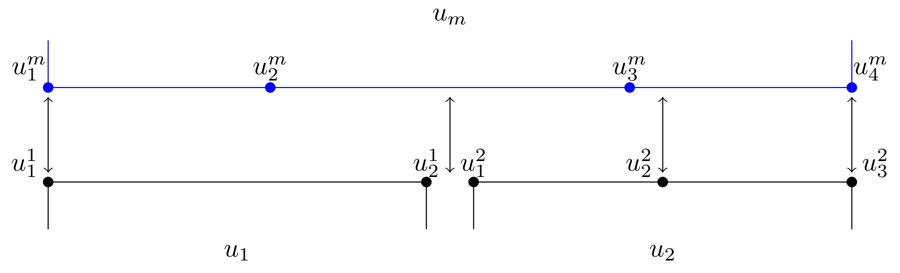


Figure 6.9: Constraint system enforced by Equation 6.11.

$$\begin{bmatrix} u_1^1 \\ u_2^1 \\ u_2^2 \\ u_3^2 \end{bmatrix} = \begin{bmatrix} \mathcal{N}_1^1\left(u^m\right) \\ \mathcal{N}_2^1\left(u^m\right) \\ \mathcal{N}_2^2\left(u^m\right) \\ \mathcal{N}_3^2\left(u^m\right) \end{bmatrix} = \begin{bmatrix} \psi_1^m\left(\xi_1^1\right) & \psi_2^m\left(\xi_1^1\right) & \psi_3^m\left(\xi_1^1\right) & \psi_4^m\left(\xi_1^1\right) \\ \psi_1^m\left(\xi_2^1\right) & \psi_2^m\left(\xi_2^1\right) & \psi_3^m\left(\xi_2^1\right) & \psi_4^m\left(\xi_2^1\right) \\ \psi_1^m\left(\xi_2^2\right) & \psi_2^m\left(\xi_2^2\right) & \psi_3^m\left(\xi_2^2\right) & \psi_4^m\left(\xi_2^2\right) \\ \psi_1^m\left(\xi_3^2\right) & \psi_2^m\left(\xi_3^2\right) & \psi_3^m\left(\xi_3^2\right) & \psi_4^m\left(\xi_3^2\right) \end{bmatrix} \begin{bmatrix} u_1^m \\ u_2^m \\ u_3^m \\ u_4^m \end{bmatrix} \tag{6.12}$$

$$\begin{bmatrix} u_1^1 \\ u_2^1 \\ u_2^2 \\ u_3^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.125 & 0.625 & 0.625 & -0.125 \\ 0.015625 & -0.05532843 & 0.99282843 & 0.046875 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{N}_1^m\left(u^m\right) \\ \mathcal{N}_2^m\left(u^m\right) \\ \mathcal{N}_3^m\left(u^m\right) \\ \mathcal{N}_4^m\left(u^m\right) \end{bmatrix} \tag{6.13}$$

6.2.3 Continuity of 1-forms

For 1-forms, the situation is similar, but without the hassle of having to exclude duplicate degrees of freedom and then having to add Lagrange multipliers to make them equal. The only real difference from the process for 0-forms is that the entries of the mapping matrix instead become $\mathbb{T}^{(1)}_{i,j} = \sum_{k=1}^{j} \psi^{(0)}_k(\xi_i) - \psi^{(0)}_k(\xi_{i+1})$, as per Equation 6.9. A graphical representation of how the boundaries are connected is shown in Figure 6.10.

After the square system of equations relating the merged boundary degrees of freedom to those of the subdivisions, as shown in Equation 6.14, these can once again be substituted into any continuity relations in order to eliminate the merged boundary degrees of freedom.

$$\begin{bmatrix} \vec{u}_0 \\ \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \mathbb{T}^{(1)} \vec{u}_m \tag{6.14}$$

Following this process, connecting a subdivided boundary with a non-subdivided one, like with the example from Figure 6.6 can be done for an arbitrary number of subdivisions and arbitrary orders of these subdivisions. The system here is assembled in the same way as it is done for 0-forms, except without the need to worry about duplicated boundary degrees of freedom, as in two dimensions, each edge may only be in exactly two elements.

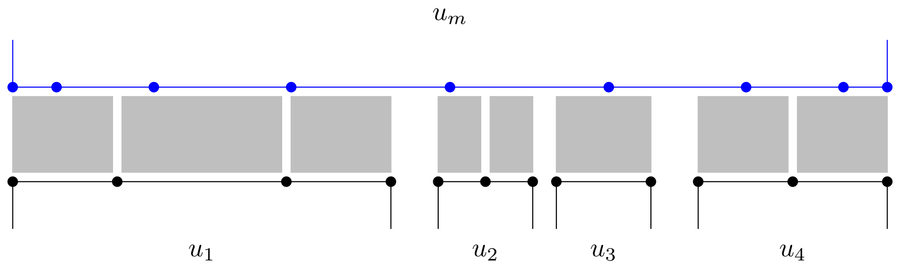


Figure 6.10: Constraint system enforced by Equation 6.14.

6.3 Combined *hp*-refinement

With all the relations and procedures for continuity derived for p-refinement in section 6.1 and h-refinement in section 6.2 a systematic process for creating constraints for connecting all the individual (bottom level) elements can be created. The prerequisites for this is first having the element hierarchy, so that it can be determined which are parent and which are child elements of what element. Another of the prerequisites is having both the primal mesh topology and the dual topology, though the dual can be computed from the primal.

Continuity constraints can be divided based on their origin into inter-element and intra-element constraints, meaning they either originate from continuity between root (top-level) elements or from subdivisions that make up the elements. The second division can be made based on where they are applied on the boundary - along the interior of the edge, which is the case for 1-forms and inner 0-form degrees of freedom, or at the corners of the edge, which is the case for the first and last degrees of freedom of 0-forms on that edge.

Connecting edges together can be done with the following steps, with an example it could be used on in Figure 6.11 and Figure 6.12:

1. Check if either of the elements have child elements on their respective boundaries.
2. If both have children (as in first level of Figure 6.11):
   (a) Recursively perform the edge connection procedure on the pairs of children that border each other (bottom part of Figure 6.11).
   (b) Connect the corner degrees of freedom for 0-forms that are shared between the all four children.
3. Otherwise:
   (a) If only one has children instead and the k-form is a 0-form connect the degrees of freedom on the corners of subdivisions along the boundary (as is the case in Figure 6.12).

(b) Find the highest order of the boundaries of the two elements.
(c) Obtain degrees of freedom for merged boundary if one is subdivided.
(d) Connect the two boundaries together based on the *p*-refinement method.

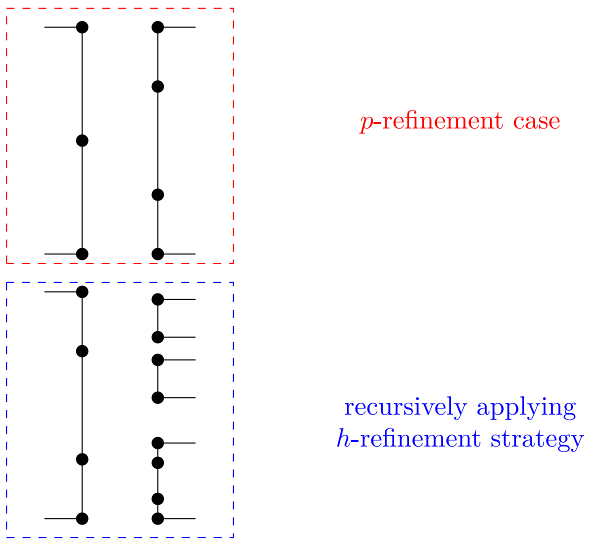


Figure 6.11: Example of how the connectivity procedure is applied to a boundary where both sides have different levels of refinement.

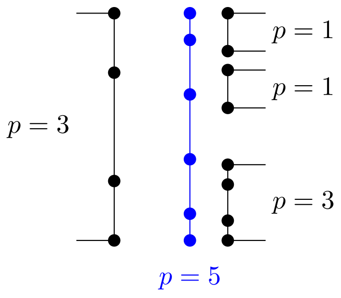


Figure 6.12: Example of connecting the boundaries on the bottom half of the example from Figure 6.11, after the connectivity process is recursively applied.

With this method for connecting edges, all the continuity constraints can be generated with the following steps:

1. For each surface in the mesh, obtain the root element and generate the intra-element constraints:
   (a) Connect edges of the child elements.
   (b) Connect the corner between the four of the child elements.
   (c) Recursivly perform this for each of the child elements if they have their own children.
2. For each dual line in the dual mesh connect the edges of the two root elements (corresponding to surfaces that are dual to the beginning and end points of the dual line).
3. For each dual surface in the dual mesh connect the corners of elements around the point the dual surface is dual of.

6.4 Refinement Criterion

Now, a very important question to answer is how exactly should the refinement be done. As mentioned in the literature study in chapter 2, there are quite a few ways to do this, varying in both complexity, cost, and usefulness. For this work, the first and foremost aim is to explore how VMS can be used as a criterion and as such performance will not be considered until later.

First, a very good summary of methods developed by 2003 is given by Houston et al. [19], who also themselves created a refinement criterion. The ones they present in their work were:

- Using the information known about the solution in advance. That is mainly used for the cases where the problem involves some material properties or coefficients, which either have very large local variations or which are chosen to be represented with discontinuous functions. In such cases using h-refinement near such discontinuities or variations while using p-refinement elsewhere yields good results.
- Type parameter, defined as a function of a local solution, local polynomial order, and some measure of the element's size can be used. For this case, the type parameter is computed for the current mesh and solution and then for the same problem, but solved on a mesh one order finer/coarser. From there, a decision about whether to take h-refinement or p-refinement is taken based on the ratio of the fine and the coarse type parameter value.
- The "Texas 3-step" method can also be used, where the decision between the h-refinement and p-refinement is very straight forward: after target accuracy for the other two steps and some error estimator are defined in the step one, second step involves appliying h-refinement until first error tolerance is achieved, followed by the third step which uses p-refinement until second error tolerance is met.
- Mesh optimization can be performed for a reference known solution. This means that some other solution, which is hopefully somewhat similar to the original problem's solution, projection error is computed for both the h-refinement and p-refinement of the element. From there, the one which would result in the greatest reduction in projection error for the lowest number of newly introduced degrees of freedom is used. This is related to work on error estimator work by Demkowicz et al. [16], which will be discussed again under error estimators in section 6.5.
- The rate of exponential decay fit to Legendre coefficient can also be used. In this situation, the expansion coefficients α_i of the solution with Legendre basis are used to fit an exponentially decaying function $\alpha_i \approx Ce^{-\gamma i}$ with two parameters: initial magnitude C and the rate of decay γ. These are typically determined using a least-squares fit. From the value of the rate of decay γ, the sharpness of solution can be estimated, and thus the decision between h-refinement and p-refinement can be reached.
- Another method is based on the fact that the maximum rate of convergence for a FEM method can not exceed the order of the Sobolev space the solution is in. As such, a local regularity estimation is performed for each of the elements in the mesh. Based on this information, p-refinement is used until the order of the element is the same as the regularity of the Sobolev space. Otherwise, h-refinement is used once the order is already high enough.

In their work, Houston et al. [19] introduced another method for estimating the local regularity of the solution, based on the last two methods presented - the Legendre coefficient expansion and the local Sobolev index estimation.

A common theme among almost all of these methods is the fact that they do not rely on an error estimate and instead only rely on the solution, with some also incorporating some properties of the mesh. While that is undoubtedly also a feasible approach, these can not take advantage of VMS, which provides what is appears to provide a very good error estimate. As such, a different error estimate will be derived, this time based on the assumption that the perfect error estimate is available for it.

First, to come up with a refinement criterion, consider the ideal scenario - perfect knowledge of error! Next, suppose we do not just immediately use that to get the perfect solution. If the error were perfectly known, what are the available options? For each element individually, there are two options for improving local accuracy when changing its order: divide the element first or leave it undivided.

Given that for 2-forms division of an element with order $2n$ into four elements with order n retains the number of degrees of freedom (for 0-forms and 1-forms it is true for the case of $n \to \infty$), an element of order $2n$ can be considered about equivalent to an $2n$ element in terms of number of degrees of freedom. However, this does not actually mean that the same accuracy would be achieved.

To illustrate this, consider a sample problem of linear advection-diffusion in the mixed formulation given by Equation 6.15 and Equation 6.16. For this example, the problem was solved using a single element for different polynomial orders. First as is, then followed by another time with one step of h-refinement. The domain was $\Omega := (x, y) \in [-1, -1] \times [+1, +1]$. The advection field was chosen as $\vec{a} = (2x - y)\frac{\partial}{\partial x} + (3y + x^2)\frac{\partial}{\partial y}$.

$$-\left(p^{(1)}, q^{(1)}\right)_\Omega + \left(\mathrm{d}p^{(1)}, u^{(2)}\right)_\Omega = \int_\Omega p^{(1)} \wedge \star u^{(2)} \quad \forall p^{(1)} \in \Lambda^{(1)}(\Omega) \tag{6.15}$$

$$\left(v^{(2)}, \mathrm{d}q^{(1)}\right) + \left(i_{\vec{a}} v^{(2)}, q^{(1)}\right)_\Omega = \left(v^{(2)}, f^{(2)}\right)_\Omega \quad \forall v^{(2)} \in \Lambda^{(2)}(\Omega) \tag{6.16}$$

The manufactured solution was given by Equation 6.17, where the function g is given by Equation 6.18. This parametrization was done to check what happens when the solution (with infinite Taylor expansion) has different positions and sharpness.

$$u(x, y; r, x_0, y_0) = g(x; r, x_0) \cdot g(y; r, y_0) \tag{6.17}$$

$$g(t; r, t_0) = e^{-r\cdot(t-t_0)^2} \tag{6.18}$$

6.4.1 Effect of Sharpness

The first thing that was investigated was the effect of solution sharpness. This means having the value of x_0 and y_0 fixed, while the value of r was varied. The values of $x_0 = 0$ and $y_0 = 0$ were chosen and the problem was repeated for different values of $r = 0.01$, $r = 0.1$, $r = 1$, and $r = 10$. This would result in a cross section as can be seen in Figure 6.13. For these values, the first two values both represent very smooth solutions, while the last one is quite sharp, considering that only a single element is used for solution.

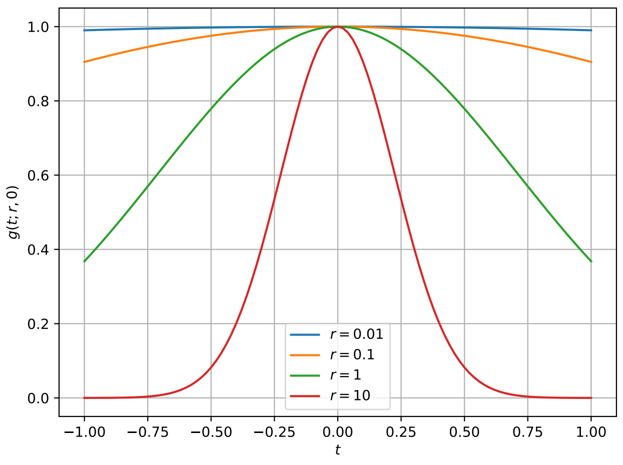
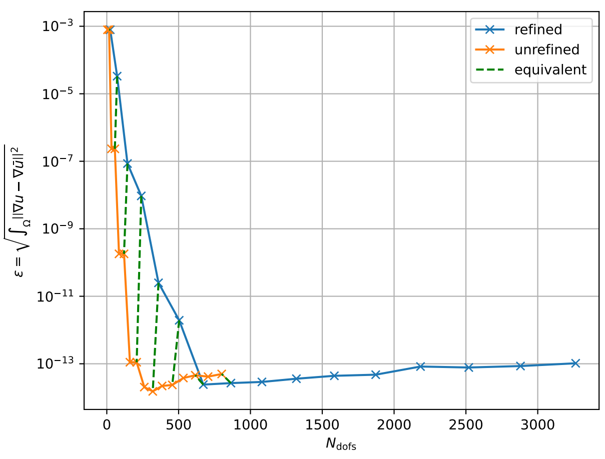


Figure 6.13: Solution cross section for varying values of r

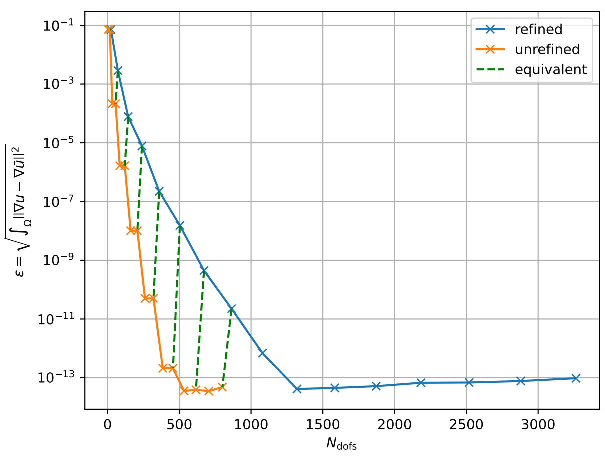
The convergence behavior changes greatly between these, as can be seen in Figure 6.14. There, as well as in subsequent plots in this section, the lines marked as "unrefined" denote results obtained on the baseline mesh, while the "refined" lines denote the results obtained once that mesh was h-refined once. Note that there the green dashed lines connect results of unrefined element of order $2n$ with the split elements with order n. What can be seen is that as the value of r increases, and the solution becomes more and more sharp, the difference between these cases starts to drop.

To compare the two most extreme cases, consider first the case of $r = 0.01$ in Figure 6.14a. The error increases by two or for decades if instead of a single element with order $2n$ four order n elements are used. In a situation like this, there is no real point in performing h-refinement, since p-refinement is just so much more efficient. On the other hand, for $r = 10$, which is shown in Figure 6.14d, the solution is so sharp, that error for a single element of order $2n$ is equal to or sometimes greater than the error for four elements of order n.

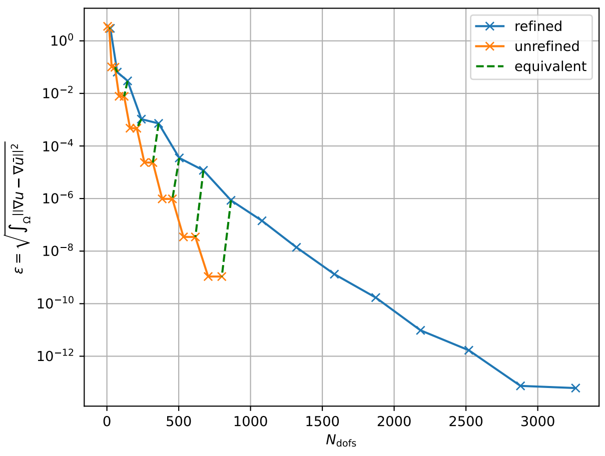
For other two intermediate values of r in Figure 6.14b and Figure 6.14c, the situation is as expected somewhere between the two extremes, showing that as the solution becomes smoother, p-refinement is more and more attractive, as if h-refinement is applied, the only way to not loose too much accuracy, it can not simply be done with four order n elements, but order $n+2$ or even $n+3$ order elements, which increases the number of degrees of freedom significantly.



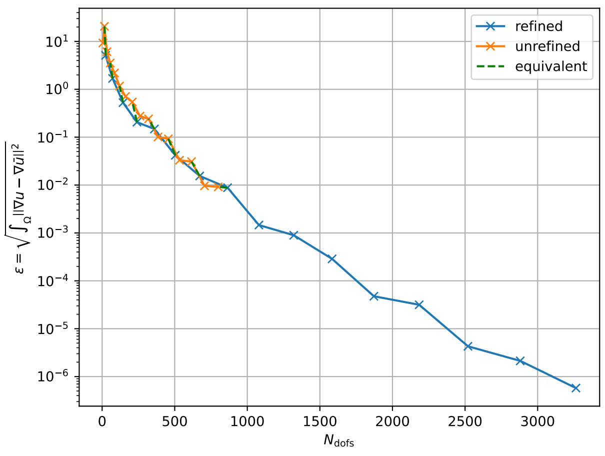
(a) Convergence in the H^1 norm for the case of $r = 0.01$



(b) Convergence in the H^1 norm for the case of $r = 0.1$



(c) Convergence in the H^1 norm for the case of $r = 1$



(d) Convergence in the H^1 norm for the case of $r = 10$

Figure 6.14: Convergence behaviour of the solution for varying values of r

6.4.2 Effect of Location

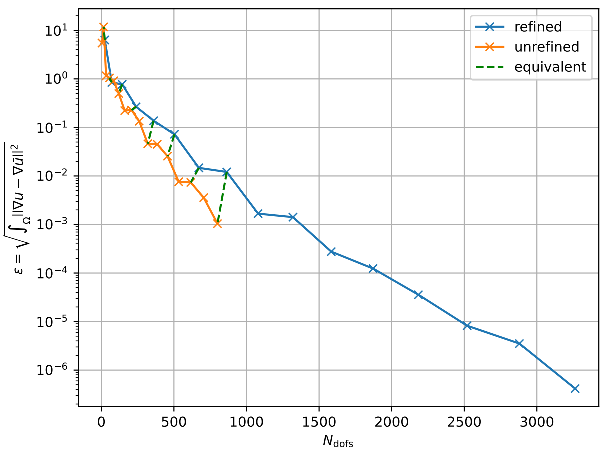
For a sharp solution, the effect of the spatial distribution was also examined. Since the element can be split into four sub-elements, the following cases were checked:

- Peak in one quadrant,
- peaks in two diagonal quadrants,
- peaks in two neighboring quadrants,
- peaks in three out of four quadrants.

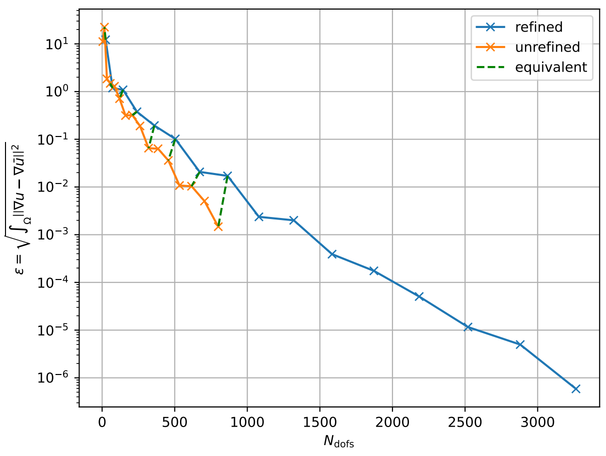
The main purpose of this was to check how the behaviour would differ for the case where the sharp solution is centered, which is what was checked in subsection 6.4.1. As such, the same value of $r = 10$ was used. To add peak to the solution, they would be placed at $(\pm 0.75, \pm 0.75)$, depending on what quadrant they should be in. Since this problem is linear advection-diffusion, the manufactured solutions were made by simply adding together that defined by Equation 6.17, with these values of peak positions.

The results of these can be seen in Figure 6.15. The first thing to notice is that all of these perform better than the centered peak, which was shown in Figure 6.14a. This can likely be attributed to the fact that in that case, no part of the peak was outside of the solution, while here some parts of the peak might be outside the domain. As for the comparison between these different cases, the first thing that can be seen is that as far as a *single* refinement step, these all perform about the same in terms of error behaviour.

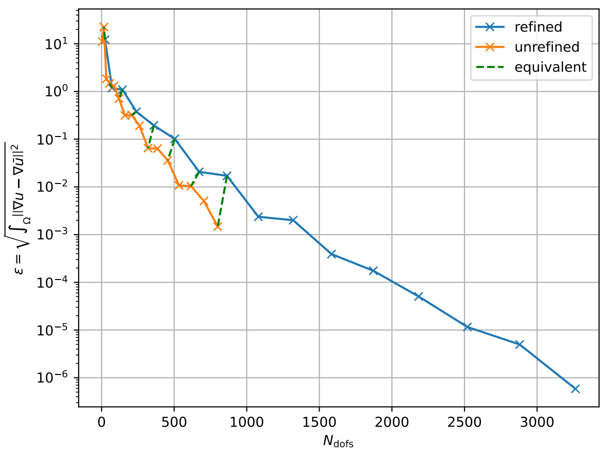
As such, the location of the error seems less important than the sharpness, which caused dramatic differences in the behavior of the error convergence. There is still one important note that while the case of a peak centered on the middle of the element had a single element of order $2n$ be almost exactly equivalent to four n order elements, the cases where the peak is not centered require the four elements to be about order $n+1$, thus adding a few more degrees of freedom.



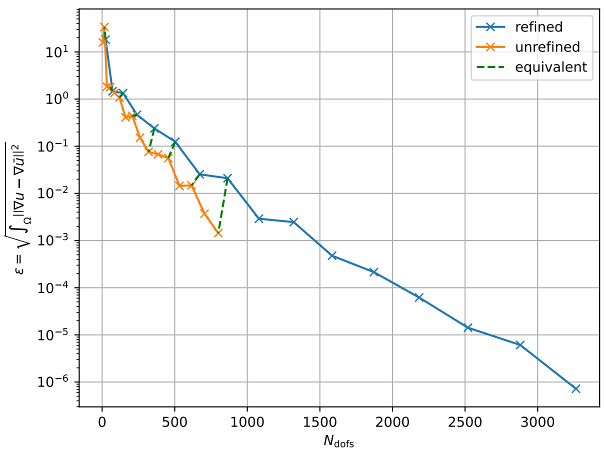
(a) Convergence in the H^1 norm for the case of a single corner peak



(b) Convergence in the H^1 norm for the case with neighboring two peaks



(c) Convergence in the H^1 norm for the case with diagonal peaks



(d) Convergence in the H^1 norm for the case with three peaks

Figure 6.15: Convergence behaviour of the solution for varying number and position of peaks.

6.4.3 Choosing the Type of Refinement

An important thing to check if h-refinement is really worth it is see how much the error increases due the resulting order drop. Consider first a simple 1D case where the exact solution is written in terms of Legendre basis as $u = \sum_{i=0}^{\infty} u_i P_i$. In this case, the lowest L^2 error case is the computed solution of order q matches the coefficients of u exactly up to its order, meaning it is given as $u_q = \sum_{i=0}^{q} u_i P_i$. If the polynomial order would be decreased to $p \leq q$ (since h-refinement with equivalent number of degrees of freedom would need $p \approx \frac{q}{2}$), the highest that could be resolved would be $u_p = \sum_{i=0}^{p} u_i P_i$. As such, the error increase between these two would be the difference of the two, given by $u_q - u_p = \sum_{i=p+1}^{q} u_i P_i$. From there, the increase in the L^2 error norm would in this case be given by Equation 6.19.

$$||u - u_p||_{L^2}{}^2 - ||u - u_q||_{L^2}{}^2 = \frac{1}{2}\int_{\Omega} (u - u_p)^2 - (u - u_q)^2 \, dx = \frac{\Delta x}{4} \underbrace{\sum_{i=p+1}^{q} \frac{2{u_i}^2}{2i+1}}_{\text{change of order error}} \tag{6.19}$$

However, this is assuming that the computed solutions on the q order element and p order element (u_q and u_p respectively) are given by their exact L^2 projections, since their Legendre expansion coefficients match the expansion of the exact solution u. For any case of interest, this is only close to being true at the point where p is high enough to obtain the solution to machine precision, so much too late. As such, the error increase will not quite follow from Equation 6.19.

Now consider the if instead there is an error in the solution u_q, it is given as $e_q = u - u_q = \sum_{i=0}^{q} {e^q}_i P_i + \sum_{i=q+1}^{\infty} u_i P_i$. The error in the solution u_p is then given by $e_p = u - u_p = \sum_{i=0}^{p} {e^p}_i P_i + \sum_{i=p+1}^{\infty} u_i P_i$. As such, the real change in the L^2 error's square is given by Equation 6.20, which is a modification of Equation 6.19. From here, the error now has two sources - first one from the difference in the error of the two cases due to difference in convergence rate and the second one from the change of order errors.

$$||e_p||_{L^2} - ||e_q||_{L^2} = \frac{\Delta x}{4}\left(\underbrace{\sum_{i=0}^{p} \frac{2}{2i+1}\left(({e^p}_i)^2 - ({e^q}_i)^2 \right)}_{\text{convergence rate error}} + \underbrace{\sum_{i=p+1}^{q} \frac{2}{2i+1}\left({u_i}^2 - ({e^q}_i)^2 \right)}_{\text{change of order error}} \right) \tag{6.20}$$

Assuming these calculations are done on the element with the order q, then error term due to change of order can be found if coefficients for ${e^q}_i$ and u_i, or at least their estimates are known for $g \in [p+1, q]$. Convergence rate errors are harder to estimate if the solution (and error) are not already computed at order p. If it is assumed that the convergence rate error is small compared to change of order error, an estimate for the change in the L^2 error can be given as Equation 6.21, when the definition $u = u^q + e^q$ is used.

$$\Delta\varepsilon_{L^2} \approx \frac{\Delta x}{4} \sum_{i=p+1}^{q} \frac{2}{2i+1}\left({u_i}^2 - {e^q_i}^2 \right) = \frac{\Delta x}{4} \sum_{i=p+1}^{q} \frac{2{u^q}_i \left({u^q}_i + 2{e^q}_i \right)}{2i+1} \tag{6.21}$$

From there, the element can be refined according to the rule that if $\Delta\varepsilon_{L^2} \leq \nu_h \varepsilon_{L^2}$, where ν_h is a global hyper-parameter and ε_{L^2} is the element L^2 error, then h-refinement is performed in that refinement step. If the condition does not hold, p-refinement is to be performed instead. Simply observing the approximation of the criterion as given by Equation 6.21, it is clear that the cases where h-refinement would occur is either where high coefficients ${u^q}_i$ are not important, while the error in those high orders is not too large. What can also be seen is that in absence of any error (meaning that $e^q = 0$ and ${u^q}_i = u_i$), the criterion reverts to Equation 6.19.

There is something to be said about applicability of this criterion. Namely, the fact that it does not take into account the fact that error on an an element with polynomial order p is higher than it would be on an element following h-refinement, since it would be split into four smaller elements. As such, if the convergence rate error is indeed small, it may happen that actually the estimate given by Equation 6.21 is higher than the real value.

Extension to Higher Dimensions

The criterion can be easily extended to higher dimensions when elements are parallelotope, which can then be covered with basis resulting from tensor products. From there the criterion is given by Equation 6.22, where m is the number of dimensions, V is the "volume" of the element, given by integrating over it, with $p_i \leq q_i$ being the orders of the coarser and finer element in the dimension i. For a case in two dimensions, Equation 6.22 becomes Equation 6.23.

$$\Delta\varepsilon_{L^2} = \frac{V}{2^{m+1}} \left(\sum_{i_1=0}^{p_1} \cdots \sum_{i_m=0}^{p_m} 2^m \frac{\left(e_{i_1,\ldots,i_m}^{p_1,\ldots,p_m}\right)^2 - \left(e_{i_1,\ldots,i_m}^{q_1,\ldots,q_m}\right)^2}{\prod_{k=1}^{m} 2i_k + 1} + \sum_{i_1=p_1+1}^{q_1} \cdots \sum_{i_m=p_m+1}^{q_m} 2^m \frac{\left(u_{i_1,\ldots,i_m}\right)^2 - \left(e_{i_1,\ldots,i_m}^{q_1,\ldots,q_m}\right)^2}{\prod_{k=1}^{m} 2i_k + 1} \right) \tag{6.22}$$

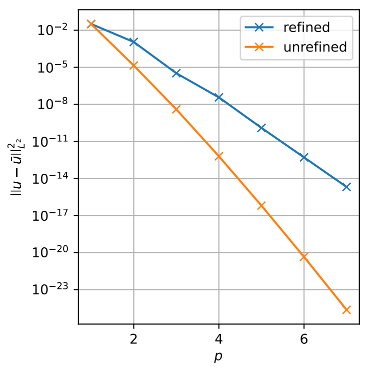
$$\Delta\varepsilon_{L^2} = \frac{\int_\Omega dxdy}{8} \left(\sum_{i=0}^{p_1} \sum_{j=0}^{p_2} 4 \frac{\left(e_{i,j}^{p_1,p_2}\right)^2 - \left(e_{i,j}^{q_1,q_2}\right)^2}{(2i+1)(2j+1)} + \sum_{i=p_1+1}^{q_1} \sum_{j=p_2+1}^{q_2} 4 \frac{\left(u_{i,j}\right)^2 - \left(e_{i,j}^{q_1,q_2}\right)^2}{(2i+1)(2j+1)} \right) \tag{6.23}$$

6.4.4 Criterion Verification

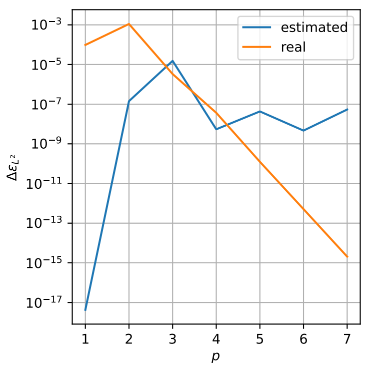
The criterion derived in subsection 6.4.3 should at the very least perform well with the cases that were tried in subsection 6.4.1 and subsection 6.4.1. The results for the case of centered peak were done for values of sharpness given by $r = \{0.20, 0.30, 0.50, 0.75, 1.00, 2.00, 4.00\}$. These were then repeated for the non-centered peak located at $(x, y) = (0.75, 0.75)$. Since the results in subsection 6.4.2 suggested that the number of peaks does not matter, it was assumed that doing this one test would be sufficient to assess suitability of this criterion.

Convergence plots for the square of the error in the L^2 norm are shown in Figure 6.16 for the case of $r = 0.20$, Figure 6.17 for the case of $r = 1.00$, and Figure 6.18 for $r = 4.00$. In these the line marked as "refined" is the results obtained with h-refined element which has the order p, while the "unrefined" are the results with a single unrefined element of order $2p$. The first thing that can be seen looking at the convergence plots in Figure 6.16a, Figure 6.17a, and Figure 6.18a, that the convergence of an h-refined element with equivalent number of degrees of freedom approaches that of the non-refined element of order $2p$ as the sharpness r increases. This was already shown in subsection 6.4.1 , so this acts more as a validation.

The important results are the performance of error estimates shown in Figure 6.16b, Figure 6.17b, and Figure 6.18b. These contain plots of the actual difference in the L^2 error's square between the "refined" and "unrefined" results for each case, along with their estimates, which were computed using the 2D version of Equation 6.21. What can be seen is that these estimates are terrible for $p = 1$, which is understandable, since the solution is beginning to converge and the neglected terms from the estimate are likely dominant. Following $p = 3$, the estimate is either of about the same order of magnitude or larger.



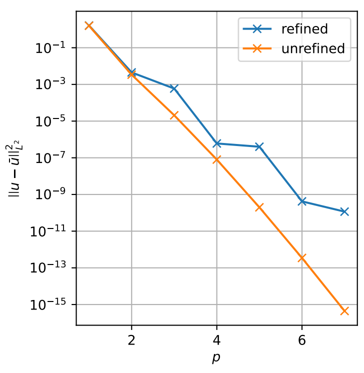
(a) Convergence of square of L^2 error for the case with $r = 0.20$



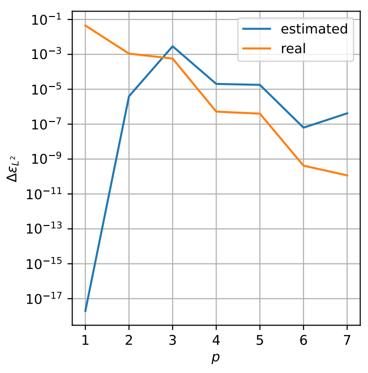
(b) Estimate of change of square of L^2 error for the case with $r = 0.20$

Figure 6.16: Convergence of the square of L^2 error and performance of error estimate for $r = 0.20$.

Next, the case with the moved peak was also examined for the same values of sharpness r. The results for the cases where $r = 0.20$, $r = 1.00$, and $r = 4.00$ are shown in Figure 6.19, Figure 6.20, and Figure 6.21 respectively.

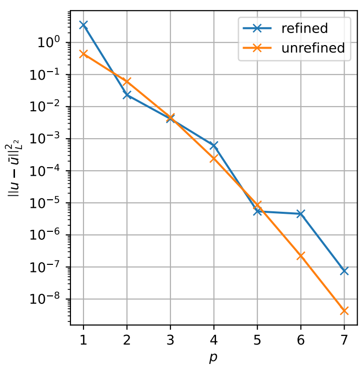


(a) Convergence of square of L^2 error for the case with $r = 1.00$

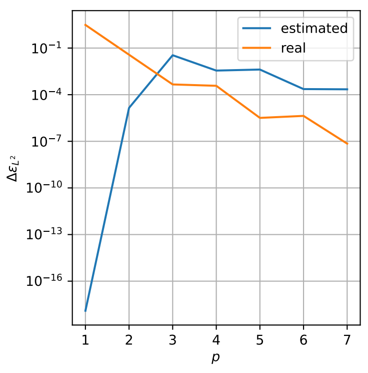


(b) Estimate of change of square of L^2 error for the case with $r = 1.00$

Figure 6.17: Convergence of the square of L^2 error and performance of error estimate for $r = 1.00$.



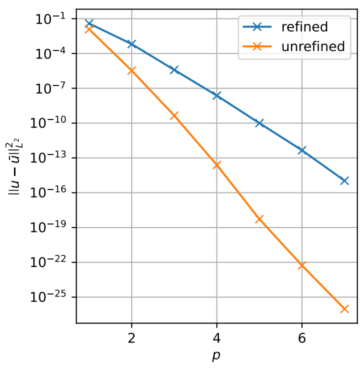
(a) Convergence of square of L^2 error for the case with $r = 4.00$



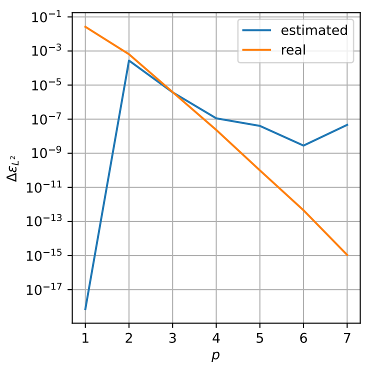
(b) Estimate of change of square of L^2 error for the case with $r = 4.00$

Figure 6.18: Convergence of the square of L^2 error and performance of error estimate for $r = 4.00$.

While the convergence plots show about the same behavior to those for the centered peaks, the behavior of the error estimates in Figure 6.19b, Figure 6.20b, and Figure 6.21b is much more regular.
Additionally, the criterion was applied to a case set up with the solution given as $u = \sin(\omega x)\sin(\omega y)$ for different values of $\omega = \{0.20\pi, 0.30\pi, 0.50\pi, 0.75\pi, 1.00\pi, 2.00\pi, 4.00\pi\}$. This was to check that the criterion should also work well for the cases where the solution is globally smooth or sharp. The results show a similar trend of convergence, though it should be noted that the sine function with $\omega = 1.00\pi$ will be significantly more sharp than a Gaussian with the parameter $r = 1$. Still, the behavior for the convergence is about what is about what could be expected based on previous results. One significant difference is regarding the results for convergence of $\omega = 4.00\pi$ in Figure 6.24a. Since that solution is extremely sharp, the convergence does not behave regularly at all. The behavior shown by the estimate of the error change again shows similar behavior for sharper solutions, such as the case of $\omega = 4.00\pi$ in Figure 6.24b, which then degrades and overestimates it as the sharpness declines, which can be seen for $\omega = 0.20\pi$ in Figure 6.22b.

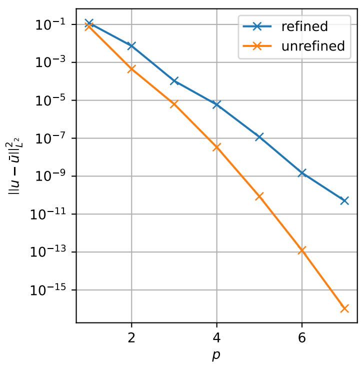


(a) Convergence of square of L^2 error for the case with $r = 0.20$

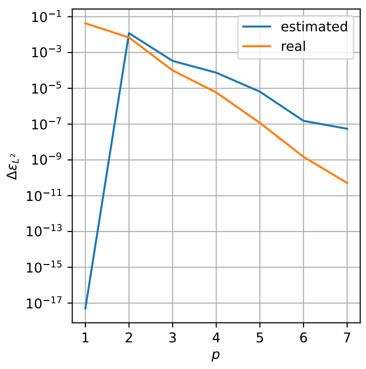


(b) Estimate of change of square of L^2 error for the case with $r = 0.20$

Figure 6.19: Convergence of the square of L^2 error and performance of error estimate for $r = 0.20$.

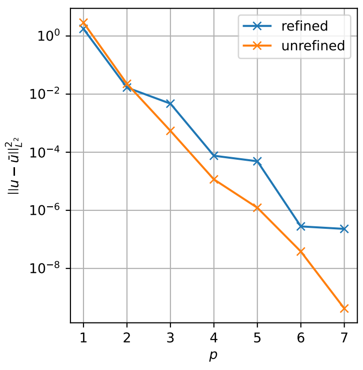


(a) Convergence of square of L^2 error for the case with $r = 1.00$

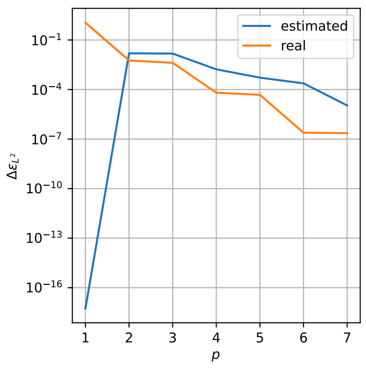


(b) Estimate of change of square of L^2 error for the case with $r = 1.00$

Figure 6.20: Convergence of the square of L^2 error and performance of error estimate for $r = 1.00$.

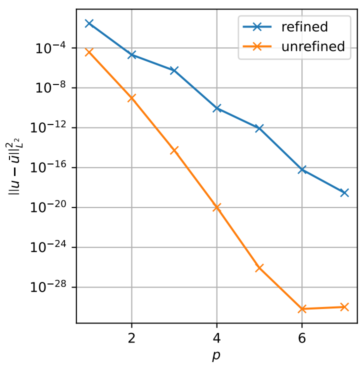


(a) Convergence of square of L^2 error for the case with $r = 4.00$

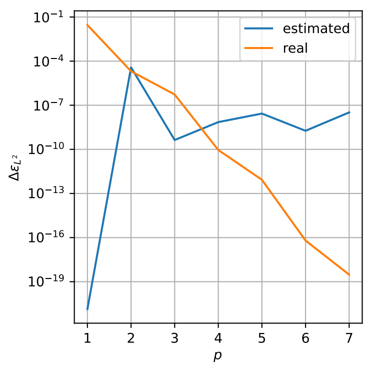


(b) Estimate of change of square of L^2 error for the case with $r = 4.00$

Figure 6.21: Convergence of the square of L^2 error and performance of error estimate for $r = 4.00$.

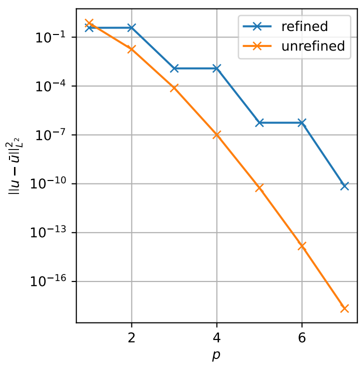


(a) Convergence of square of L^2 error for the case with $\omega = 0.20$

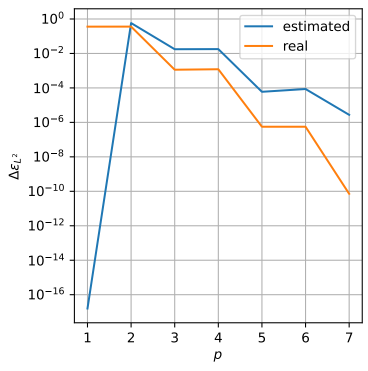


(b) Estimate of change of square of L^2 error for the case with $\omega = 0.20$

Figure 6.22: Convergence of the square of L^2 error and performance of error estimate for $\omega = 0.20$.

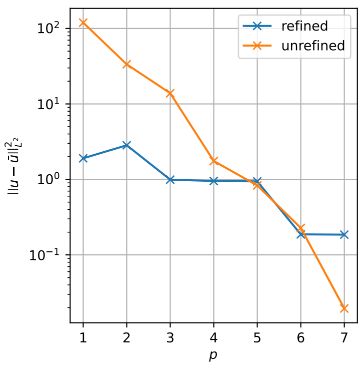


(a) Convergence of square of L^2 error for the case with $\omega = 1.00$

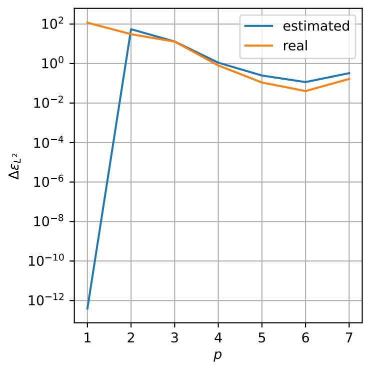


(b) Estimate of change of square of L^2 error for the case with $\omega = 1.00$

Figure 6.23: Convergence of the square of L^2 error and performance of error estimate for $\omega = 1.00$.



(a) Convergence of square of L^2 error for the case with $\omega = 4.00$



(b) Estimate of change of square of L^2 error for the case with $\omega = 4.00$

Figure 6.24: Convergence of the square of L^2 error and performance of error estimate for $\omega = 4.00$.

6.5 Estimating Error

The refinement criterion's most important input is the error in the solution. As such, it is important to have a good way to estimate it. For benchmarks, the baseline can be obtained using the manufactured solution, for which the error is exactly known. However, this is not something that can be done in practice, since the exact solution is not known for any practical problem of interest.

There are a few ways to try and overcome this issue, which are discussed in this section. These each have inherent design advantages and drawbacks, which are compared at the end of the section. Some of these are also among ones detailed

6.5.1 Using the Exact Solution

Using the exact solution u allows for exact error e to be obtained from the computed solution $\bar{u}$, by simply subtracting the two $e = u(x, y) - \bar{u}$. The most glaring problem of using this is that it is not something that can be done for any problem of practical application or interest outside of test cases, for which manufactured solutions.

This error "estimator" is thus presented as a benchmark, since a better error estimate than the one obtained from this method can not be obtained. As such, by comparing the performance of the other estimators to this, their effectiveness can be judged more easily.

6.5.2 Using a Higher Order Solution

Since the exact solution is only available for test problems, a more practical approach is to use an estimate which is significantly better than the current solution, such that the difference between the two is mainly due to current solution's error, rather than that of the higher order solution. This method is in part inspired by work by Demkowicz et al. [18], who employed this method not only for error estimation, but also as the deciding criterion between h-refinement and p-refinement.

A very simple way to do this is to solve the problem on a finer mesh, which is uniformly refined. As an example, the order of each of the elements can be increased by $\Delta p \geq 1$. This does add quite a notable cost, as now not only does the problem need to be solved twice, but the second solve on the finer mesh is significantly more expensive.

One could also look at this the other way around, which is that error estimation and refinement is done based on a mesh which is coarser, with each element being an order Δp lower. In practice, the outcomes of this technique is the same, if the final results used are taken from the higher order solution.

Mathematically, this can be formulated the way it is in Equation 6.24, where L^2 Hermitian adjoint projection is used is used to project the coarse solution $\overline{u}$ to the fine space to take the difference between it and $\widetilde{u}$. Since the coarse space is fully contained within the fine space, the coarse solution $\overline{u} = \overline{L}^{-1}(\overline{f})$ can be rewritten as projection of the fine solution $\widetilde{u}$ through the differential operator $\mathcal{L}$. While it can be written like this, it is not practical for computing it, since $\widetilde{f} = \widetilde{L}(\widetilde{u})$ is already known, so for practical use, the forcing $\widetilde{f}$ should just be L^2 projected to the coarse mesh, where the coarse solution $\overline{u}$ is computed, then L^2 projected back to the fine mesh.

$$e \approx \widetilde{u} - \widehat{\mathcal{P}}_{L^2}\overline{u} = \widetilde{u} - \widehat{\mathcal{P}}_{L^2}\mathcal{L}^{-1}\mathcal{P}^T_{L^2}\widetilde{\mathcal{L}}\widetilde{u} = \left(1 - \widehat{\mathcal{P}}_{L^2}\mathcal{P}_{\mathcal{L}}\right)\widetilde{u} \tag{6.24}$$

6.5.3 Higher Order Bubble Functions

Another way to perform error estimation is using higher order bubble functions. This method promises much greater parallelization at the trade-off of accuracy. The underlying problem being solved is that if the residual given by Equation 6.25 is known, the (linearized) inverse of $\mathcal{L}$ could be used to obtain error from it.

$$r = f - \mathcal{L}(\overline{u}) \tag{6.25}$$

Since the residual is zero for the original space in which $\overline{u}$ is computed, it must be computed in a higher space. As such, the residual should be computed in a higher order space. This results in Equation 6.26. When using the fact that the coarse space is fully contained in the fine space, it can be further rewritten into Equation 6.27. When the error is computed from Equation 6.27 by applying the inverse of the differential operator, Equation 6.28 is recovered, which can be rewritten to become exactly the relation used by the method in subsection 6.5.2.

$$\widetilde{r} = \widetilde{f} - \widetilde{\mathcal{L}}\widehat{\mathcal{P}}_{L^2}\overline{u} \tag{6.26}$$

$$\widetilde{r} = \widetilde{f} - \widetilde{\mathcal{L}}\widehat{\mathcal{P}}_{L^2}\overline{\mathcal{L}}^{-1}\mathcal{P}_{L^2}\widetilde{f} \tag{6.27}$$

$$e \approx \widetilde{e} = \widetilde{\mathcal{L}}^{-1}\widetilde{r} = \widetilde{u} - \widehat{\mathcal{P}}_{L^2}\overline{\mathcal{L}}^{-1}\mathcal{P}_{L^2}\widetilde{\mathcal{L}}\widetilde{u} \tag{6.28}$$

Taking a step back, consider the relation Equation 6.26. If instead of expressing quantities in terms of $\widetilde{u}$ and applying the full proper inverse $\mathcal{L}^{-1}$, an approximate inverse $\mathcal{B}^{-1}$ is applied, the expression for the approximation of the error is given by Equation 6.29. If $\widetilde{\mathcal{B}}^{-1}$ can be applied locally, on per-element basis, then the entire expression can be computed entirely in parallel, once the coarse solution $\overline{u}$ is known.

$$e \approx \widetilde{\mathcal{B}}^{-1}\widetilde{r} = \widetilde{\mathcal{B}}^{-1}\left(\widetilde{f} - \widetilde{\mathcal{L}}\widehat{\mathcal{P}}_{L^2}\overline{u}\right) \tag{6.29}$$

From Equation 6.28 it was already shown that if no approximation is taken and if $\widetilde{\mathcal{B}}^{-1} = \widetilde{\mathcal{L}}^{-1}$, this refinement criterion reduces exactly to using the difference between the finer and coarser solution, just as it was is subsection 6.5.2, so any approximation made from there, can be expected to yield worse results.

With the hybridized MSEM approach, there is a very natural approach to approximate the operator $\mathcal{L}^{-1}$ in a local way. The system $\widetilde{\mathcal{L}}x = y$ in the hybridized approach is given by Equation 6.30, where the local systems are discretized by $\mathbf{A}$ and the additional constraints in $\mathbf{N}$ are added to enforce continuity and boundary conditions with Lagrange multipliers $\vec{\lambda}$. The block $\mathcal{A}$ is actually itself a block-diagonal matrix, meaning that it can be inverted on per-element basis. This local inversion sometimes necessitates additional constraints in the case when the individual element matrices are not inevitable. These can be generated by forcing the error to be zero on the boundary of each element strongly.

$$\begin{bmatrix} \mathbf{A} & \mathbf{N}^T \\ \mathbf{N} & 0 \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{\lambda} \end{bmatrix} = \begin{bmatrix} \vec{y} \\ \vec{\phi} \end{bmatrix} \tag{6.30}$$

As such, if the constraints were to be removed, the result would be that the inverse of the system in Equation 6.30 could be inverted for each element independently. This approximation sacrifices the continuity and consequently enforces that only representable functions on each element are *bubble functions*. This is because in MSEM, 0-forms and 1-forms have natural boundary conditions, which Lagrange multipliers provide when constraints are added. Without them, the solution for each element will be exactly what it would be with zero natural boundary conditions for the 0-forms and 1-forms on that element. What this means is that when the error equation Equation 6.29 is solved, the resulting error estimate will predict zero error on element boundaries.

From there, a problem with this approximation is quite evident - whilst it can probably capture error caused by failure to resolve the forcing $\widetilde{f}$ locally, it can not catch the propagation of the error from one element to another. As such, whenever the solution has locally high error, this approach will likely not be able to locate it very well, since the propagation of the error to neighboring elements can not happen.

Still, this approach warrants an investigation, because while its drawbacks do not look promising, the advantage it promises is quite substantial, as it allows for only a local problem to be solved, instead of having to actually deal with global problems.

6.5.4 Using Projection

From noticing that the method in subsection 6.5.2 using the difference of solution computed on two different mesh sizes can be reformulated as taking a difference of the solution with its coarsened projection, a family of estimation methods can be defined. The underlying concept is to compute a solution on a fine mesh, then project it on a coarser mesh using some projector. The coarse projection of the solution is then (exactly) projected to the original polynomial space, where the difference of the original and the coarsened solution is taken as the error estimate.

The reasoning behind this can also be seen as that since when the solution is well converged, slightly reducing its order should not have much of an impact on its value. To illustrate, if the real solution is locally third order and computed on a fifth order element, it should be the same when it is coarsened down to a fourth or third order, then projected back to the fifth order.

Based on how the projector is defined, the solution obtained will be different. If the forward projector from fine to coarse polynomial space is defined by $\mathcal{P}$ and the Hermite adjoint projector from coarse to fine is defined as $\widehat{\mathcal{P}}$, then the definition of this error estimate is given as per Equation 6.31. When to projector is taken to be $\mathcal{P} = \mathcal{P}_{\mathcal{L}}$ and the adjoint projector is the L^2 projector $\widehat{\mathcal{P}} = \widehat{\mathcal{P}}_{L^2}$, the method from subsection 6.5.2 is recovered.

$$e \approx u - \widehat{\mathcal{P}}\mathcal{P}u = \left(1 - \widehat{\mathcal{P}}\mathcal{P}\right) u \tag{6.31}$$

Another projector of interest is also the L^2 projector, since it can be applied locally, and is cheap. As such, it will also be one of the refinement criterion that will be used for the numerical tests.

6.5.5 Using VMS

The theory pertaining to VMS is covered more extensively in chapter 5, so here only its usage as pertaining to mesh refinement is discussed. It follows quite clearly that the most useful result one can use from VMS is the reconstructed unresolved scales u', which can be obtained after their effect through the non-symmetric operator is evaluated.

As already touched upon in section 5.2, the unresolved scales' reconstruction on the fine mesh u' allows for an error estimate. It actually leads to a situation similar to that when using a higher order solution, as discussed in subsection 6.5.2. Essentially, since u' can not give higher accuracy than the order and resolution of the fine mesh that VMS uses to approximate the continuous space, it can be seen that u' fills the role of the difference of the coarse and fine solution, which can in turn be used as the error estimator.

As was discussed in section 5.3, imitation of VMS was used, since the implementation of could not be completed in time. To briefly recap the process used, which was discussed in more depth in section 5.3, the solution was obtained on the fine mesh for the symmetric part of the problem, then again on the coarse mesh. The difference of the two was what was counted as the unresolved scales u', since that is what theoretically the best possible result would be.

Chapter 7

Results

This section evaluates the performance of MSEM with the refinement methods based on section 6.4. This was done using a few error estimation techniques and problems, to show performance and behavior of the method in different situations, as well as to show how specific methods compare.

7.1 Refinement Criterion Used

The refinement criterion used was, as mentioned before, based on section 6.4. The refinement process was done as follows:

1. Solve problem
2. Estimate error for each element
3. For N_E elements with the highest error:
    (a) Estimate the ratio of h-refinement error increase to element error based on Equation 6.23,
    (b) If bellow E_H and order $p \geq 2$, split into four elements of order $\lfloor \frac{p+1}{2} \rfloor$
    (c) Otherwise increase order to $p + 1$,

This process is governed by the hyper-parameters N_E and E_H. Increasing N_E allows for more elements to be refined per iteration. A lower value means that more iterations must be done, but it permits for more accurate refinement, as each element which is either h- or p-refined will end up changing the error distribution. At the same time, if only a small part of the domain needs to be refined, so a high value of N_E means that a large part of the domain will be unnecessarily refined.

As for the E_H, if the value is very low, then no h-refinement takes place, which makes it local p-refinement only. On the other hand, a very high value will allow for h-refinement in the elements that would likely benefit much more from h-refinement.

7.2 Error Estimators

To be able to use process in section 7.1, error needs to be estimated. There are a few ways used, each with their own advantages and drawbacks. The ones discussed here are:

- Using the exact solution,
- Using the difference from the higher order solution,
- Using bubble function approximation to invert the residual,
- Using the difference from a coarser L^2 projection,
- Using unresolved scales from VMS (or imitation of it).

It should be noted that since VMS only works on non-symmetrical problems, those are the only ones where it is used. Otherwise, it would be exactly equivalent to using a higher order solution. In the plots in the following section, the following labels are used for these respectively:

- ”exact” - exact error,
- ”finer solve(n)” - using the difference from a solution computed on a mesh n orders finer,
- ”local inv(n)” - using bubble functions for local inversion with the finer space being n orders higher,
- ”local drop(n)” - using the difference from the L^2 projection coarser by n orders,
- ”vms(n)” - using unresolved scales from VMS with finer space being n orders finer.

7.3 Test Problem Overview

In order to check how well local refinement performs for different error estimates, some test problems first had to be defined. As such, performance of a specific criterion could be judged and compared to some baselines. The key properties of the test cases were:

- Have an analytical solution - necessary to compute exact error,
- Have very non-uniform error distribution - otherwise local refinement is useless,
- Be finite - otherwise there is no way to compute it,
- Be steady - this is for the sake of ease of comparison and does not mean these techniques can not be used in unsteady cases.

Based on these criteria, the test cases presented in Table 7.1 were used. The specific meshes, manufactured solutions, and systems of equations being solved are explained in more detail in their individual subsections. For each of these refinement is performed using different error estimators.

Table 7.1: Summary of test problems used, the systems of equations solved, and what the meshes they were solved were.

Case Number	Solution	Problem	Mesh
1	Gaussian	mixed Poisson	deformed square
2	shock	adv-dif	deformed square
3	singular corner	direct Poisson	unit step
4	quadratic sine	Navier-Stokes	deformed square

7.3.1 Overview of Meshes

Since only two distinct meshes were used, they are briefly covered in this subsection to avoid repetition in the subsequent sections. These meshes are then used as baseline cases for all refinement tests.

Deformed Square

First is the ”deformed square” mesh. This is a mesh obtained by first creating an uniform mesh on a square $(\xi, \eta) \in [-1, +1]$, then defining a mapping to physical coordinates $(x, y) \in [-1, +1]$. The coordinates of the mesh were thus given by Equation 7.1 and Equation 7.2, with the value of perturbation amplitude c taken as $c = 0.1$. As for the original mesh resolution and order, five equal size elements of polynomial order $p = 3$ were used in each direction.

$$x(\xi, \eta) = \xi + c \sin(\pi\xi) \sin(\pi\eta) \tag{7.1}$$

$$y(\xi, \eta) = \eta - c \sin(\pi\xi) \sin(\pi\eta) \tag{7.2}$$

Unit Step

Unit step is used for the case with the ”singular corner” solution, because that solution rather nice behavior for $\theta \in \left[-\frac{\pi}{2}, \pi\right]$, as such, it resembles a corner, when only that part of the domain is considered. For the sake of simplicity, the mesh was left undeformed, as well as doubly uniform in terms of element sizes. The polynomial order chosen for all elements was $p = 3$, just as for the ”deformed square” mesh.

7.3.2 Iterative Refinement Setup

For each case, the mesh was iteratively refined. This was done for the same number of rounds for each of the error estimators, to be able to see how effective they are compared to one another. As for when each round of iterative refinement would be stopped, a few options were considered, which are all possible in practice, assuming all elements are sorted from the one with the highest error estimate to the lowest:

- when a specific number of elements has been refined,
- when error estimate in the next element is bellow an absolute or relative threshold,
- when a specific number of degrees of freedom has been added to the system.

For these test cases the first one has been chosen, and specifically set to allow for only up to 10 % of all elements to be refined each step. This is admittedly a somewhat arbitrary number, but it should limit refinement to only the necessary elements, whilst scaling as the number of elements grows if h-refinement is used. This deals with the refinement hyper-parameter N_E from section 7.1.

There is also the matter of selecting the h-refinement threshold E_H to use for the h-refinement criterion. This is again taken to be at 0.2, which means that it is estimated that converting that element from order $2p$ to p would cause an increase in error of about 20 %. This choice is again somewhat arbitrary, with the main guide for it being that it can not be too large, as otherwise only h-refinement would occur. To show that this value works reasonably well, the tests always compare the results obtained to the version with only p-refinement ($E_H = 0$) to show how h-refinement changes the behaviour.

7.4 Test Case 1: Gaussian with Mixed Poisson

For this case, the system being solved was the Poisson problem in the mixed formulation. Using differential forms, it is expressed as system of Equation 7.3 and Equation 7.4.

$$\int_\Omega p^{(1)} \wedge \star q^{(1)} + \int_\Omega \mathrm{d}p^{(1)} \wedge \star u^{(2)} = \int_{\partial\Omega} p^{(1)} \wedge \star u^{(2)} \quad \forall p^{(1)} \in \Lambda^{(1)}(\Omega) \tag{7.3}$$

$$\int_\Omega v^{(2)} \wedge \star \mathrm{d}q^{(1)} = \int_\Omega v^{(2)} \wedge \star f^{(2)} \quad \forall v^{(2)} \in \Lambda^{(2)}(\Omega) \tag{7.4}$$

The manufactured solution $u^{(2)}$ was a Gaussian given by Equation 7.5, with parameters $r = 40$, $x_0 = y_0 = 0.5$. This gives the source term f as per Equation 7.6. This solution is locally very sharp.

$$u(x, y) = e^{-r\left((x-x_0)^2+(y-y_0)^2\right)} \tag{7.5}$$

$$f(x, y) = 4r\left(r\left((x-x_0)^2 + (y-y_0)^2\right) - 1\right) e^{-r\left((x-x_0)^2+(y-y_0)^2\right)} \tag{7.6}$$

The mesh used for this case was "deformed square", as discussed in section 7.3.1. The plot of the exact solution and the absolute value of the error at the initial polynomial order of $p = 3$ can be seen in Figure 7.1. The solution is very sharp, and consequently the error is very localized, having a much larger magnitude in the top right corner, where it exceeds $2 \cdot 10^{-1}$, while in the remainder of the domain it hovers around $1 \cdot 10^{-5}$.

The plot of error in the L^2 norm against the number of degrees of freedom for different error estimates strategies is presented in Figure 7.2. It is quite clear that for this case the refinement criterion based on local inversion using bubble functions is not suitable, at least given the refinement strategy hyper parameters, as the method results in only local h-refinement, which in turn results in linear convergence, as opposed to exponential convergence offered by p-refinement for smooth solutions.

The refinement criterion state variables for the last iteration of the "local inv(3)" refinement criterion is shown in Figure 7.3. The refinement method has correctly identified the region where the error originates, but the cost of h-refinement compared to p-refinement is grossly under-estimated.

To be able to judge the other methods, the plot of convergence from Figure 7.2 is repeated in Figure 7.4, but with the "local inv" methods removed. This allows for clearer examination of the other methods and how they compare to how the refinement would be done if the exact error information would be provided. What can be seen that on average, the behaviour between these is quite similar - mostly applying p-refinement, with an occasional h-refinement, which causes an occasional jump in the L^2 error, before p-refinement benefits from that improvement. What can also be seen is that both cases for "local drop" method result in about the same convergence as the exact error information.

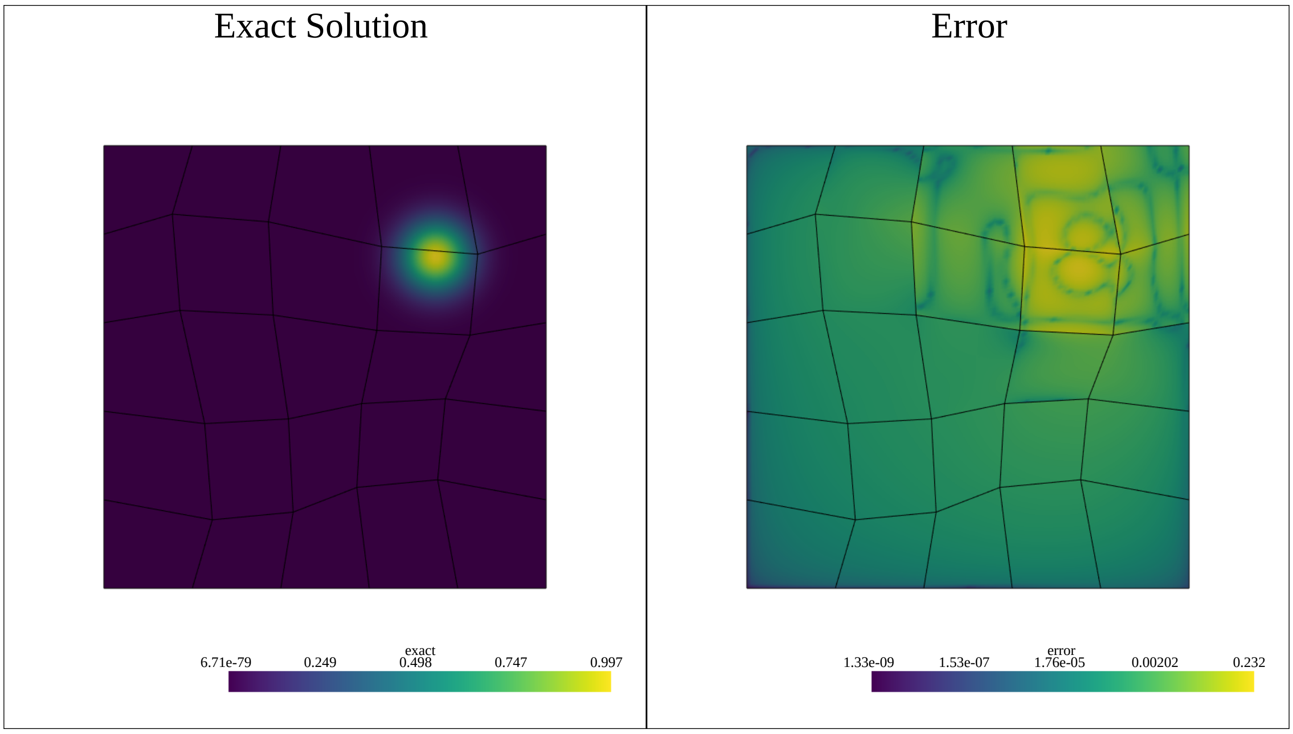


Figure 7.1: Plot of the exact solution along with the error obtained on the initial mesh

The end result of the three other methods is quite similar in nature. For the illustrative purposes, the result is presented in Figure 7.5, which shows the state of the error and estimates at the last iteration for ”local drop(2)” criterion.

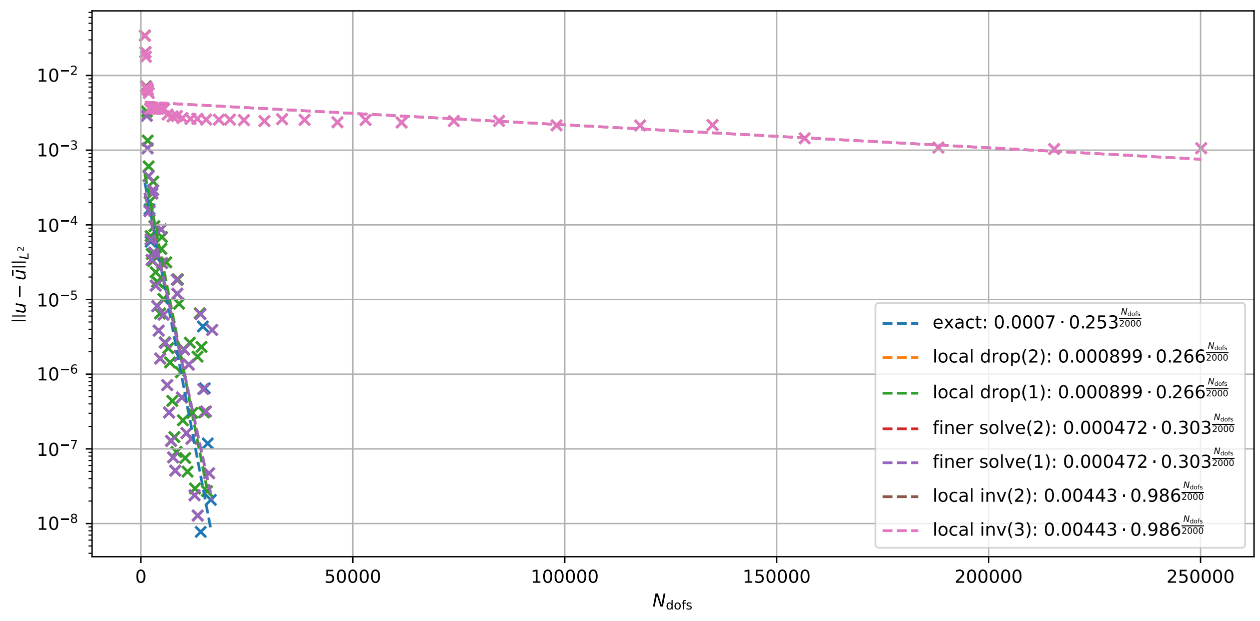


Figure 7.2: Convergence in the L^2 norm for different error estimates strategies.

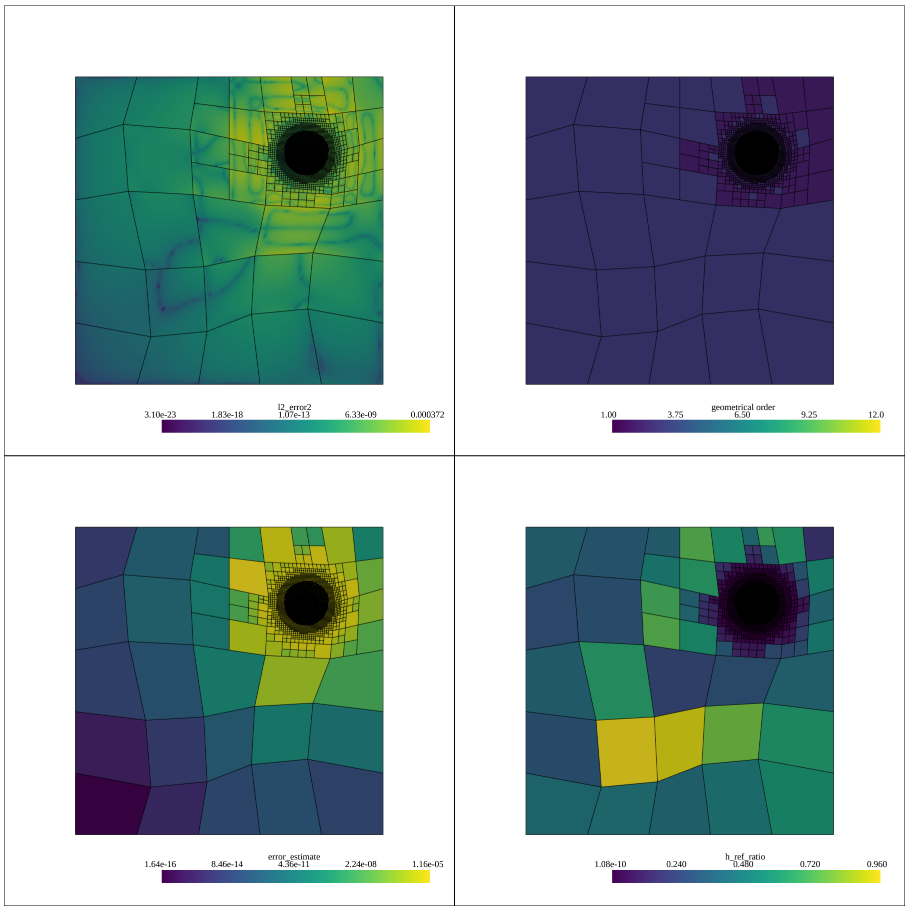


Figure 7.3: State of the "local inv(3)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

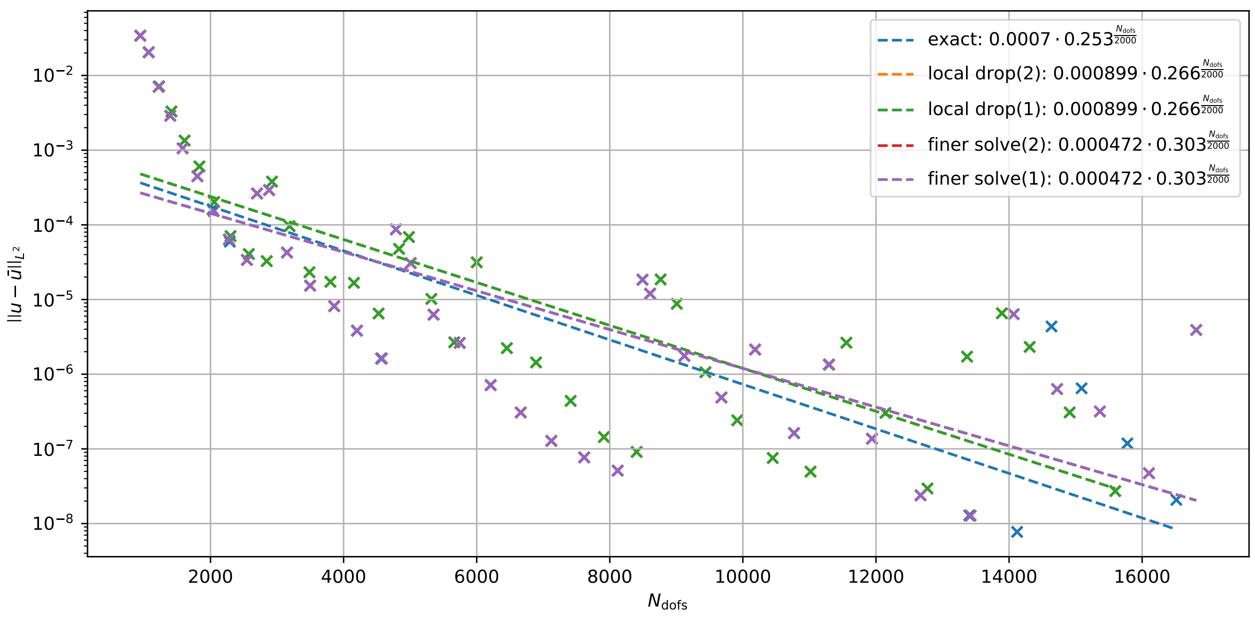


Figure 7.4: Convergence for different error estimates strategies in the L^2 norm .

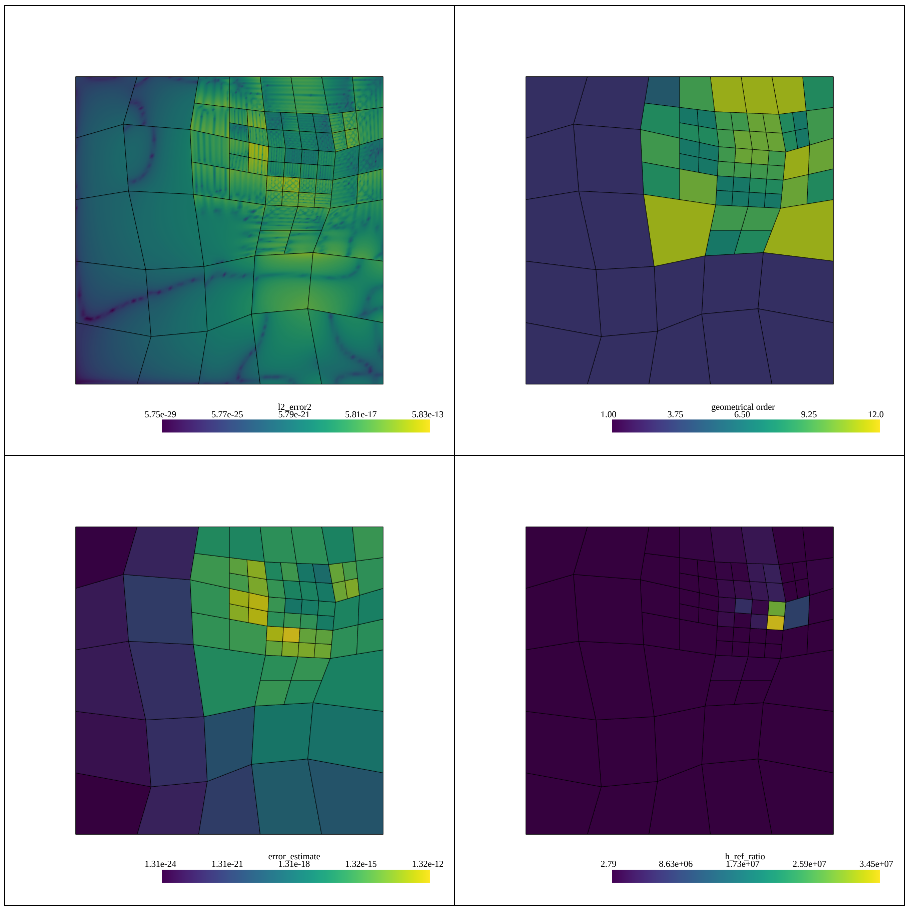


Figure 7.5: State of the "local drop(2)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

7.5 Test Case 2: Shock with Linear Advection-Diffusion

The second test case is based on the linear advection-diffusion equation. In the mixed formulation, it is similar to the mixed Poisson case, but with an advection term added. As such, the system of equations has the first part given by Equation 7.7 equivalent to the mixed Poisson's in Equation 7.3, but the second part in Equation 7.8 has the additional term. Note that because it is in the mixed formulation, the interior product was moved to the weight form.

$$\int_\Omega p^{(1)} \wedge \star q^{(1)} + \int_\Omega \mathrm{d}p^{(1)} \wedge \star u^{(2)} = \int_{\partial\Omega} p^{(1)} \wedge \star u^{(2)} \quad \forall p^{(1)} \in \Lambda^{(1)}(\Omega) \tag{7.7}$$

$$\int_\Omega v^{(2)} \wedge \star \mathrm{d}q^{(1)} + \int_\Omega i_{\vec{a}} v^{(2)} \wedge \star q^{(1)} = \int_\Omega v^{(2)} \wedge \star f^{(2)} \quad \forall v^{(2)} \in \Lambda^{(2)}(\Omega) \tag{7.8}$$

The advection vector field $\vec{a}$ from Equation 7.8 was given by Equation 7.9. This vector field was chosen to be smooth, so as to make the manufactured solution the primary source of error.

$$\vec{a}(x,y) = (x^2 + y)\frac{\partial}{\partial x} + y(1-x)\frac{\partial}{\partial y} \tag{7.9}$$

The manufactured solution for $u^{(2)}$ for this case was given by Equation 7.10, with parameters $\alpha = 100$, $r_0 = 1$, $x_0 = 1.5$, and $y_0 = -1.5$. The plot of this solution and its error on the initial mesh can be seen in Figure 7.7. The solution results in a very sharp transition from being around -1 to then go to +1. The profile cross section of this solution as a function of distance from the point (x_0, y_0) for $\alpha = 100$ and $r_0 = 1$ is plotted in Figure 7.6.

$$u^{(2)}(x,y) = \frac{2}{\pi}\arctan\left(\alpha\left(\sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0\right)\right) \tag{7.10}$$

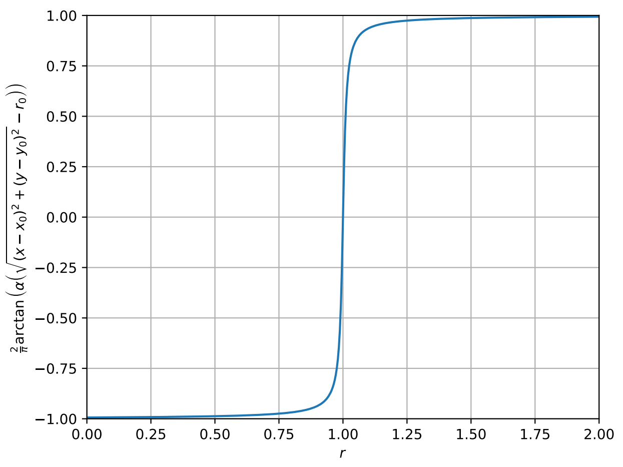


Figure 7.6: Cross section of the solution as a function of r for $r_0 = 1$ and $\alpha = 100$

The convergence plot for this problem is presented in Figure 7.8. Since the PDE being solved is not symmetrical, results for VMS imitation are also present. The refinement criteria based on ”local inv” are once again the worst performing ones, as they converge at much lower rate. Looking once again at the state of the last iteration in Figure 7.9. The error estimator fails to identify that the error is concentrated on the shock region and just performs uniform h-refinement.

These results are in quite a stark contrast to how the refinement strategy based on solving on a fine mesh behaves. It can be seen in Figure 7.10 that the result is a combination of h- and p-refinement, which results in high mesh resolution near the shock and increased polynomial order. What is also very clear is that the ”local drop” method is the slowest to have any changes. This is due to it avoiding h-refinement and over-estimating the error away from the shock. The result is the state presented in Figure 7.11, which

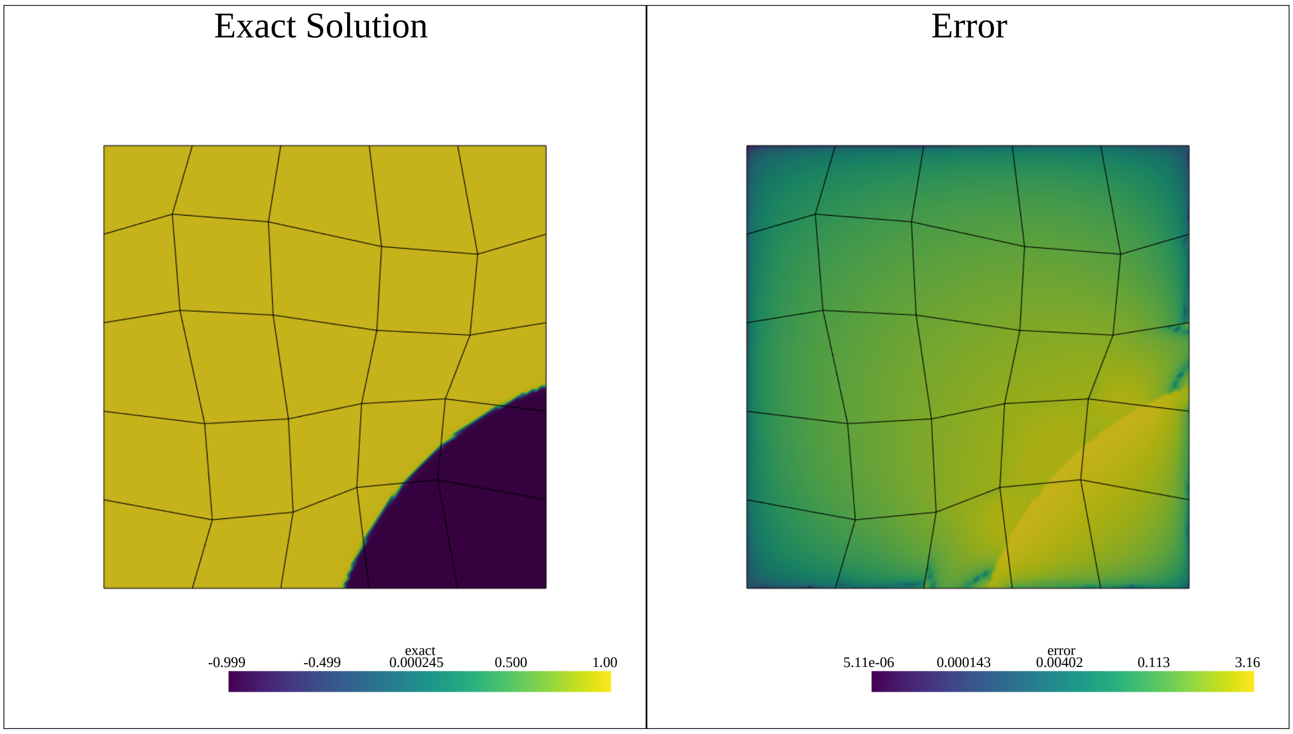


Figure 7.7: Plot of the exact solution along with the error obtained on the initial mesh

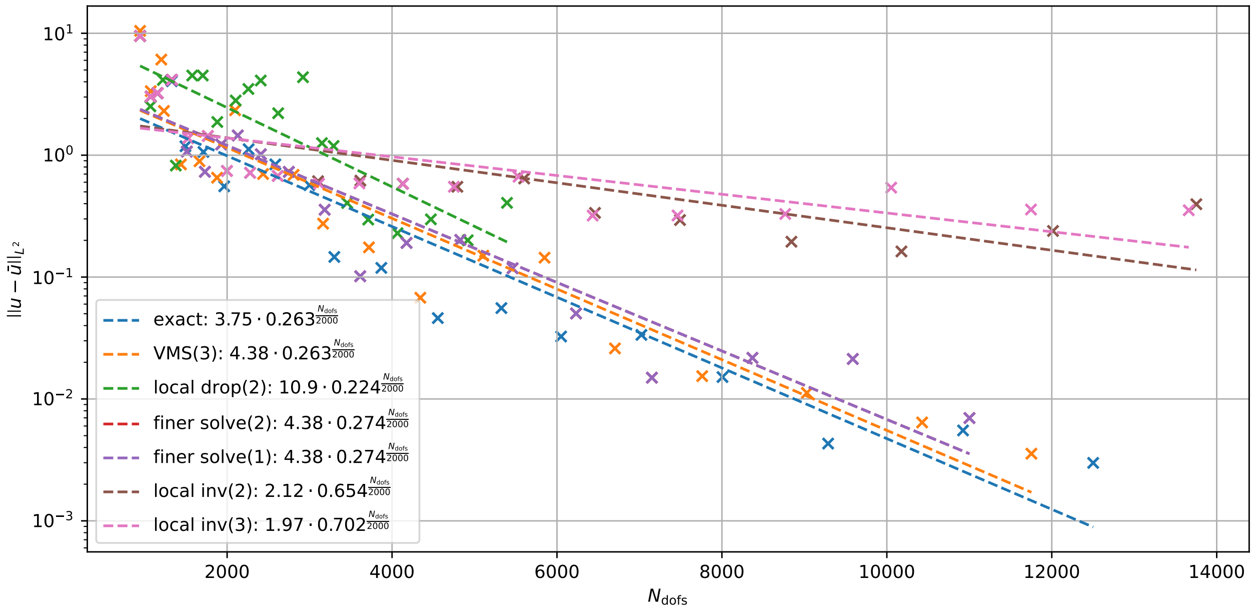


Figure 7.8: Convergence for different error estimates strategies in the L^2 norm.

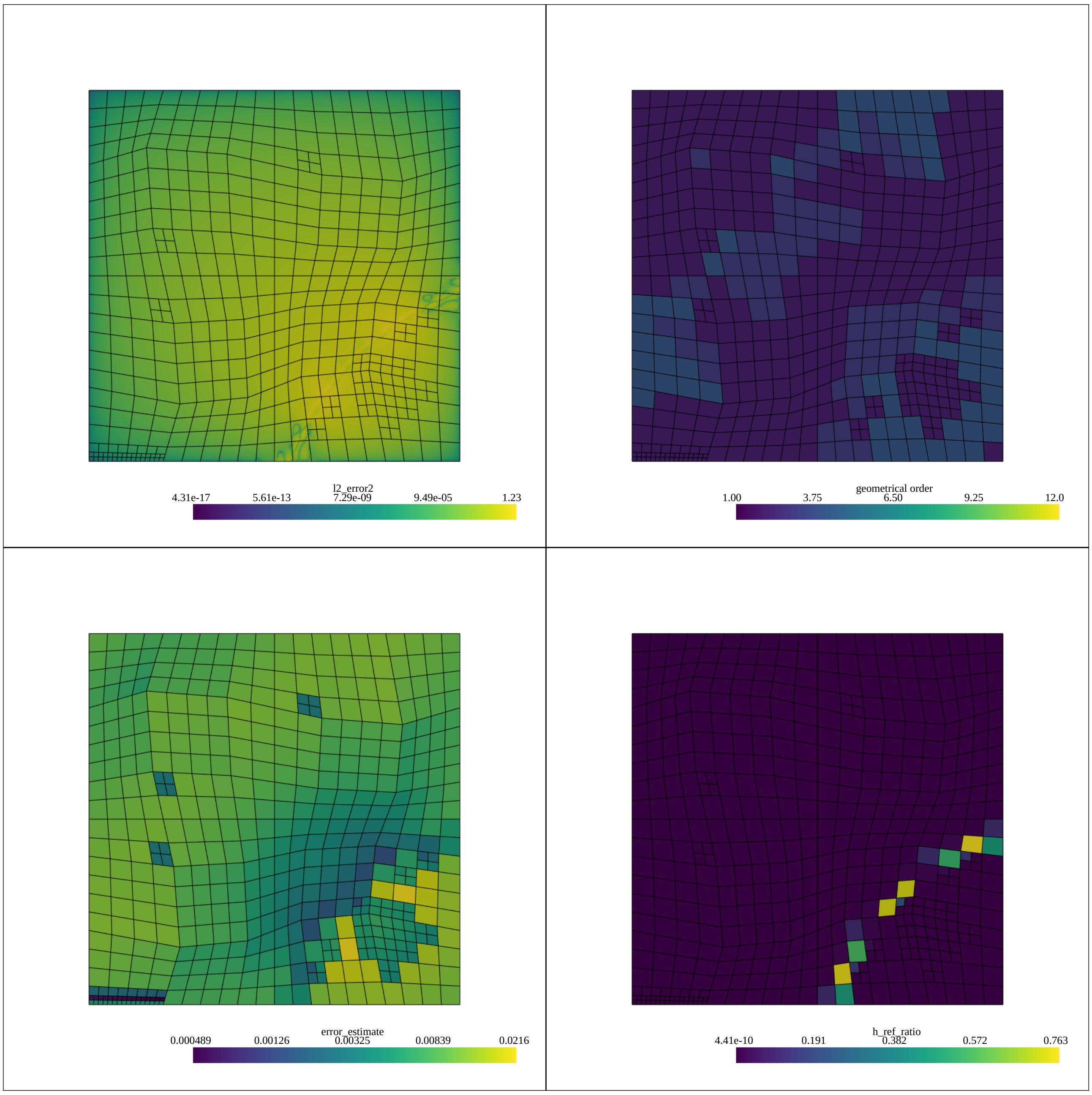


Figure 7.9: State of the "local inv(3)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

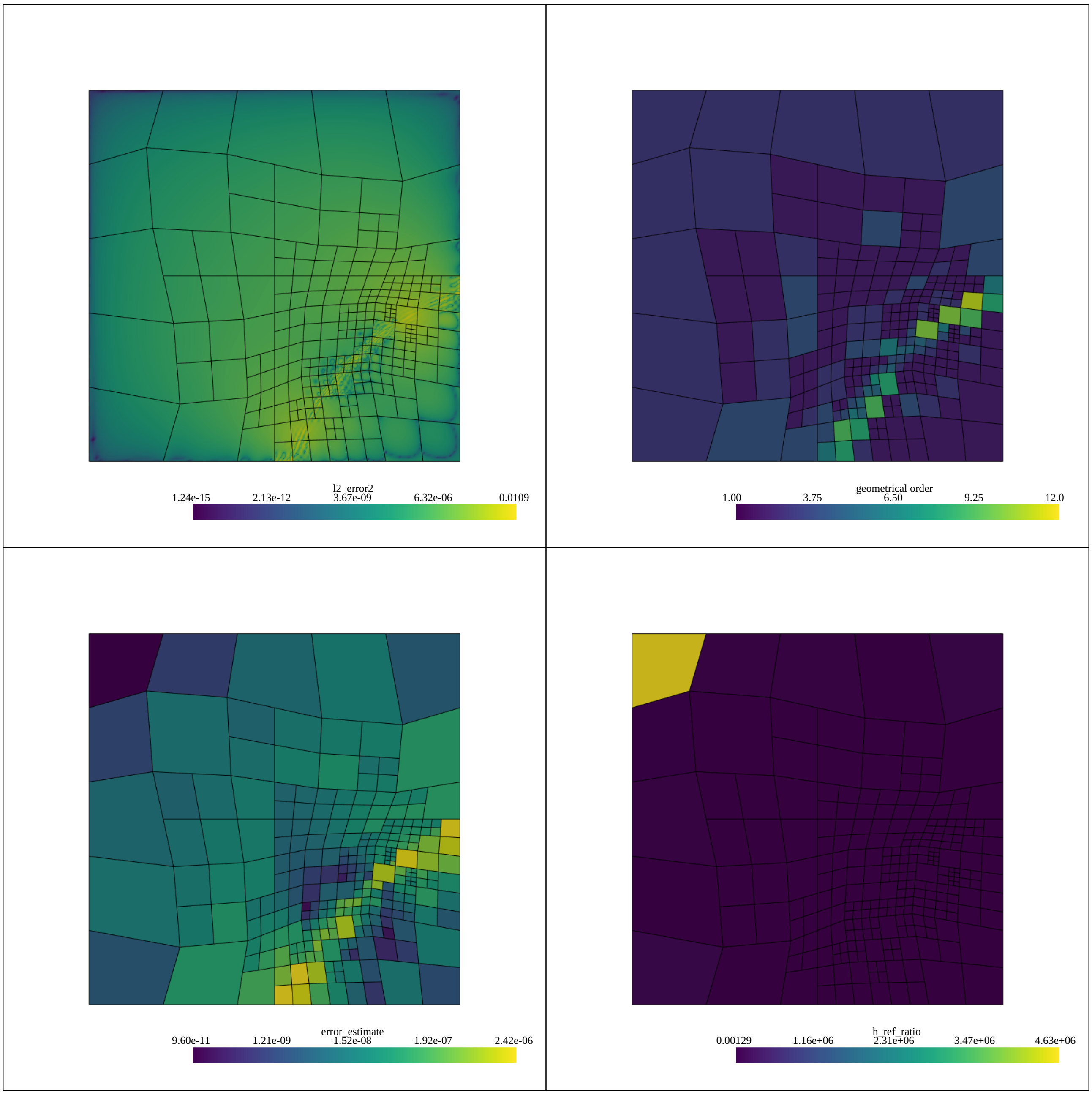


Figure 7.10: State of the ”finer mesh(3)” criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, *h*-refinement cost estimate ratio.

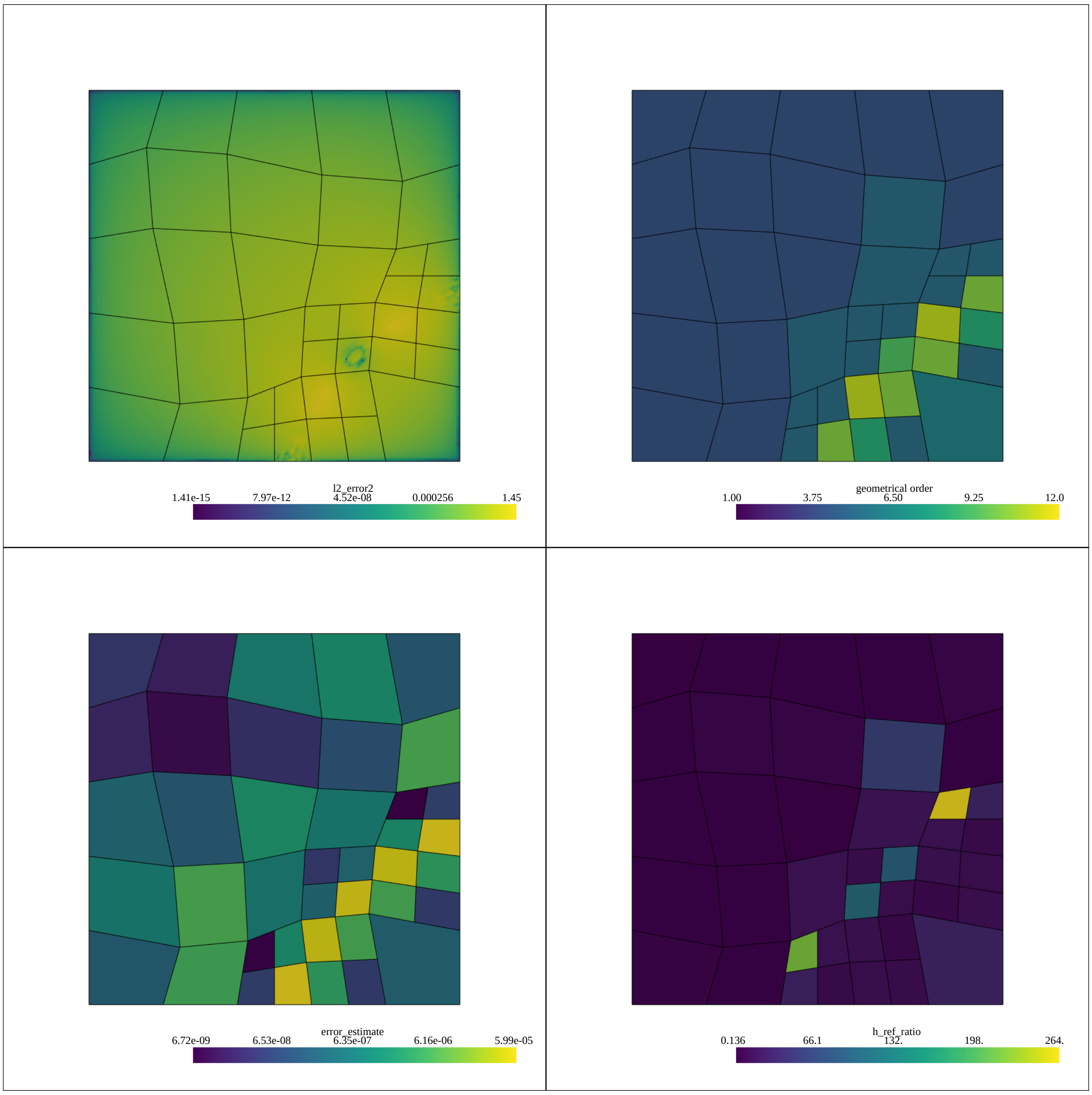


Figure 7.11: State of the "local drop(2)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

7.6 Test Case 3: Corner Singularity with Direct Poisson

For this test case, the equation being solved was the Poisson equation in the direct formulation, as given by Equation 7.11. The reason for choosing Poisson problem in direct formulation is because the solution has a singular gradient at the origin, as

$$\int_{\Omega} \mathrm{d}v^{(0)} \wedge \star \mathrm{d}u^{(0)} = \int_{\Omega} v^{(0)} \wedge \star f^{(0)} + \int_{\partial\Omega} v^{(0)} \wedge \star \mathrm{d}u^{(0)} \quad \forall v^{(0)} \in \Lambda^{(0)}(\Omega) \tag{7.11}$$

The mesh used was changed to the unit corner mesh, as discussed in section 7.3.1. The solution for this case was given by Equation 7.12, where $r^2 = x^2 + y^2$ and $\tan\theta = \frac{y}{x}$. This solution and the initial error on the mesh with order $p = 3$ can be seen in Figure 7.12.

$$u^{(2)} = r^{\frac{2}{3}} \sin\left(\frac{2\theta + \pi}{3}\right) \tag{7.12}$$

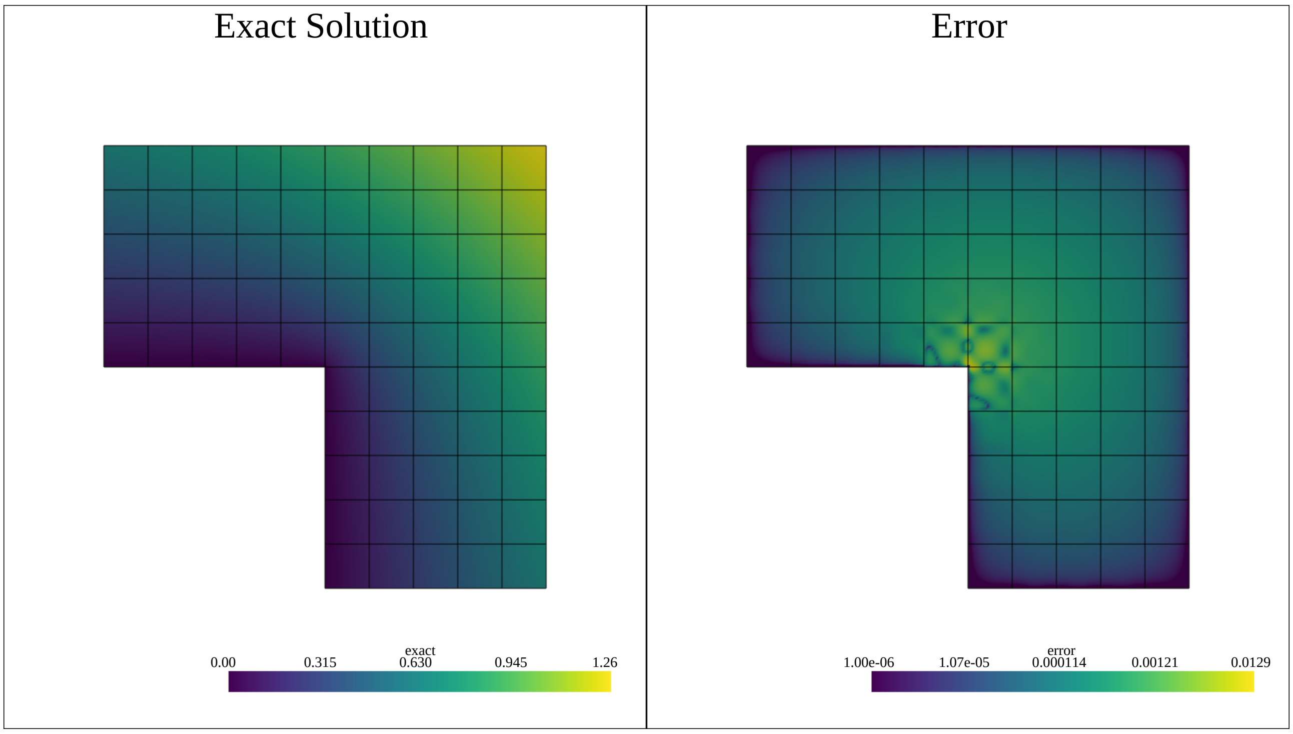


Figure 7.12: Plot of the exact solution along with the error obtained on the initial mesh

The convergence of the refinement based on the different error estimators is presented in Figure 7.13. What can be seen is that for the chosen refinement strategy, the exact error is actually the worst input to give. This is likely mostly due to the fact that performing any sort of local refinement may introduce errors due to different continuity and interfacing of different elements. While this likely says more about the refinement strategy's suitability rather than anything else, all other error estimates gave better performance.

Even the "local inv" error estimates work correctly. Perhaps this is not too surprising, since the solution has a singular gradient at the corner and as such, the local inversion with bubble functions method might be capable of picking up that difference in that region, which is also where the most error is present. Another important factor to consider is the fact that refinement is performed in an iterative way, which means that even with the exact error, one can only guarantee that the error will be best reduced in one iteration, which might make the subsequent iteration worse.

To see how the meshes produced by the exact solution vary from the other ones, the final states for the exact error and the "local inv(3)" error estimator can be seen in Figure 7.14. The state for the exact error estimate is shown in Figure 7.14a and is seen to produce a refined mesh with h-refinement quite far from the corner with the largest error concentration and having many elements even further away from the corner. To contrast this, refinement based on using bubble function inversion of the residual is shown in Figure 7.14b, which results in a very focused mesh refinement. The elements further from the corner have higher order, but h-refinement is only really performed at the corner, which results in much more efficient refinement.

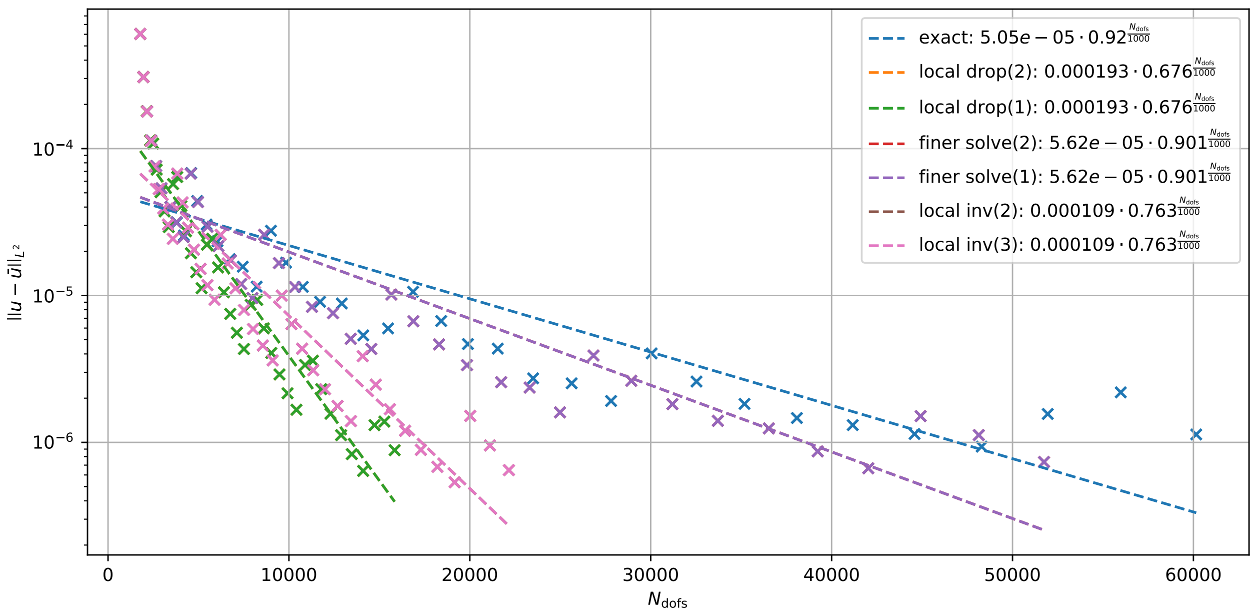
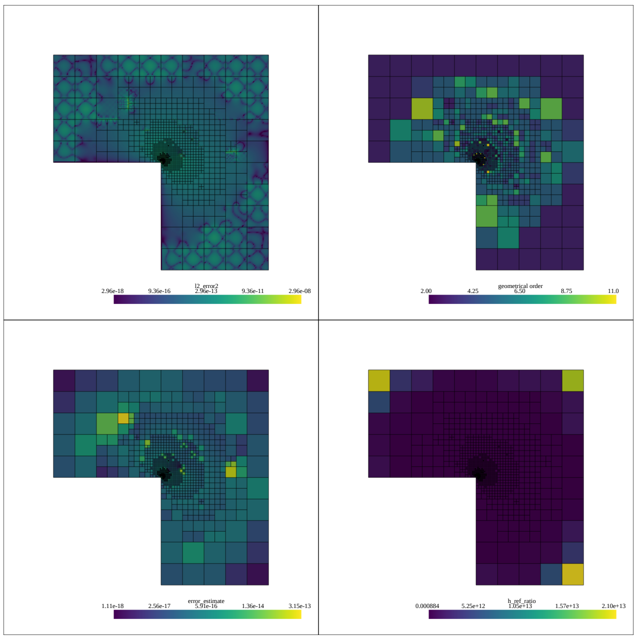
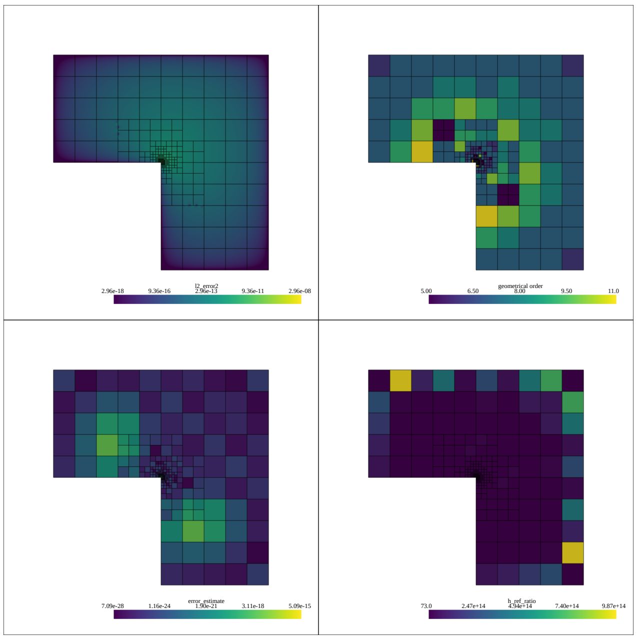


Figure 7.13: Convergence for different error estimates strategies in the L^2 norm.



(a) State of the "exact" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.



(b) State of the "local inv(3)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

Figure 7.14: Comparison of the error estimator state for the "exact" and "local inv(3)" cases.

7.7 Test Case 4: Quadratic Sine with Navier-Stokes

The last test case used was based on the (steady) Navier-Stokes equations. This system is given by three equations - the vorticity relation in Equation 7.13, the momentum equation in Equation 7.14, and the mass conservation equation in Equation 7.15. Note that in the momentum equation Equation 7.14 involves the total pressure p, which means that the "real" pressure has been combined with (half) the square magnitude of the velocity.

$$-\int_\Omega \phi^{(0)} \wedge \star\omega^{(0)} + \int_\Omega \star \mathrm{d}\phi^{(0)} \wedge u^{(1)} = \int_{\partial\Omega} \phi^{(0)} \wedge \star u^{(1)} \qquad \forall \phi^{(0)} \in \Lambda^{(0)}(\Omega) \tag{7.13}$$

$$\int_\Omega v^{(1)} \wedge \star \mathrm{d}\omega^{(0)} + \int_\Omega \mathrm{d}v^{(1)} \wedge \star p^{(2)} = \int_{\partial\Omega} v^{(1)} \wedge \star p + \int_\Omega i_{u^{(1)\flat}} v^{(1)} \wedge \star\omega^{(0)} \qquad \forall v^{(1)} \in \Lambda^{(1)}(\Omega) \tag{7.14}$$

$$\int_\Omega r^{(2)} \wedge \star \mathrm{d}u^{(1)} = 0 \qquad \forall r^{(2)} \in \Lambda^{(2)}(\Omega) \tag{7.15}$$

The L^2 norm of the manufactured solution for $u^{(1)}$ used for this case was given by Equation 7.16, with the parameter $\omega = 10$. This solution is divergence free ($\mathrm{d}u^{(1)} = 0$), so it satisfies Equation 7.15 without the need to add any source terms. However, to have a pressure solution as $p^{(2)} = 0$, a source term had to be added to the right side. The exact solution and error on the initial mesh is presented in Figure 7.15.

$$u^{(1)}(x, y) = \left(\sin\left(\omega x^2\right)\cos\left(y\right)\right)\mathrm{d}y - \left(2\omega x \cos\left(\omega x^2\right)\sin\left(y\right)\right)\mathrm{d}x \tag{7.16}$$

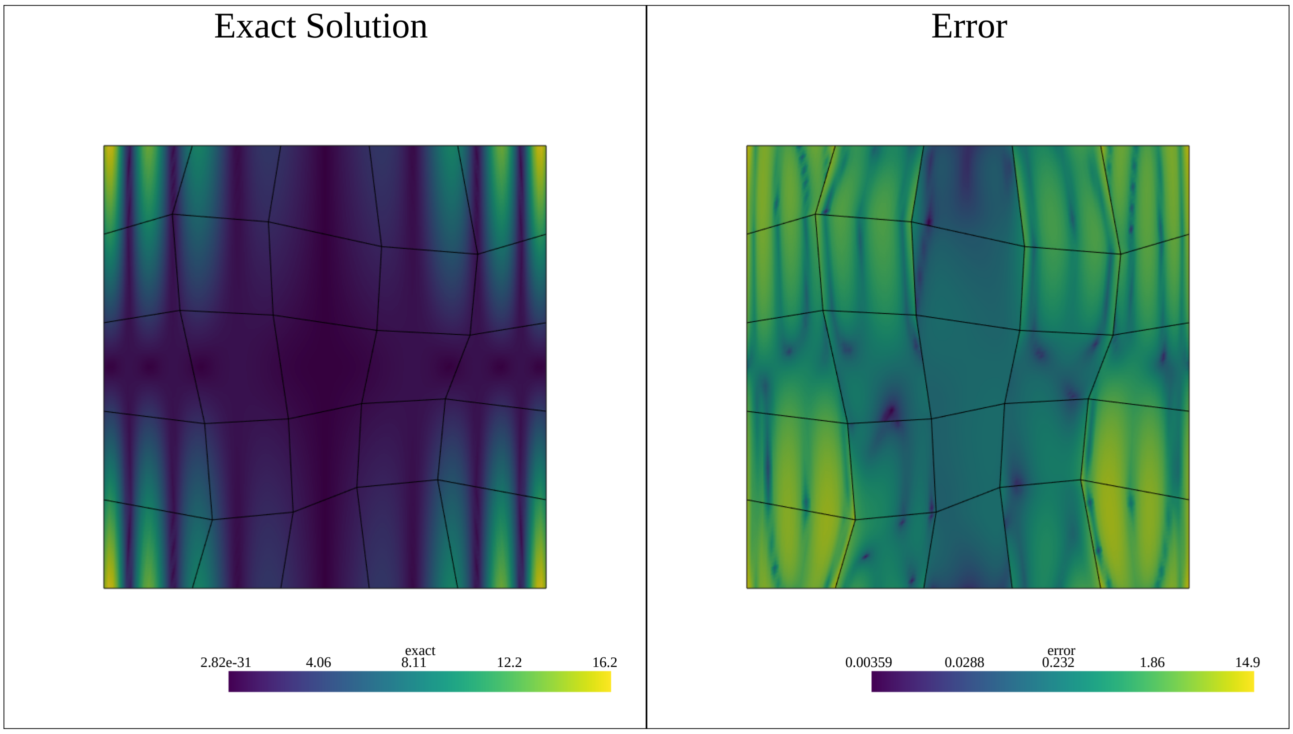


Figure 7.15: Plot of the exact solution magnitude along with the error obtained on the initial mesh

The convergence plot comparing the different error estimators is shown in Figure 7.16. Once again, using bubble functions for local inversion performs the worst, which is largely due to it incorrectly predicting both the error magnitude and distribution. This leads to very suboptimal refinement choices. To be able to better see the differences between the other, more useful error estimates, the figure without the bubble functions-based ones is repeated in Figure 7.17. What can be seen is that the methods based on "local drop" are somewhat worse off compared to VMS, exact, and using a finer solution.

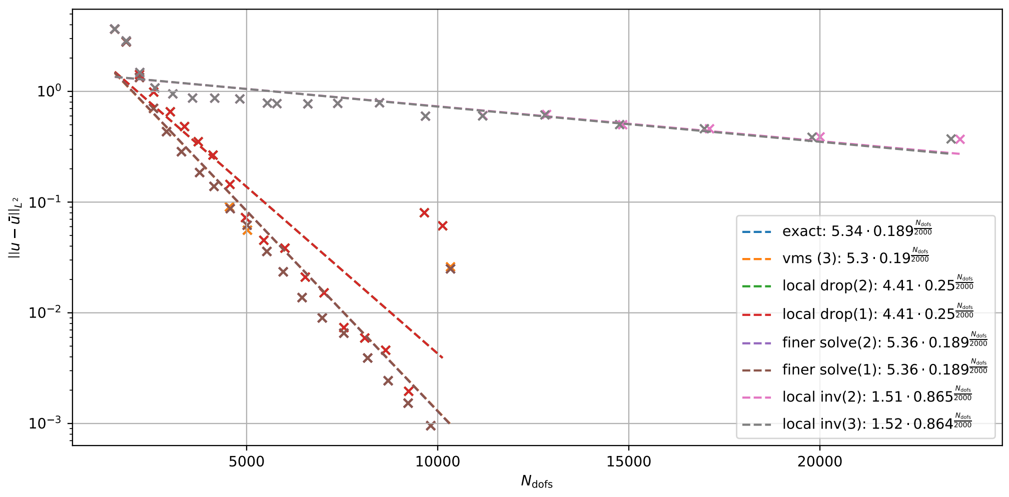


Figure 7.16: Convergence for different error estimates strategies in the L^2 norm.

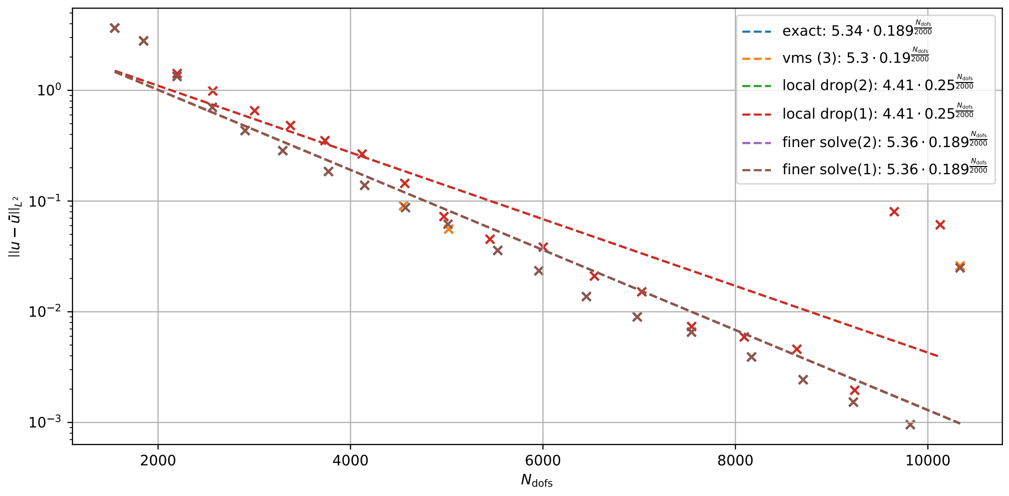
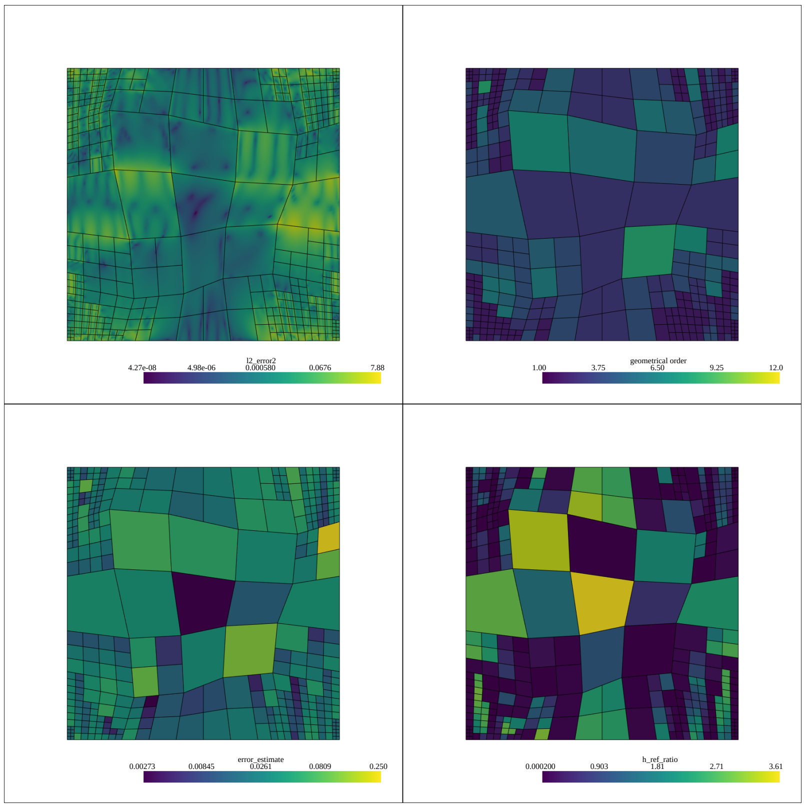
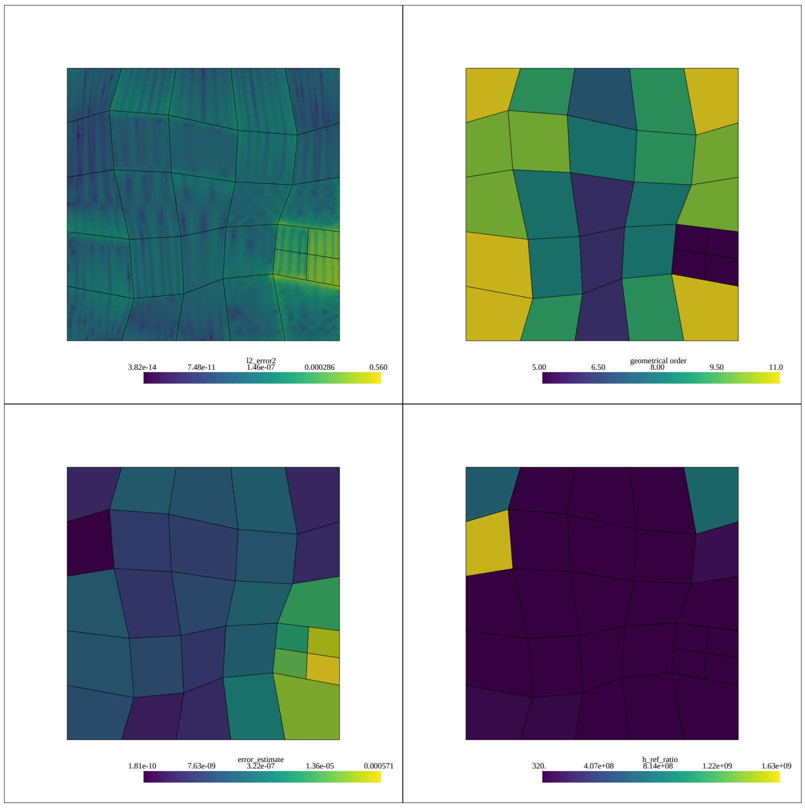


Figure 7.17: Convergence for different error estimates strategies in the L^2 norm.

In the end, there are two main groups of refinement paths that were taken based on different error estimators - those taken by bubble functions based ones and then those taken by all others. The results of the former is illustrated by the final state of refinement for the bubble functions based, which is shown in Figure 7.18a. As representative of other, more successful error estimators, the VMS-based results can be seen in Figure 7.18b. A clear difference between the two is that the error estimate obtained through bubble function local inversion method is wildly inconsistent with reality, with refinement thus being located in parts of the domain, which are not as important, along with the fact that h-refinement was once again strongly favoured at the cost of the p-refinement, which would be much more efficient for this case, given that the exact solution was composed of sine and cosine functions.



(a) State of the "local inv(3)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.



(b) State of the "vms (3)" criterion. Left to right and top to bottom these are: exact error distribution, order of elements, error estimate of the method, h-refinement cost estimate ratio.

Figure 7.18: Comparison of the error estimator state for the "vms (3)" and "local inv(3)" cases.

7.8 Analysis of Test Case Results

Quite some conclusions can be drawn from the four test cases from section 7.4, section 7.5, section 7.6, and section 7.7. First, the very obvious conclusion that could have been predicted from the beginning is that there is no criterion, which is the best in every single situation with the refinement strategy which was employed. The refinement strategy used still had to be characterized by some hyper-parameters, as it was not derived how to otherwise perform local refinement to obtain the absolute optimal refinement, without knowing the solution perfectly in advance. As such, it should also be kept in mind that tweaking these hyper-parameters might somewhat change performance of some of the considered test cases.

When it comes down to the individual error estimators, each showed a quite consistent behaviour between the different cases. Using the exact error produced very consistent results, only really failing when the fact that each iteration the best choice is judged only on it's immediate impact, instead of it's long-term effect on the refinement path.

The error estimator, which in most cases resulted in about the same choices as the exact error was the method, which solved the problem first on the coarse, then on the fine mesh. This is no unexpected, since if the entire mesh is uniformly p-refined, the error should decrease by a similar factor everywhere. As such it makes sense that the relative difference between the two solutions can be a very good error estimator.

Very similar to the behaviour of the aforementioned error estimator based on a finer solution, VMS error estimation proved just as good, or even slightly better. It was only used for two cases with the non-symmetric differential operators, since it is otherwise exactly the same as the finer solution-based error estimator. It is also important to consider that the cost of computing these VMS results for non-linear equations, such as Navier-Stokes is amortized by the fact that only part of the system must be inverted on the fine mesh.

The next best error estimator was the very economical local order drop-based estimator. It was surprisingly useful in most cases, but tended to struggle with predicting the smoothness of the error distribution, thus while it would be able to identify regions with high error, it would often not choose between p-refinement and h-refinement correctly, which was most evident in the second test in section 7.5. However, that did at some point manage to work into its benefit, as in the third test in section 7.6, methods which were more accurate resulted in worse long-term performance.

The worst error estimator used is the proof that there's no such thing as a free lunch, or a free error estimator. Error estimators based on locally inverting the residual system to obtain error proved to not be completely useless for all cases, as it performed quite well in the third test case in section 7.6, where more accurate error estimators struggled, but overall, it was the worst estimator in all other cases, where it would very often lead to either local or even global h-refinement. This means that is incapable of predicting the smoothness of the error, as well as that it often struggles with even predicting it's L^2 norm. This is perhaps not surprising given how it is constructed, with forced zero boundary conditions for error on each element.

In summary, if the exact error is not available, which is what is true for all problems of any interest to the real world and even academia beyond just testing, the best one can use is either use VMS- or finer solution-based error estimator. The choice between the two has minimal impact on the results, so it mainly boils down to the ease of implementation and relative cost for the specific case. It should also be noted that if the problem is based on a symmetric PDE, then VMS is exactly equivalent to the finer solution-based case. VMS main advantage is that for non-linear cases, which would require iterative solvers, it provides an additional correction and very efficient error estimate for the cost of solving the symmetric and linear part of the differential operator on the finer mesh, rather than the entire PDE. If this is too expensive, even a local order drop through local L^2 projection could be feasible, as it would likely at least be able to identify the location where refinement should be done, though it might require a different and more specialized criterion to choose between h- and p-refinement better. As the worst of the tried criteria was the bubble function based local inversion based. In most cases it completely failed to identify the location of the error and tended to also not be sufficiently accurate for the criteria used to choose between h- and p-refinement. It could perhaps be more usable, if it was instead used for local order reduction, rather than increase, based on how the local order drop through L^2 proved to be quite satisfactory in most cases.

Chapter 8

Conclusions

This chapter summarizes all the results obtained in this thesis. First the research questions are restated, followed by a section trying surmise the answers to each of them. The section is then concluded by a reflection on the work done, shortcomings discovered during the work, recommendations for future work that could be done to overcome these shortcomings, and remaining questions remaining in this topic.

8.1 Research Questions

In the beginning of the thesis, the following research questions were posed based on the knowledge gaps in the field of study:

1. How can *hp*-refinement be formulated in the MSEM context,
2. How to choose between *h*-refinement and *p*-refinement,
3. How do different error estimators compare with VMS.

These were all answered to sufficient degree, while also revealing some new research gaps that should be investigated in future work.

8.2 Formulating *hp*-Refinement in MSEM

To support incorporating *hp*-refinement into MSEM, first the formulation for connecting elements of differing polynomial orders was derived. It was based on constraining the boundary of the higher order element to that of the lower order element. This allowed for continuity of 0-forms and 1-forms to be enforced.

To deal with *h*-refinement part, first the method was restricted to only work by splitting a (parent) element into four (child) elements. From there, a recursive algorithm for connecting subdivided boundaries was devised. Each pair of boundaries would be checked for having at least one level of subdivision. If neither was subdivided, this was a case that could be handled by *p*-refinement. If one had subdivision(s) while other did not, the subdivided boundary would be merged into an artificial boundary which could once again be handled with *p*-refinement. If both had at least one level of subdivisions, this process could be recursively applied to each half.

The way *h*-refinement was implemented in this work is however not the only way it could be done. The method might be more applicable if instead of dividing the 2D element into four pieces, it would only ever half an element in different directions. That way, it might be more usable for 3D cases, though it would require more book keeping.

Another question, which applies to both *h*-refinement and *p*-refinement is that if there is a benefit to representing boundaries of elements at a lower order than they really are when enforcing continuity using Lagrange multipliers. This would introduce a trade off between the global system's size and accuracy, which might be beneficial for some cases, at least when applied locally. This might for example allow for some interesting applications, such as having a preconditioner for the system or a guess for the initial solution, based on the original system but with much weaker continuity requirements.

8.3 Choosing Between h-Refinement and p-Refinement

After the refinement technique was mathematically sound and implemented, the next step was to find a way to decide between h-refinement and p-refinement in elements which were to be refined. It was decided that p-refinement would be the default choice, since an element with order $p + 1$ is in almost all but most extreme cases be more efficient in terms of error reduction per number of degrees of freedom, when compared to four elements with order $\lfloor \frac{p+1}{2} \rfloor$ (which is roughly equivalent outcome for h-refinement).

As such, the question of choice between h- and p-refinement became the question of when would h-refinement be worth it. Assuming the error would be perfectly known and expressed with Legendre basis, the change in the square of the error's L^2 norm due to order change from q to p could be obtained in one dimension, as per Equation 8.1. This could be approximated by ignoring terms involving e^p, which could be rewritten into an approximate form in Equation 8.2. This could also be extended to higher dimensions by correcting the scaling of the integrals for Legendre polynomial basis.

$$||e_p||_{L^2} - ||e_q||_{L^2} = \frac{\Delta x}{4} \left(\sum_{i=0}^{p} \frac{2}{2i+1} \left(({e^p}_i)^2 - ({e^q}_i)^2 \right) + \sum_{i=p+1}^{q} \frac{2}{2i+1} \left({u_i}^2 - ({e^q}_i)^2 \right) \right) \tag{8.1}$$

$$\Delta\varepsilon_{L^2} \approx \frac{\Delta x}{4} \sum_{i=p+1}^{q} \frac{2{u^q}_i \left({u^q}_i + 2{e^q}_i \right)}{2i+1} \tag{8.2}$$

Based on this estimate, it was possible to set a threshold at which it would be judged acceptable to incur the cost of an h-refinement for a single refinement iteration. This was left as a hype-parameter, which could be tweaked on per-case basis, but for numerical experiments performed, a fixed value for the ratio between the error increase estimate and total error of the element estimate proved effective.

8.4 Comparison of Different Error Estimators with VMS

As the last part of the thesis work, a few different error estimators were compared, most notably including VMS. Since VMS was not implemented in the solver itself, the results it would provide were imitated based on the theoretical values. The error estimators based on the following were used:

- Exact error, used as baseline,
- Difference between the solution on a coarse and fine mesh,
- Recovering unresolved scales from VMS,
- Computing the difference between the solution and its coarser projection,
- Locally inverting the residual-error system on a finer mesh without continuity using bubble functions.

These are ordered in terms of reliability, though VMS- and finer mesh-based performed almost the same. It should be noted that for problems based on a symmetrical PDE the two are exactly equivalent, as such the choice between the two can not be made based on relative performance as error estimators.

The worst of the considered error estimators was most likely the worst due to it ignoring continuity and attempting to increase the order. If it instead retained some continuity, either locally or globally, it would likely have performed much better.

8.5 Reflection and Recommendation

During the research done in this thesis, quite some new questions and research gaps arose. First which arose during the integration of hp-refinement into MSEM was the question of how would using lower order of continuity for neighboring elements trade off numerical accuracy for smaller global system size. This could allow for a kind of an iterative solver pre-conditioner, where the approximation of the inverse would be based on low order continuity, which would in turn make it a very cheap approximation of the system inverse, which could be obtained using Schur's compliment.

Following that, a question arose on how effective would it be to employ uni-directional h-refinement, as currently the refinement was performed based on splitting an element into four children. It could instead be divided into uni-directional refinement and grid-like refinement instead, each of which might be more beneficial in different situations.

On the question of choice between h-refinement and p-refinement, quite some approximations were made. First of all, it was assumed that the error increase due to order change that would happen with h-refinement, there would be no improvement due to increasing the element count from one to four. This was a conservative choice, but was then countered by an optimistic approximation coming from the assumption that the error would remain the same for the lower order components. This is not true for any differential operator, which is not the L^2 projector, in which case they are, and stay zero. It would be beneficial to examine how significant these two effects are in practice.

Another part of error estimation which was entirely ignored was the effect of changing orders of neighboring elements so that they are no longer equal, or what effect there is when an element has to be divided into lower order elements. These questions might lead to a more adequate refinement criterion, as results from the third test in Figure 8.1 show that having more accurate error estimators lead to worse results.

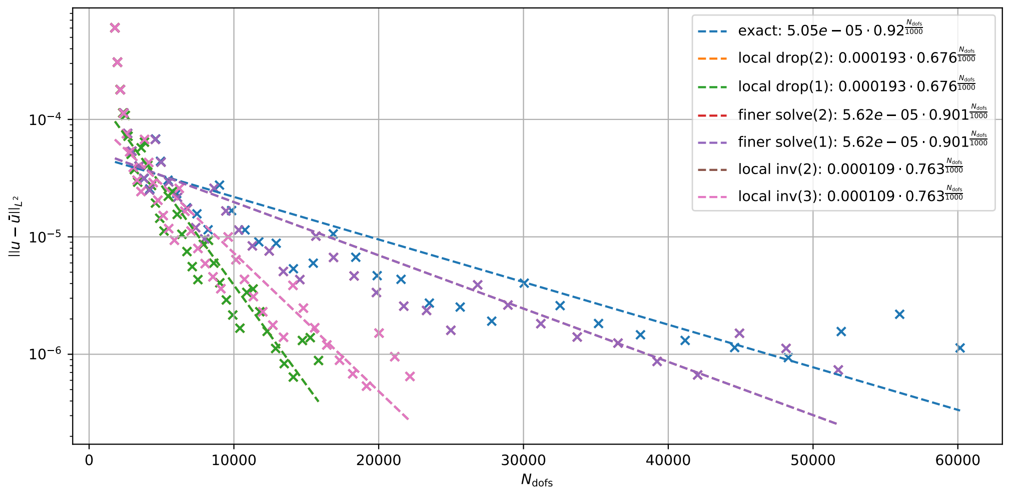


Figure 8.1: Convergence of the solution for iteratively refined meshes in the L^2 norm using different error estimators.

Another aspect that could be further examined comes from approximating error estimators. The most successful error estimators in test other than the third one, unsurprisingly showed the most accurate error estimators to be the most efficient. Those are however also the most expensive. As such, it would be beneficial if it was possible to approximate them. This could be done especially for VMS, where some parts of the system might be possible to further approximate and potentially even make the estimator a local operation.

On the topic of error estimators, the one based on using bubble functions for local inversion, which proved to be the worst performing of the group overall, could likely be adapted into a much better performing one by retaining some order of continuity when approximating the inverse of the system. This would bring in a trade-off between locality and accuracy, which might be interesting to look at.

It would of course also be interesting to integrate these refinement methods into an unsteady solver, or perhaps even a mimetic space-time solver, where it might be possible to perform local p-refinement for the time basis, locally increasing accuracy of time marches in a specific region. This would of course also raise of question of how would the solver perform if p-refinement would perform if it could also be done uni-directionally.

References

[1] "Anysis Fluent User Guide: Chapter 38: Adapting the Mesh," , 2025. URL https://ansyshelp.ansys.com/public/////Views/Secured/corp/v251/en/flu_ug/flu_ug_chp_adapt.html.

[2] Shi, L., and Wang, Z. J., "Adjoint-based error estimation and mesh adaptation for the correction procedure via reconstruction method," *Journal of Computational Physics*, Vol. 295, 2015, pp. 261–284. 10.1016/j.jcp.2015.04.011, URL https://www.sciencedirect.com/science/article/pii/S0021999115002570.

[3] Balasubramanian, R., and Newman, J. C., "Adjoint-based error estimation and grid adaptation for functional outputs: Application to two-dimensional, inviscid, incompressible flows," *Computers & Fluids*, Vol. 38, No. 2, 2009, pp. 320–332. 10.1016/j.compfluid.2008.04.003, URL https://www.sciencedirect.com/science/article/pii/S004579300800087X.

[4] Hughes, T. J. R., and Sangalli, G., "Variational Multiscale Analysis: the Fine-scale Green's Function, Projection, Optimization, Localization, and Stabilized Methods," *SIAM Journal on Numerical Analysis*, Vol. 45, No. 2, 2007, pp. 539–557. 10.1137/050645646, URL https://epubs.siam.org/doi/10.1137/050645646, publisher: Society for Industrial and Applied Mathematics.

[5] Shrestha, S., Dekker, J., Gerritsma, M. I., Hulshoff, S. J., and Akkerman, I., "Construction and application of an algebraic dual basis and the Fine-Scale Greens' Function for computing projections and reconstructing unresolved scales," *Computer Methods in Applied Mechanics and Engineering*, 2024. 10.1016/j.cma.2024.116833, URL https://resolver.tudelft.nl/uuid:e731427f-24d4-4e2d-a24a-4dc46ebf7972.

[6] Kuystermans, P., "Mimetic Mesh Refinement," Master's thesis, TU Delft, 2012. URL https://resolver.tudelft.nl/uuid:830611e8-104d-4b56-aede-8515e45be90f.

[7] Shrestha, S., "Hybridised Mimetic Discretisation and Variational Multiscale Theory for Advection-Dominated Problems," Master's thesis, TU Delft, 2022. URL https://resolver.tudelft.nl/uuid:afb6f8c2-58e7-42d6-b765-de30bad8e302.

[8] Jain, V., Zhang, Y., Palha, A., and Gerritsma, M., "Construction and application of algebraic dual polynomial representations for finite element methods on quadrilateral and hexahedral meshes," , Sep. 2020. 10.48550/arXiv.1712.09472, URL http://arxiv.org/abs/1712.09472, arXiv:1712.09472 [math].

[9] FSF, "The GNU General Public License v3.0 - GNU Project - Free Software Foundation," , 2007. URL https://www.gnu.org/licenses/gpl-3.0.en.html.

[10] Babuška, I., Kellogg, R. B., and Pitkäranta, J., "Direct and inverse error estimates for finite elements with mesh refinements," *Numerische Mathematik*, Vol. 33, No. 4, 1979, pp. 447–471. 10.1007/BF01399326, URL http://link.springer.com/10.1007/BF01399326.

[11] Babuška, I., and Suri, M., "The $h-p$ version of the finite element method with quasiuniform meshes," *ESAIM: Modélisation mathématique et analyse numérique*, Vol. 21, No. 2, 1987, pp. 199–238. URL https://www.numdam.org/item/?id=M2AN_1987__21_2_199_0.

[12] Irigaray, O., Ansa, Z., Fernandez-Gamiz, U., Larrinaga, A., García-Fernandez, R., and Portal-Porras, K., "Adaptive mesh refinement (AMR) criteria comparison for the DrivAer model," *Heliyon*, Vol. 10, No. 11, 2024, p. e31966. 10.1016/j.heliyon.2024.e31966, URL https://www.sciencedirect.com/science/article/pii/S2405844024079970.

[13] Shi, L., Zhang, C., and Wen, C.-Y., "Adaptive mesh refinement algorithm for CESE schemes on unstructured quadrilateral meshes," *Computer Physics Communications*, Vol. 311, 2025, p. 109565. 10.1016/j.cpc.2025.109565, URL https://www.sciencedirect.com/science/article/pii/S0010465525000682.

[14] Asghari, M., and Jahanbakhshi, S., "Development of an adaptive mesh refinement technique for field-scale simulation of matrix acidizing in carbonate reservoirs," *Fuel*, Vol. 399, 2025, p. 135645. 10.1016/j.fuel.2025.135645, URL https://www.sciencedirect.com/science/article/pii/S0016236125013705.

[15] Ainsworth, M., and Oden, J. T., "A posteriori error estimation in finite element analysis," *Computer Methods in Applied Mechanics and Engineering*, Vol. 142, No. 1, 1997, pp. 1–88. 10.1016/S0045-7825(96)01107-3, URL https://www.sciencedirect.com/science/article/pii/S0045782596011073.

[16] Demkowicz, L., Rachowicz, W., and Devloo, P., "A Fully Automatic hp-Adaptivity," *Journal of Scientific Computing*, Vol. 17, No. 1, 2002, pp. 117–142. 10.1023/A:1015192312705, URL https://doi.org/10.1023/A:1015192312705.

[17] Demkowicz, L., *Computing with hp-ADAPTIVE FINITE ELEMENTS: Volume 1 One and Two Dimensional Elliptic and Maxwell Problems*, Chapman and Hall/CRC, New York, 2006. 10.1201/9781420011685.

[18] Demkowicz, L., Kurtz, J., Pardo, D., Paszynski, M., Rachowicz, W., and Zdunek, A., *Computing with hp-ADAPTIVE FINITE ELEMENTS: Volume II Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*, Chapman and Hall/CRC, New York, 2007. 10.1201/9781420011692.

[19] Houston, P., Senior, B., and Süli, E., "Sobolev regularity estimation for hp-adaptive finite element methods," *Numerical Mathematics and Advanced Applications*, edited by F. Brezzi, A. Buffa, S. Corsaro, and A. Murli, Springer Milan, Milano, 2003, pp. 631–656. 10.1007/978-88-470-2089-4_58.

[20] Hughes, T. J. R., "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 127, No. 1, 1995, pp. 387–401. 10.1016/0045-7825(95)00844-9, URL https://www.sciencedirect.com/science/article/pii/0045782595008449.

[21] Sun, B., Gao, L., Wang, K., Ye, J., Xuan, J., Wu, Y., and Ou, J., "A practical multi-scale simulation framework for high-cycle fatigue analysis of wind turbines," *Ocean Engineering*, Vol. 335, 2025, p. 121699. 10.1016/j.oceaneng.2025.121699, URL https://www.sciencedirect.com/science/article/pii/S0029801825014052.

[22] Zhou, C., Wang, X., Ma, C., Lu, S., Zhang, L., Chen, Y., Xiong, H., Zhang, W., Liu, S., and Zhang, R., "The multi-scale mechanical properties of double-double layup technology in Type IV fiber-wound pressure vessels," *International Journal of Pressure Vessels and Piping*, Vol. 217, 2025, p. 105539. 10.1016/j.ijpvp.2025.105539, URL https://www.sciencedirect.com/science/article/pii/S0308016125001097.

[23] Li, J., Mao, M., Zhang, Z., Zhang, F., He, F., Shi, J., Liu, Y., Wang, Y., Li, C., and Gao, H., "Numerical study of flame tilt instability during filtration combustion with reverse flow and extra lean methane/air mixture through a multi-scale model," *Applied Thermal Engineering*, Vol. 276, 2025, p. 126882. 10.1016/j.applthermaleng.2025.126882, URL https://www.sciencedirect.com/science/article/pii/S1359431125014747.

[24] Geers, M. G. D., Kouznetsova, V. G., and Brekelmans, W. A. M., "Multi-scale computational homogenization: Trends and challenges," *Journal of Computational and Applied Mathematics*, Vol. 234, No. 7, 2010, pp. 2175–2182. 10.1016/j.cam.2009.08.077, URL https://www.sciencedirect.com/science/article/pii/S0377042709005536.

[25] Shrestha, S., Gerritsma, M., Rubio, G., Hulshoff, S., and Ferrer, E., "Optimal solutions employing an algebraic Variational Multiscale approach part I: Steady Linear Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 438, 2025, p. 117832. 10.1016/j.cma.2025.117832, URL https://www.sciencedirect.com/science/article/pii/S0045782525001045.

[26] Arnold, D., Falk, R., and Winther, R., "Finite element exterior calculus: from Hodge theory to numerical stability," *Bulletin of the American Mathematical Society*, Vol. 47, No. 2, 2010, pp. 281–354. 10.1090/S0273-0979-10-01278-4, URL https://www.ams.org/bull/2010-47-02/S0273-0979-10-01278-4/.

[27] Sharma, H., Patil, M., and Woolsey, C., "A review of structure-preserving numerical methods for engineering applications," *Computer Methods in Applied Mechanics and Engineering*, Vol. 366, 2020, p. 113067. 10.1016/j.cma.2020.113067, URL https://linkinghub.elsevier.com/retrieve/pii/S0045782520302516.

[28] Kreeft, J., Palha, A., and Gerritsma, M., "Mimetic framework on curvilinear quadrilaterals of arbitrary order," , Nov. 2011. 10.48550/arXiv.1111.4304, URL http://arxiv.org/abs/1111.4304, arXiv:1111.4304 [math].

[29] Gerritsma, M., "An Introduction to a Compatible Spectral Discretization Method," *Mechanics of Advanced Materials and Structures*, Vol. 19, No. 1-3, 2012, pp. 48–67. 10.1080/15376494.2011.572237, URL https://doi.org/10.1080/15376494.2011.572237, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15376494.2011.572237.

[30] Palha, A., "High order mimetic discretization; development and application to Laplace and advection problems in arbitrary quadrilaterals," Ph.D. thesis, TU Delft, 2013. 10.4233/uuid:300360b9-e534-4a45-8fd7-9fa8beebd0ef, URL https://repository.tudelft.nl/record/uuid:300360b9-e534-4a45-8fd7-9fa8beebd0ef.

[31] Gerritsma, M., "Mimetic spectral element methods," *AIP Conference Proceedings*, Vol. 1648, No. 1, 2015, p. 700002. 10.1063/1.4912923, URL https://doi.org/10.1063/1.4912923.

[32] Gerritsma, M., Palha, A., Jain, V., and Zhang, Y., "Mimetic Spectral Element Method for Anisotropic Diffusion," , Feb. 2018. URL https://arxiv.org/abs/1802.04597v1.

[33] Zhang, Y., Palha, A., Gerritsma, M., and Rebholz, L. G., "A mass-, kinetic energy- and helicity-conserving mimetic dual-field discretization for three-dimensional incompressible Navier-Stokes equations, part I: Periodic domains," *Journal of Computational Physics*, Vol. 451, 2022, p. 110868. 10.1016/j.jcp.2021.110868, URL https://www.sciencedirect.com/science/article/pii/S0021999121007634.

[34] Jain, V., Palha, A., and Gerritsma, M., "Use of algebraic dual spaces in domain decomposition methods for Darcy flow in 3D domains," *Computer Methods in Applied Mechanics and Engineering*, Vol. 404, 2023, p. 115827. 10.1016/j.cma.2022.115827, URL https://www.sciencedirect.com/science/article/pii/S0045782522007836.

[35] Tu, L. W., *An Introduction to Manifolds*, Universitext, Springer, New York, NY, 2011. 10.1007/978-1-4419-7400-6, URL https://link.springer.com/10.1007/978-1-4419-7400-6.

[36] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Van Der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., Van Mulbregt, P., SciPy 1.0 Contributors, Vijaykumar, A., Bardelli, A. P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C. N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D. A., Hagen, D. R., Pasechnik, D. V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G. A., Ingold, G.-L., Allen, G. E., Lee, G. R., Audren, H., Probst, I., Dietrich, J. P., Silterra, J., Webber, J. T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J. L., De Miranda Cardoso, J. V., Reimer, J., Harrington, J., Rodríguez, J. L. C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N. J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P. A., Lee, P., McGibbon, R. T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T. J., Robitaille, T. P., Spura, T., Jones, T. R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y. O., and Vázquez-Baeza, Y., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, Vol. 17, No. 3, 2020, pp. 261–272. 10.1038/s41592-019-0686-2, URL https://www.nature.com/articles/s41592-019-0686-2.

[37] Davis, T. A., "Algorithm 832: UMFPACK V4.3 - an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, Vol. 30, No. 2, 2004, pp. 196–199. 10.1145/992200.992206, URL https://doi.org/10.1145/992200.992206.