Improving Efficiency in the Discontinuous Galerkin Spectral Element Method for Complex Geometry Physics

Amit Nayak

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the

Master of Applied Science

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

© Amit Nayak, Ottawa, Canada, 2025

Abstract

The discontinuous Galerkin spectral element method (DGSEM) is a numerical scheme that has risen in popularity, largely due to its high-order accuracy and geometrical flexibility. It is well suited for convection dominated problems such as high-speed flows and wave propagation. Despite these benefits, it still suffers from the drawback of a high computational cost. In this thesis, we aim to model physics in complex geometries using DGSEM with a focus on efficiency. We report on four different approaches to improve efficiency.

First, we use a transfinite mapping to extend a Cartesian grid DGSEM solver for the acoustic wave equation. This code possesses adaptive mesh refinement (AMR) capabilities through both h-refinement where the elements are split, and p-refinement where the polynomial order of the element is increased. hp-Adaptivity increases the capabilities of simulating complex physics as resolution can be provided in select regions of the domain where necessary. To address the load imbalance caused by AMR in parallel computing, the Cartesian code is dynamically load balanced and was shown to scale well in a high performance computing (HPC) environment. We were able to successfully model the acoustic wave equation in complex geometries using transfinite mapping while retaining the high accuracy of the method. We tested the scaling performance of the code after applying the transfinite mapping and compared to the Cartesian grid performance. The transfinite mapped code scales relatively well but does incur some computational overhead that affects the scaling. Based on the results, we pursued the immersed boundary method as an alternative approach to model complex geometries.

The immersed boundary method (IBM) is a technique where an object is modeled,

without conforming to the computational grid. We specifically use the volume penalty method to model objects as porous obstacles for acoustic wave propagation problems. The IBM promises time savings for the end user by simplifying the mesh generation process, as only Cartesian grids are required. However, it does come with the tradeoff of a loss in accuracy and oscillations. We address these issues with hp-adaptivity and a low porosity parameter, where we observed an increase in accuracy and a reduction in oscillatory behavior. In addition to the time savings during the mesh generation process, computational savings can be obtained as the metrics associated with the transfinite mapping are not required.

Next, we explore the half-closed discontinuous Galerkin (HCDG) method to model incompressible flows. In the HCDG method, "half-closed" nodes are used which allow for more efficient operator construction and good sparsity patterns. Complex geometries are represented using unstructured grids. We successfully simulate the incompressible Navier-Stokes equations for a variety of problems at low Reynolds numbers.

Finally, we improve efficiency by developing a new "eliminated" linear solver. We use static condensation to eliminate degrees of freedom based on the local discontinuous Galerkin (LDG) switch function, and apply p-multigrid on the eliminated linear system. We apply this eliminated p-multigrid solver on elliptic problems and the incompressible Navier-Stokes equations for both structured and unstructured grids. We observed significant reductions in the number of degrees of freedom and number of non-zero entries for such problems using our elimination procedure. In addition to the benefits of operating on the smaller linear systems, we also observed reductions in the number of V-multigrid-cycles required for convergence, up to a factor of two for elliptic problems and a 20% reduction for the incompressible Navier-Stokes equations.

***Keywords*:** Discontinuous Galerkin Spectral Element Method, hp-Adaptivity, Transfinite Mapping, Immersed Boundary Method, Wave Propagation, Half-Closed Discontinuous Galerkin, Static Condensation, p-Multigrid, Incompressible Fluid Flow.

Acknowledgments

First, I would like to extend my deepest gratitude towards my thesis supervisor Professor Catherine Mavriplis. I truly appreciate all the mentorship and guidance you have provided me over the years. I am especially grateful for all the enormous patience you have given me. Thank you also for caring about both my academic and my professional development. I would also like to express gratitude for all the amazing research opportunities you have presented me. It has been a long and enjoyable journey, and I am very proud to call myself your student.

Thanks are also due to the other members in the research group (in alphabetical order): Aktham, Bharathwaj, Ming, and Saleh. This journey would not have been as fun without you guys. Thank you for all the great memories. I hope we all stay good friends for a long time. I would also like to take the time to thank my other office mates (in alphabetical order): Blake, Kazem, Payam, and Shahin. It has been amazing hanging out with you guys and I wish you all the best in the future.

It should be mentioned that a significant portion of my thesis was conducted during a summer research visit at Lawrence Berkeley National Lab (LBL) under the supervision of Professor Per-Olof Persson and Yulong (Lewis) Pan. Thank you Per for hosting me over the summer, it was definitely an amazing experience. I truly appreciate your mentorship, kindness and generosity. Thank you Lewis as well for your mentorship and the time you spent with me. I deeply enjoy our conversations, both about research or anything else. I look forward to any future research endeavors with either of you.

I would also like to thank the others at LBL who made my experience extremely enjoyable (in alphabetical order): Dinesh, Eric, Jeff, Mark, Rob, and Zixi. I always felt

enriched from our daily interactions. I hope you all take care of the coffee machine and continue to elevate your coffee experiences.

The biggest thanks goes out to my family including my mom, dad, sister and bird. Thank you so much for always encouraging me to do my best and for always having my back. I am very appreciative of all the advice and support you have given me during this journey.

Finally, I would like to acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Digital Research Alliance of Canada, and the University of Ottawa.

List of Acronyms

AMR	Adaptive Mesh Refinement
CEM	Computational Electromagnetics
CFD	Computational Fluid Dynamics
CSM	Computational Solid Mechanics
DG	Discontinuous Galerkin
DGSEM	Discontinuous Galerkin Spectral Element Method
DNS	Direct Numerical Simulation
HCDG	Half-Closed Discontinuous Galerkin
HPC	High Performance Computing
IBM	Immersed Boundary Method
LDG	Local Discontinuous Galerkin
LES	Large Eddy Simulation
MDAO	Multidisciplinary Design and Optimization
PDE	Partial Differential Equation
RANS	Reynolds Averaged Navier-Stokes
SEM	Spectral Element Method

Contents

Chapter 1

Introduction

The motivation for this thesis is computational science and engineering or scientific computing. Due to the rapid rise in computing power over the years, there has been an increased reliance on simulation for engineering design. Simulation techniques provide an alternative when experiments are too costly to perform. Furthermore, theoretical techniques can often be too difficult to apply to real world problems. Thus, simulations have become an important tool for engineers. However, they do not replace the aforementioned methods, rather they complement them, for example, by developing new physical models or by reducing the number of experiments required. Many industries which involve mechanical engineering heavily utilize simulation technology during the design process such as automotive, biomedical, construction, and energy. Therefore, it is crucial to continue to improve the field as it will have a widespread societal impact. Despite the prevalence of scientific computing, there is still much more research to be done to improve the accuracy and robustness of the methods in addition to ensuring the methods effectively use high performance computing (HPC) resources. The research in this thesis attempts to explore and improve the underlying numerical methods or "numerics" for solving partial differential equations (PDEs), specifically for mechanical engineering applications. Some examples of disciplines which numerically solve PDEs include computational electromagnetics (CEM) [1], computational solid mechanics (CSM) [2], computational fluid dynamics (CFD) [3], and multidisciplinary design

and optimization (MDAO) [4].

To provide further motivation specifically for CFD, we highlight NASA's CFD Vision 2030 report [5]. Although the report acknowledges the prevalence of CFD in the field of aeronautics, many more improvements to CFD remain to be made. One challenging problem in computational aerodynamics is the accurate prediction of turbulent flow separation. Thus, a much desired improvement to the field is the increased fidelity of turbulent flow simulations. The current state of the art in industry uses turbulence models such as the Reynolds-averaged Navier-Stokes (RANS) equations or large eddy simulation (LES). In these turbulence models, some, if not all, the spatial and temporal scales of turbulence are modeled rather than resolved. The ultimate goal is to use direct numerical simulation (DNS), where the full range of spatial and temporal scales of turbulence are resolved, not modeled. To make DNS practical, much more research must be conducted in terms of algorithmic development for CFD. This is extremely important as aeronautics is dominated by turbulent flows. To demonstrate this visually, an example of a DNS simulation over an aircraft wing in takeoff position is presented in Figure 1.1. We observe important fluid mechanics phenomena associated with turbulence such as turbulent flow separation and hairpin vortices. Even outside the field of aeronautics, there is a wide range of engineering applications including automotive/motorsport [6], combustion [7], energy [8], and marine [9] to name a few, that are dominated by turbulent flows. Thus, improvements made to the simulation of turbulent flows will have widespread societal impact as it will enable the design of more fuel efficient, economical, robust and safe engineering devices.

An approach to improved fidelity in numerical simulations is the use of high-order methods. For the sake of clarity, a numerical method is said to be o^{th} order accurate when the error ϵ is proportional to: $\epsilon \propto h^o$, where h is the mesh size. To aid in the discussion of high-order methods with regards to CFD, we refer to the findings in the paper by Wang et. al [11]. First, in the aerospace sector, the industrial standard for CFD is second-order finite volume schemes. However, it is generally believed that

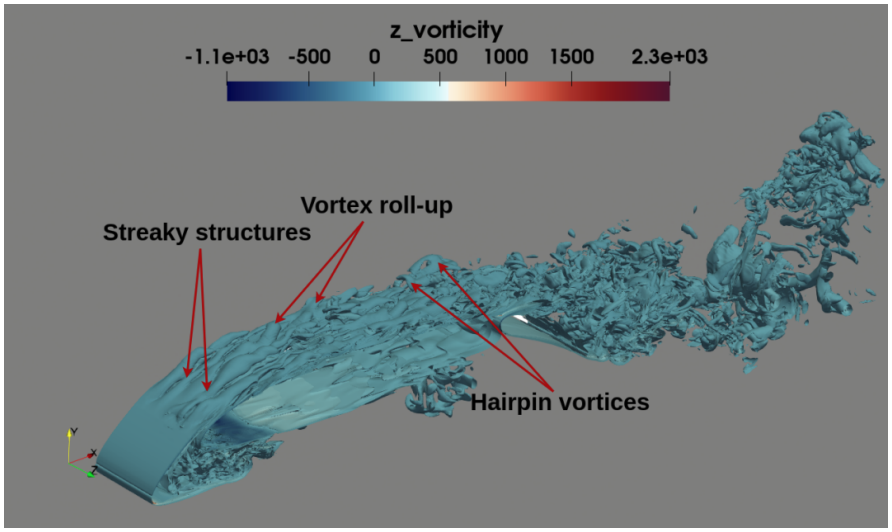


Figure 1.1: Direct numerical simulation over the McDonnell Douglas 30P30N wing at $\mathrm{Re}_c = 1.83 \times 10^4$. Reprinted from [10].

third-order and greater are considered by some as high-order. There has been extensive research conducted on high-order methods producing schemes such as the: spectral element method (SEM) [12], compact finite differences [13], spectral volumes [14], wavelet collocation [15], and discontinuous Galerkin (DG) [16, 17] to list a few. Each has its own advantages and disadvantages with respect to each other. However, high-order methods have the following advantage. High-order methods have better convergence properties. If a high degree of accuracy is desired, fewer mesh refinements are required to achieve this degree of accuracy when compared to low-order methods. An example of this is presented in Figure 1.2. Comparing the third-order and fourth-order error curves, we observe only one element size refinement (moving right to left in the plot) is required for the fourth-order approximation whereas the third-order approximation requires three refinements to achieve the same error. Furthermore, for a given element size (moving vertically in the plot), we obtain an order of magnitude reduction in error as we increase the order of approximation.

Thus, high-order methods are more efficient for problems that require a high level of accuracy. Furthermore, there are some applications that necessitate the use of high-order methods such as unsteady vortex dominated flows and wave propagation problems. Low-order methods are often too dissipative to sufficiently resolve unsteady

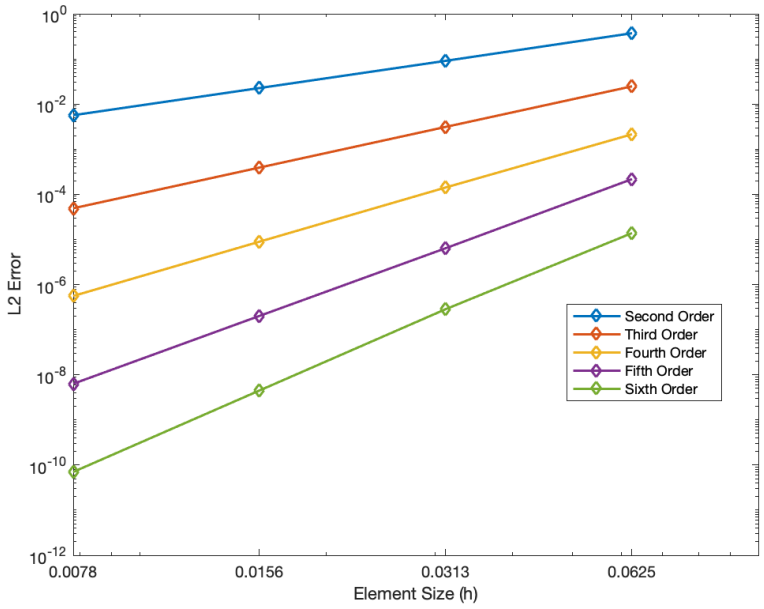


Figure 1.2: h-Convergence test for $-\nabla^2 u = f$ with different orders of convergence plotted using the local discontinuous Galerkin discretization. Solution is $u = e^{sin(4\pi x)} - 1$, with domain $x \in [0, 1]$, with Dirichlet boundary conditions $x(0) = x(1) = 0$.

vortical structures or possess too much dispersive error to properly model wave propagation for long distances. Thus, for aeroacoustic simulations, high-order methods are extremely prevalent as they are able to sufficiently resolve the propagation of waves. Furthermore, high-order methods are very popular for LES and DNS simulations as they are able to capture unsteady vortices in a flow field without excess dissipative errors.

Despite the promising advantages of high-order methods, they do possess their drawbacks [11]. While high-order methods produce higher accuracy, they are more computationally expensive. For the same mesh, low order methods are computationally cheaper to run. Additionally, they are not as robust as their low-order counterparts. High-order methods can be more sensitive to shocks and under-resolution. One way to address these disadvantages is by using adaptive mesh refinement (AMR). By utilizing AMR, additional resolution can be provided specifically to regions of the domain with more complicated dynamics such as a boundary layer or a vortex. An example of AMR used for this purpose in an aerodynamic simulation is presented in Figure 1.3. This

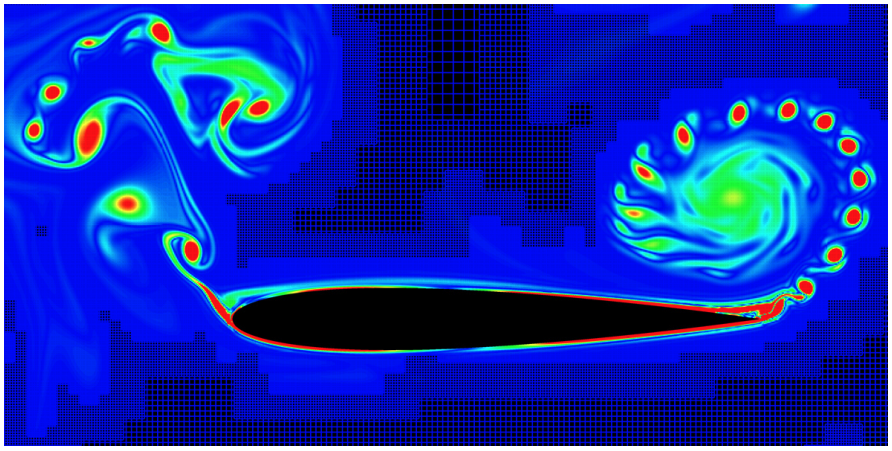


Figure 1.3: Adaptive mesh refinement used in a simulation over a plunging airfoil (vorticity plot). Extra resolution provided to sufficiently resolve boundary layer and vortices. Reprinted from [18].

helps mitigate the cost of high-order methods as it avoids uniformly refining the mesh, potentially providing extra resolution where it is not required. Furthermore, AMR helps address the robustness issue by avoiding under-resolution of complex features in the solution. AMR also allows engineers to save time performing numerous mesh design iterations. By running a simulation using AMR, it can produce a grid that has appropriate refinement, without a priori knowledge of the solution. The engineer can use this adapted grid as a starting grid for a higher-fidelity simulation.

1.1 Numerical Methods

In this thesis, we focus on the discontinuous Galerkin spectral element method (DGSEM) [19]. In the SEM, the domain is discretized into many elements and the solution is represented as a weighted combination of high-order orthogonal polynomials. For example, we approximate a function $f(x)$ as a weighted combination of orthogonal basis functions:

$$
\begin{aligned}
& f(x) = \sum_{n=0}^{\infty} a_n \phi_n(x), \\
\implies & f(x) = \sum_{n=0}^{N} a_n \phi_n(x) + \sum_{n=N+1}^{\infty} a_n \phi_n(x), \\
\implies & f(x) = \sum_{n=0}^{N} a_n \phi_n(x) + \tau, \\
\implies & f(x) \approx \sum_{n=0}^{N} a_n \phi_n(x),
\end{aligned}
\tag{1.1}
$$

where ϕ_n are the orthogonal basis functions and a_n are the coefficients. We approximate up to $N+1$ terms where N is the polynomial order, thus incurring a truncation error τ. In this thesis, Legendre polynomials are employed as the orthogonal basis functions. A sample plot of the first six Legendre polynomials is presented in Figure 1.4.

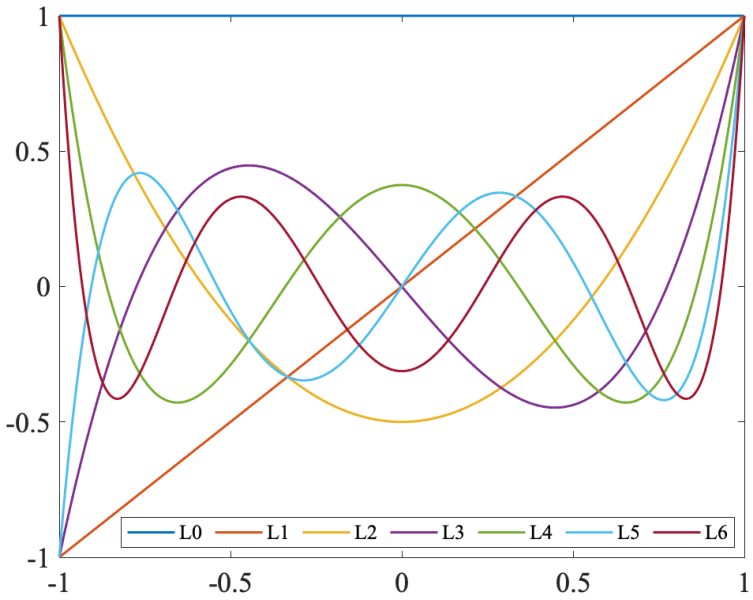


Figure 1.4: Legendre polynomials defined from $x \in [-1, 1]$. Zeroth order up to sixth order polynomials are plotted.

Exponential convergence to smooth solutions is obtained by increasing the polynomial order of the approximation. In the SEM, continuity is enforced between elements, making it well suited for elliptic problems. However, in the DGSEM, discontinuities are

permitted at the boundaries between elements and are stabilized by numerical fluxes. This gives the method a local stencil, suitable for parallel computing. Furthermore, the use of a discontinuous function space is well suited for adaptive mesh refinement, since adaptivity usually incurs nonconformities. Finally, stabilizing with numerical fluxes is advantageous for convection dominated problems such as wave propagation. Despite the attractive features of the DGSEM, it still possesses the typical issues of high-order methods such as the high computational cost, which hinders adoption for use in industrial applications.

1.2 Governing Equations

As mentioned previously, the ultimate goal is to simulate the full compressible Navier-Stokes equations without the use of turbulence models or in other words DNS. However, in this thesis, model PDEs are used to improve upon the numerics, by testing and understanding the behavior of the scheme on each component of the equations. The improved numerics will then be applied to the compressible Navier-Stokes equations in the future. The first PDE considered in this thesis is the two-dimensional acoustic wave equation

$$P_{tt} - c^2 \nabla^2 P = 0, \tag{1.2}$$

$$u_t = -P_x, \tag{1.3}$$

$$v_t = -P_y, \tag{1.4}$$

where P is the pressure, c is the wave speed, t is time, and u, and v are the velocity components. One of the reasons why we consider the acoustic wave equation is that it is a model PDE for hyperbolic problems encountered in high speed flow. Additionally, wave propagation problems necessitate the use of high-order methods to accurately capture their shape and avoid dissipation and dispersion errors. Finally, acoustics is an

important discipline within the field of mechanical engineering.

The second set of governing equations considered in this thesis is the non-dimensional incompressible Navier-Stokes equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla P + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \boldsymbol{f}, \tag{1.5}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.6}$$

where $\boldsymbol{u}$ is the non-dimensional velocity vector, Re is the Reynolds number and $\boldsymbol{f}$ is the body force vector. The incompressible Navier-Stokes equations provide a high-fidelity model for incompressible fluid flow, applicable to low speed aerodynamics and hydrodynamics. It is a challenging case to solve numerically since the equations are non-linear. In the context of this thesis, it provides an opportunity to model elliptic-parabolic problems and a wide range of physical flows.

1.3 Thesis Objectives

The overall objective of this thesis is to push the DGSEM to model complex geometries, in efficient ways. We work towards this goal in four approaches, exploring directions in both discretization and linear solvers. First, we extend a Cartesian hp-adaptive DGSEM acoustic wave equation code [20] to model curvilinear geometries by using transfinite mapping. We then test its computational performance in a high-performance computing (HPC) environment. We also try to extend the same Cartesian code to model more complicated geometries by using the immersed boundary method (IBM) [21]. The advantage of using the IBM is that the object of interest does not need to conform to the underlying mesh as shown in Figure 1.5. Next, we investigate the properties of using different "half-closed" nodes in the DGSEM to obtain optimal sparsity patterns. This method is known as the half-closed discontinuous Galerkin (HCDG) method [22]. We model complex geometries by using unstructured grids (Figure 1.5) and apply HCDG

to the incompressible Navier-Stokes equations. Finally, we develop a novel linear solver and specifically apply it to HCDG on unstructured grids for elliptic problems and the incompressible Navier-Stokes equations.

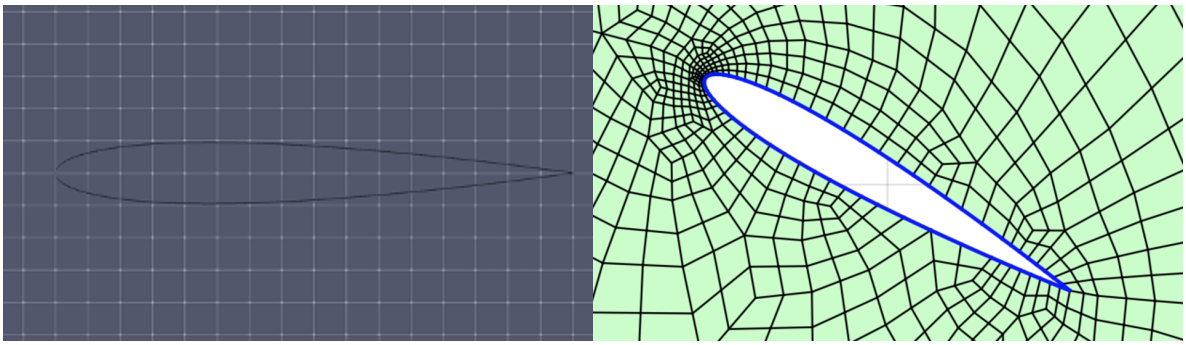


Figure 1.5: Left: NACA 0012 airfoil represented using the immersed boundary method, where object of interest does not need to conform to the Cartesian grid. Right: NACA 0012 airfoil represented with an unstructured grid of body-conforming quadrilateral elements.

1.4 Thesis Organization

This thesis is a collection of four works, in the form of one chapter and three papers.

T1 **Chapter:** A Parallel Transfinite Mapped hp-Adaptive Discontinuous Galerkin Spectral Element Method Acoustic Wave Equation Solver

T2 **Journal paper submitted to Computers and Fluids:** A. Nayak, C. Mavriplis, "Immersed Boundaries in the Discontinuous Galerkin Spectral Element Method through hp-Adaptivity".

T3 **Conference paper published in the proceedings of AIAA SciTech 2025:** A. Nayak, Y. Pan, P.-O. Persson, "Half-Closed Discontinuous Galerkin Methods For Incompressible Flow Problems".

T4 **Journal paper in preparation:** A. Nayak, Y. Pan, P.-O. Persson, "Eliminated p-Multigrid for Half-Closed Discontinuous Galerkin Methods".

The background on the DGSEM formulation and the hp-adaptivity used is presented in T2. Further background on DGSEM can also be found in [23]. Information on the underlying codebase and the dynamic load balancing used for T1 and T2 can be found in Shiqi He's thesis [20]. The formulation for the HCDG is presented in T3, further background on the HCDG formulation and its properties can be found in [22]. Background on multigrid methods is presented in T4.

Chapter 2

A Parallel Transfinite Mapped hp-Adaptive Discontinuous Galerkin Spectral Element Method Acoustic Solver

Contributions

For this chapter, the author implemented the transfinite mapping and performed all the simulations on curved geometries which included the scaling tests on the transfinite mapped code.

Nomenclature

a_i	Covariant basis vector
$\widetilde{a}_n$	Modal coefficients
c	Wave speed
d	Constant
F	Flux
F^*	Numerical flux
J	Jacobian
Ja^i	Contravariant basis vector
k_x, k_y	Wave components
$L_n(x)$	Legendre polynomial
$l_j(x)$	Lagrange interpolating polynomial
N	Number of elements
$\hat{n}^i$	Boundary normal
P	Pressure
p	Polynomial order
$\mathbf{q}$	Vector of conserved variables
scal_i	Scaling factors
t	Time
u, v	Velocity components
w	Barycentric weights
$X_\xi, X_\eta, Y_\xi, Y_\eta$	Metric terms
x, y	Physical space coordinates
$\hat{x}, \hat{y}$	Unit vectors
Γ	Curved boundary
ξ, η	Computational space coordinates
σ	Modal decay rate

2.1 Background

This chapter explores extending a Cartesian grid based discontinuous Galerkin spectral element method (DGSEM) solver to model curvilinear geometries. Shiqi He created a DGSEM code for solving the two-dimensional acoustic wave equation [20]. This code possess adaptive mesh refinement (AMR) capabilities to provide sufficient resolution in the solution where necessary. However, using AMR causes a load imbalance during parallel computing, as some processors may have a much higher computational load than others. Shiqi He addressed this issue by using a space filling curve approach to efficiently re-distribute the workload among the processors. Very promising results were shown when scaling tests were performed on this code. The full details on the dynamic load balancing procedure can be found in [20]. However, despite the computational performance of the code, it is limited to running on Cartesian grids.

In this work, we follow the procedure outlined in [23] to extend the hp-adaptive DGSEM code to curvilinear geometries using a transfinite mapping procedure. Afterwards, we test the effects of the transfinite mapping on the computational performance by performing scaling tests. The scaling tests are then compared to the Cartesian grid scaling tests.

2.2 Discontinuous Galerkin Spectral Element Discretization

We now briefly present the DGSEM discretization used for the wave equation solver. The full derivation can be found in [23]. We start with the two-dimensional acoustic wave equation:

$$\frac{\partial^2 P}{\partial t^2} - c^2(P_{xx} + P_{yy}) = 0, \tag{2.1}$$

$$u_t = -P_x, \tag{2.2}$$

$$v_t = -P_y, \tag{2.3}$$

where P is pressure, u and v are the velocity components, and c is the wave speed. Combining Equations (2.1), (2.2), and (2.3), the following is obtained:

$$\frac{\partial^2 P}{\partial t^2} + c^2 \left((u_x)_t + (v_y)_t \right) = 0. \tag{2.4}$$

We integrate Equation (2.4) with respect to time to obtain a first order PDE

$$P_t + c^2 \left(u_x + v_y \right) = 0. \tag{2.5}$$

Expanding Equation (2.5) in matrix-vector form produces:

$$\begin{bmatrix} P \\ u \\ v \end{bmatrix}_t + \begin{bmatrix} 0 & c^2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P \\ u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & 0 & c^2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P \\ u \\ v \end{bmatrix}_y = 0. \tag{2.6}$$

Since the matrices in Equation (2.6) are constant, they are combined with the derivatives to produce the following expression

$$\mathbf{q}_t + \mathbf{f}_x + \mathbf{g}_y = 0, \tag{2.7}$$

where $\mathbf{q}$ is the vector of conserved variables, $\mathbf{f}_x$ and $\mathbf{g}_y$ are the spatial derivatives of the flux vector $F = \mathbf{f}\hat{x} + \mathbf{g}\hat{y}$. As such, we can write Equation (2.7) as a conservation law to be discretized by a discontinuous Galerkin treatment

$$\mathbf{q}_t + \nabla \cdot F = 0. \tag{2.8}$$

The DGSEM formulation uses the weak form of the PDE:

$$\frac{\partial}{\partial t} \int_V \mathbf{q} dV + \int_V \bigtriangledown \cdot F dV = 0. \tag{2.9}$$

We apply the divergence theorem to obtain the following expression,

$$\frac{\partial}{\partial t}\int_V \mathbf{q} dV = -\int_S F \cdot \hat{n} dS, \tag{2.10}$$

where S is the surface, V is the volume, and $\hat{n}$ is the outward facing normal vector.

To compute the integrals, we use Gauss-Legendre quadrature:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{N} w_i f(x_i), \tag{2.11}$$

where w_i are the weights and $f(x_i)$ is the function evaluated at the Gauss-Legendre nodes, which are the roots of the Nth+1 order Legendre polynomial. This quadrature rule is exact for polynomials of order 2N-1 or less. The Gauss-Legendre nodes are also the solution nodes as shown in Figure 2.1.

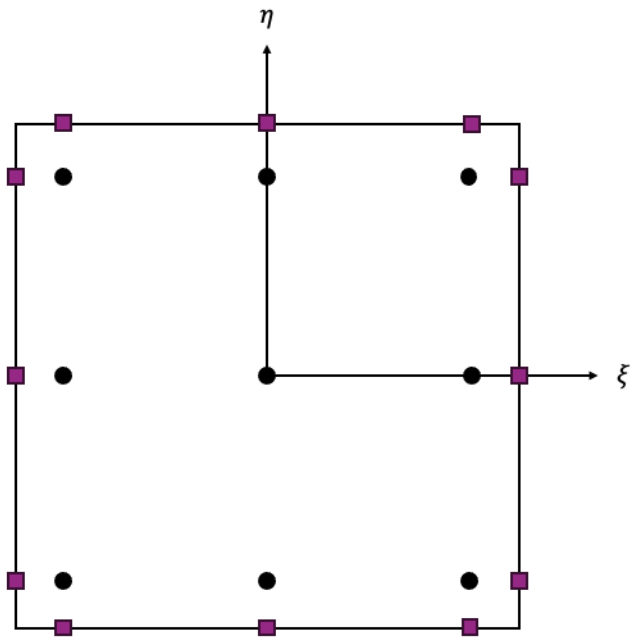


Figure 2.1: Quadrilateral element with polynomial order $p = 2$. Black dots show the location of the Gauss-Legendre nodes and the purple squares show the locations of numerical flux computations.

Note that in Figure 2.1, the nodes are not placed on the boundaries. Thus, an

interpolation is required to calculate the numerical fluxes F^* on the boundaries. The purple squares in Figure 2.1 show the location of the boundary flux calculations.

We use Lagrange (polynomial) interpolation for the interpolation procedures. An interpolant is defined as a polynomial of degree N that interpolates a function f_j at a set of points x_j. The Lagrange form of this polynomial is given by the following expression

$$p_N(x) = \sum_{j=0}^{N} p(x_j) l_j(x), \tag{2.12}$$

where $l_j(x)$ are the Lagrange interpolating polynomials defined by the following expression:

$$l_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{N} \frac{x - x_i}{x_j - x_i}. \tag{2.13}$$

Specifically in our implementation, we use the barycentric formula for the Lagrange interpolation

$$p_N(x) = \frac{\sum_{j=0}^{N} p(x_j) \frac{w_j}{x - x_j}}{\sum_{j=0}^{N} \frac{w_j}{x - x_j}}, \tag{2.14}$$

where w_j are the barycentric weights which are pre-computed. For the transfinite mapping, we approximate the curved boundaries as a polynomial interpolant of the same order as the solution, thus necessitating Lagrange interpolation.

2.3 Adaptive Mesh Refinement

As mentioned previously, the Cartesian grid solver has adaptive mesh refinement capabilities. Two types of mesh refinements are used: h-refinement, where the elements are split, and p-refinement, where the polynomial order of the element is increased. An example of this is shown in Figure 2.2.

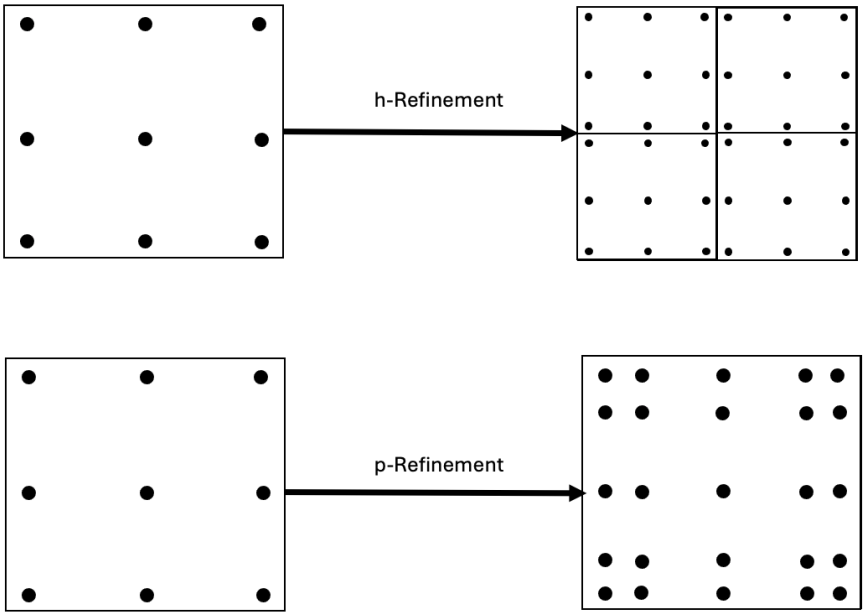


Figure 2.2: Top: h-refinement where element is split into four smaller elements. Bottom: p-refinement where the polynomial order of element is increased from two to four.

For h-refinement, we split an element into four smaller elements. For p-refinement, we increase the polynomial order of the element by two. For coarsening, the reverse of each procedure is done. To determine when to adapt, we use the modal decay error estimator [24]. First, we convert to a modal representation of the solution

$$u(x) \approx \sum_{n=0}^{p} \widetilde{a}_n L_n(x), \tag{2.15}$$

where $\widetilde{a}_n$ are the modal coefficients and L_n is the nth degree Legendre polynomial. In our work, the error estimator is evaluated using the pressure. From the coefficients, the error is estimated as

$$\epsilon_{est} \approx \left(\sum_{n=p+1}^{\infty} \frac{\widetilde{a}_n^2}{\frac{2n+1}{2}} \right)^{1/2}. \tag{2.16}$$

If the estimated error is greater than a user-defined tolerance, we flag the element for refinement. The decision to h- or p-refine is based on a smoothness indicator. First, we fit the modal coefficients to an exponential decay:

$$\widetilde{a}_n \approx Ce^{-\sigma n}. \tag{2.17}$$

When the decay rate $\sigma \leq 1$, the solution is not smooth and h-refinement is used to provide sufficient resolution. However, when the decay rate $\sigma > 1$, the solution is smooth and we use p-refinement to take advantage of the exponential convergence in polynomial order for smooth solutions. The hp-adaptivity procedure introduces both geometric (different element size) and functional (different element polynomial order) non-conforming interfaces between elements. To address the errors induced by these non-conforming interfaces, we utilize the mortar element method to maintain the high-order accuracy of the discretization scheme. The details of the mortar element method are described in Chapter 3 of this thesis. Further details on the mortar element method can be found in [25].

2.4 Transfinite Mapping Procedure

To model curvilinear geometries, a transformation is made from the solution domain to the reference square [23]. The computational domain is a reference square and the solution domain is the physical domain of interest. The derivation of the transfinite mapping procedure is briefly described here, while further details can be found in [23]. First, (ξ, η) will be known as the computational space coordinates and (x, y) are the physical coordinates. The reference square is defined as $\Omega \in$ [-1, 1] $\times$ [-1, 1]. The curvilinear geometries to be mapped are curved-sided quadrilaterals. The following figure shows the mapping of a curved-sided quadrilateral from the reference square.

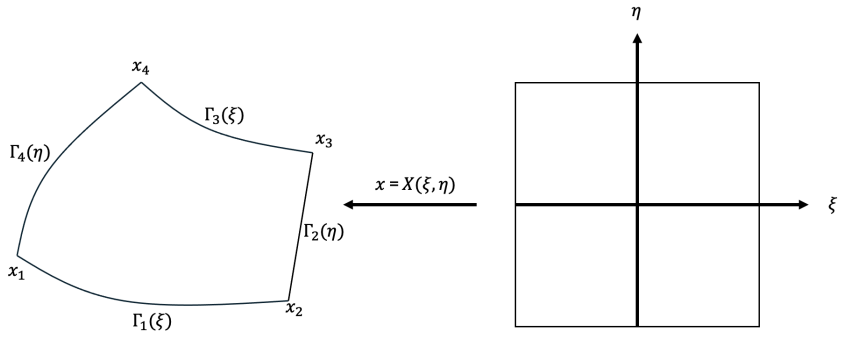


Figure 2.3: Mapping of a curved-sided quadrilateral; Γ_i are the physical domain boundaries.

The boundaries of the physical domain are denoted by Γ_i. We approximate the domain boundaries as polynomial interpolants of the same polynomial order as the solution. A linear interpolation X is created between two opposing curves. Doing this for Γ_2 and Γ_4:

$$X_{42}(\tilde{\xi}, \tilde{\eta}) = (1 - \tilde{\xi})\Gamma_4(\tilde{\eta}) + \tilde{\xi}\Gamma_2(\tilde{\eta}), \tag{2.18}$$

and for Γ_1 and Γ_3:

$$X_{13}(\tilde{\xi}, \tilde{\eta}) = (1 - \tilde{\eta})\Gamma_1(\tilde{\xi}) + \tilde{\eta}\Gamma_3(\tilde{\xi}). \tag{2.19}$$

Summing the interpolations produces

$$\sum(\tilde{\xi}, \tilde{\eta}) = (1 - \tilde{\xi})\Gamma_4(\tilde{\eta}) + \tilde{\xi}\Gamma_2(\tilde{\eta}) + (1 - \tilde{\eta})\Gamma_1(\tilde{\xi}) + \tilde{\eta}\Gamma_3(\tilde{\xi}), \tag{2.20}$$

however, the sum is no longer consistent with the boundaries.

$$\begin{aligned} \sum(0, \tilde{\eta}) &= \Gamma_4(\tilde{\eta}) + \{(1 - \tilde{\eta})\Gamma_1(0) + \tilde{\eta}\Gamma_3(0)\} \\ \sum(1, \tilde{\eta}) &= \Gamma_2(\tilde{\eta}) + \{(1 - \tilde{\eta})\Gamma_1(1) + \tilde{\eta}\Gamma_3(1)\} \end{aligned} \tag{2.21}$$

In order to match the boundaries, the linear interpolant of the terms in the braces is

subtracted.

$$X(\tilde{\xi},\tilde{\eta}) = (1-\tilde{\xi})\Gamma_4(\tilde{\eta}) + \tilde{\xi}\Gamma_2(\tilde{\eta}) + (1-\tilde{\eta})\Gamma_1(\tilde{\xi}) + \tilde{\eta}\Gamma_3(\tilde{\xi}) \\ - (1-\tilde{\xi})\{(1-\tilde{\eta})\Gamma_1(0) + \tilde{\eta}\Gamma_3(0)\} - \tilde{\xi}\{(1-\tilde{\eta})\Gamma_1(1) + \tilde{\eta}\Gamma_3(1)\} \quad (2.22)$$

We then apply the following affine map to the above expression.

$$\tilde{\xi} = \frac{\xi+1}{2}, \tilde{\eta} = \frac{\eta+1}{2} \quad (2.23)$$

Finally, we obtain the mapping function to the reference square,

$$X(\xi,\eta) = \frac{1}{2}[(1-\xi)\Gamma_4(\eta) + (1+\xi)\Gamma_2(\eta) + (1-\eta)\Gamma_1(\xi) + (1+\eta)\Gamma_3(\xi)] \\ - \frac{1}{4}[(1-\xi)\{(1-\eta)\Gamma_1(-1) + (1+\eta)\Gamma_3(-1)\} \\ + (1+\xi)\{(1-\eta)\Gamma_1(1) + (1+\eta)\Gamma_3(1)\}]. \quad (2.24)$$

Equation (2.24) is used to develop the TransfiniteQuadMap algorithm presented in Algorithm 8. Figure 2.4 illustrates the mapping from the reference square to two different sample geometries.

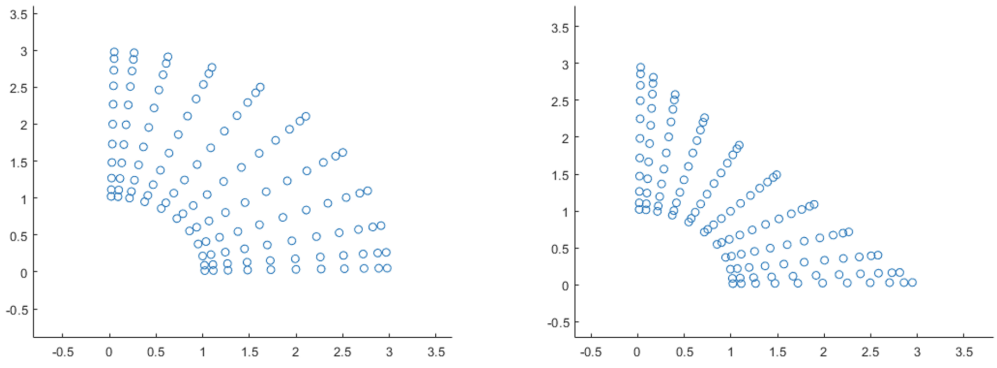


Figure 2.4: Example mappings from the reference square to curvilinear geometries for a single element with polynomial order $p = 10$. Blue circles are the element nodes. Left: quarter annulus, right: quarter annulus with a straight edge.

The derivatives of the mapping, also known as the metric terms, are required to perform the change of variables. The metric terms are as follows:

$$\begin{aligned}\frac{\partial X}{\partial \xi} &= \frac{1}{2}[\Gamma_2(\eta) - \Gamma_4(\eta) + (1-\eta)\Gamma_1\prime(\xi) + (1+\eta)\Gamma_3\prime(\xi)] \\ &\quad - \frac{1}{4}\{(1-\eta)[\Gamma_1(1) - \Gamma_1(-1)] + (1+\eta)[\Gamma_3(1) - \Gamma_3(-1)]\},\end{aligned} \tag{2.25}$$

$$\begin{aligned}\frac{\partial X}{\partial \eta} &= \frac{1}{2}[(1-\xi)\Gamma_4\prime(\eta) + (1+\xi)\Gamma_2\prime(\eta) + \Gamma_3(\xi) - \Gamma_1(\xi)] \\ &\quad - \frac{1}{4}\{(1-\xi)[\Gamma_3(-1) - \Gamma_1(-1)] + (1+\xi)[\Gamma_3(1) - \Gamma_1(-1)]\}.\end{aligned} \tag{2.26}$$

Equations (2.25) and (2.26) are used to develop the TransfiniteQuadMetrics algorithm in Algorithm 9. To finish the transformation from square to non-square geometries, we require the Jacobian, boundary normal vectors and scaling factors.

First, we introduce two types of basis vectors. The covariant basis vector varies along a coordinate line. The contravariant basis vector points normal to a coordinate line. Figure 2.5 shows a visual representation of these vectors.

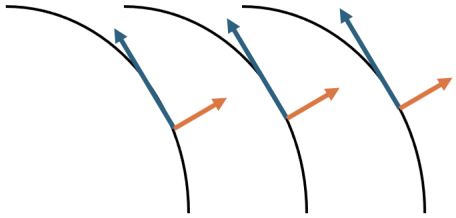


Figure 2.5: Covariant basis vectors (blue) vary along coordinate lines (black) and contravariant basis vectors (orange) point normal to coordinate lines.

The covariant basis vectors for the mapping $x = X(\xi, \eta)$ are as follows:

$$a_1 = \frac{\partial X}{\partial \xi} = X_\xi \hat{x} + Y_\xi \hat{y}, \quad a_2 = \frac{\partial X}{\partial \eta} = X_\eta \hat{x} + Y_\eta \hat{y}, \quad a_3 = \hat{z}. \tag{2.27}$$

From Equation (2.27), the Jacobian can be produced:

$$J = a_1 \cdot (a_2 \times a_3) = X_\xi Y_\eta - Y_\xi X_\eta. \tag{2.28}$$

We obtain the contravariant basis vectors by computing the cross product of the covariant basis vectors

$$Ja^1 = a_2 \times a_3 = Y_\eta \hat{x} - X_\eta \hat{y}, \tag{2.29}$$

$$Ja^2 = a_3 \times a_1 = -Y_\xi \hat{x} + X_\xi \hat{y}. \tag{2.30}$$

The scaling factors are calculated using the following equations:

$$\text{scal}_1 = \sqrt{Y_\eta^2 + X_\eta^2}, \tag{2.31}$$

$$\text{scal}_2 = \sqrt{Y_\xi^2 + X_\xi^2}. \tag{2.32}$$

Using the contravariant basis vectors, the scaling factors and the sign of the Jacobian, we can calculate the boundary normal vectors:

$$\hat{n}^1 = \frac{|J|}{J} \frac{Y_\eta \hat{x} - X_\eta \hat{y}}{\sqrt{Y_\eta^2 + X_\eta^2}}, \tag{2.33}$$

$$\hat{n}^2 = \frac{|J|}{J} \frac{-Y_\xi \hat{x} + X_\xi \hat{y}}{\sqrt{Y_\xi^2 + X_\xi^2}}. \tag{2.34}$$

Now, all of the components for performing the transfinite mapping have been presented. The following section presents the algorithmic implementation of the mapping procedure.

2.5 Algorithm Implementation

This section presents the algorithms in pseudocode format used to implement the transfinite mapping. The pseudocodes used in the implementation were developed by Kopriva [23]. To begin, a class is used to represent the curved boundaries. The following pseudocodes present the CurveInterpolant class.

Algorithm 1 CurveInterpolant Class

Attributes: $N, \text{nodes}_j, x_j, y_j, w_j$
Methods: Constructor, EvaluatePoint, DerivativePoint

Algorithm 2 CurveInterpolant Constructor

Input: $N, \text{nodes}_j, x_j, y_j$
$N \leftarrow N$
for `j = 0 to N` **do**
$\text{nodes}_j \leftarrow \text{nodes}_j$
$x_j \leftarrow x_j$
$y_j \leftarrow y_j$
end for
$w_j \leftarrow \text{BarycentricWeights}(\text{nodes}_j)$

N denotes the polynomial order, nodes_j are the Gauss-Legendre nodes in the interval [-1,1] and (x_j, y_j) are the node locations. The barycentric weights used for Lagrange interpolation are denoted by w_j. The pseudocodes for computing the barycentric weights, evaluating points and the derivatives at the points are given in Algorithms 3, 4 and 5 respectively. The pseudocodes for Lagrange interpolation and evaluating the Lagrange interpolant derivative are presented in Algorithms 6 and 7.

Algorithm 3 BarycentricWeights

Input: x_j
Output: w_j
for `j = 0 to N` **do**
$w_j \leftarrow 1$
end for
for `j = 1 to N` **do**
for `k = 0 to j - 1` **do**
$w_k \leftarrow w_k(x_k - x_j)$
$w_j \leftarrow w_j(x_j - x_k)$
end for
end for
for `j = 0 to N` **do**
$w_j \leftarrow 1/w_j$
end for

Algorithm 4 EvaluatePoint

Input: s

Output: x, y

$x \leftarrow \text{LagrangeInterpolation}(s, \text{nodes}, x_j, w_j)$

$y \leftarrow \text{LagrangeInterpolation}(s, \text{nodes}, y_j, w_j)$

Algorithm 5 DerivativePoint

Input: s

Output: x', y'

$x' \leftarrow \text{LagrangeInterpolantDerivative}(s, \text{nodes}, x_j, w_j)$

$y' \leftarrow \text{LagrangeInterpolantDerivative}(s, \text{nodes}, y_j, w_j)$

Algorithm 6 LagrangeInterpolation

Input: x, x_j, f_j, w_j

Output: result

numerator $\leftarrow 0$

denominator $\leftarrow 0$

for `j = 0 to N` **do**

if $x \approx x_j$ **then** result $\leftarrow f_j$

end if

$t \leftarrow w_j(x_j - x_k)$

numerator $\leftarrow$ numerator $+ t f_j$

denominator $\leftarrow$ denominator $+ t$

end for

result $\leftarrow$ numerator/denominator

Algorithm 7 LagrangeInterpolantDerivative

Input: x, x_j, f_j, w_j

Output: result

numerator $\leftarrow 0$

flag $\leftarrow$ false

for `j = 0 to N` **do**

if $x \approx x_j$ **then**

flag $\leftarrow$ true

$p \leftarrow f_j$

denominator $\leftarrow -w_j$

$i \leftarrow j$

end if

end for

if flag is true **then**

for `j = 0 to N` **do**

if $j \neq i$ **then**

numerator $\leftarrow$ numerator $+ w_j * \frac{p - f_j}{x - x_j}$

end if

end for

else if flag is false **then**

denominator $\leftarrow 0$

$p \leftarrow$ LagrangeInterpolation

for `j = 0 to N` **do**

$t \leftarrow w_j/(x - x_j)$

numerator $\leftarrow$ numerator $+ t * \frac{p - f_j}{x - x_j}$

denominator $\leftarrow$ denominator $+ t$

end for

end if

result $\leftarrow$ numerator/denominator

Next, the pseudocodes that implement the transfinite mapping itself and the calculation of the metric terms are presented in Algorithms 8 and 9 respectively.

Algorithm 8 TransfiniteQuadMap

Input: ξ, η, Γ_j ▷ Γ is a CurveInterpolant object

Output: x, y

$(x_1, y_1) \leftarrow \Gamma_1$.EvaluatePoint$(-1)$

$(x_2, y_2) \leftarrow \Gamma_1$.EvaluatePoint$(1)$

$(x_3, y_3) \leftarrow \Gamma_3$.EvaluatePoint$(1)$

$(x_4, y_4) \leftarrow \Gamma_3$.EvaluatePoint$(-1)$

$(X_1, Y_1) \leftarrow \Gamma_1$.EvaluatePoint$(\xi)$

$(X_2, Y_2) \leftarrow \Gamma_2$.EvaluatePoint$(\eta)$

$(X_3, Y_3) \leftarrow \Gamma_3$.EvaluatePoint$(\xi)$

$(X_4, Y_4) \leftarrow \Gamma_4$.EvaluatePoint$(\eta)$

$$x \leftarrow \tfrac{1}{2}[(1-\xi)X_4 + (1+\xi)X_2 + (1-\eta)X_1 + (1+\eta)X_3]$$

$$x \leftarrow x - \tfrac{1}{4}[(1-\xi)\{(1-\eta)x_1 + (1+\eta)x_4\} + (1+\xi)\{(1-\eta)x_2 + (1+\eta)x_3\}]$$

$$y \leftarrow \tfrac{1}{2}[(1-\xi)Y_4 + (1+\xi)Y_2 + (1-\eta)Y_1 + (1+\eta)Y_3]$$

$$y \leftarrow y - \tfrac{1}{4}[(1-\xi)\{(1-\eta)y_1 + (1+\eta)y_4\} + (1+\xi)\{(1-\eta)y_2 + (1+\eta)y_3\}]$$

The points in the reference square are ξ and η, the four curved boundaries are denoted by Γ_j, x and y are the mapped points (in the solution domain). The metric terms are calculated in Algorithm 9 and correspond to Equations (2.25) and (2.26).

Algorithm 9 TransfiniteQuadMetrics

Input: ξ, η, Γ_j ▷ Γ is a CurveInterpolant object

Output: $X_\xi, X_\eta, Y_\xi, Y_\eta$

$(x_1, y_1) \leftarrow \Gamma_1$.EvaluatePoint$(-1)$

$(x_2, y_2) \leftarrow \Gamma_1$.EvaluatePoint$(1)$

$(x_3, y_3) \leftarrow \Gamma_3$.EvaluatePoint$(1)$

$(x_4, y_4) \leftarrow \Gamma_3$.EvaluatePoint$(-1)$

$(X_1, Y_1) \leftarrow \Gamma_1$.EvaluatePoint$(\xi)$

$(X_2, Y_2) \leftarrow \Gamma_2$.EvaluatePoint$(\eta)$

$(X_3, Y_3) \leftarrow \Gamma_3$.EvaluatePoint$(\xi)$

$(X_4, Y_4) \leftarrow \Gamma_4$.EvaluatePoint$(\eta)$

$(X'_1, Y'_1) \leftarrow \Gamma_1$.DerivativePoint$(\xi)$

$(X'_2, Y'_2) \leftarrow \Gamma_2$.DerivativePoint$(\eta)$

$(X'_3, Y'_3) \leftarrow \Gamma_3$.DerivativePoint$(\xi)$

$(X'_4, Y'_4) \leftarrow \Gamma_4$.DerivativePoint$(\eta)$

$$X_\xi \leftarrow \tfrac{1}{2}\{X_2 - X_4 + (1-\eta)X'_1 + (1+\eta)X'_3\} - \tfrac{1}{4}\{(1-\eta)(x_2 - x_1) + (1+\eta)(x_3 - x_4)\}$$

$$Y_\xi \leftarrow \tfrac{1}{2}\{Y_2 - Y_4 + (1-\eta)Y'_1 + (1+\eta)Y'_3\} - \tfrac{1}{4}\{(1-\eta)(y_2 - y_1) + (1+\eta)(y_3 - y_4)\}$$

$$X_\eta \leftarrow \tfrac{1}{2}\{(1-\xi)X'_4 + (1+\xi)X'_2 + X_3 - X_1\} - \tfrac{1}{4}\{(1-\xi)(x_4 - x_1) + (1+\xi)(x_3 - x_2)\}$$

$$Y_\eta \leftarrow \tfrac{1}{2}\{(1-\xi)Y'_4 + (1+\xi)Y'_2 + Y_3 - Y_1\} - \tfrac{1}{4}\{(1-\xi)(y_4 - y_1) + (1+\xi)(y_3 - y_2)\}$$

Algorithms 8 and 9 provide the necessary prerequisites for implementing the transfinite mapping. Algorithm 10 encapsulates all of the important data necessary for implementing the mapping in a so called "MappedGeometry class". The pseudocode for the MappedGeometry class constructor is presented in Algorithm 11.

Algorithm 10 MappedGeometry Class

Attributes:

N	▷ Polynomial order
$x_{i,j}, y_{i,j}$	▷ These are the internal node locations
$x\text{boundary}_i, y\text{boundary}_i$	▷ These are the boundary node locations
$X_{\xi i,j}, X_{\eta i,j}$	▷ These are the metric terms in x
$Y_{\xi i,j}, Y_{\eta i,j}$	▷ These are the metric terms in y
$J_{i,j}$	▷ This is the Jacobian
$\hat{n}_i$	▷ This is the boundary normal
scal_i	▷ This is the scaling factor

Methods:

Constructor

Algorithm 11 MappedGeometry Constructor

for `j = 0 to N` **do**

for `i = 0 to N` **do**

$(x_{i,j}, y_{i,j}) \leftarrow \text{TransfiniteQuadMap}(\Gamma_j, \text{GLnodes}_i, \text{GLnodes}_j)$

$(X_\xi, X_\eta, Y_\xi, Y_\eta) \leftarrow \text{TransfiniteQuadMetrics}(\Gamma_j, \text{GLnodes}_i, \text{GLnodes}_j)$

$J_{i,j} \leftarrow (X_\xi * Y_\eta) - (X_\eta * Y_\xi)$

end for

end for

for `j = 0 to N` **do**

$(x\text{boundary}_{j,2}, y\text{boundary}_{j,2}) \leftarrow \text{TransfiniteQuadMap}(\Gamma_j, 1, \text{GLnodes}_j)$

$(X_\xi, X_\eta, Y_\xi, Y_\eta) \leftarrow \text{TransfiniteQuadMetrics}(\Gamma_j, 1, \text{GLnodes}_j)$

$J \leftarrow (X_\xi * Y_\eta) - (X_\eta * Y_\xi)$

$\text{scal}_{j,2} \leftarrow \sqrt{Y_\eta^2 + X_\eta^2}$

$\hat{n}_{j,2} \leftarrow \text{Sign}(J) * (Y_\eta - X_\eta)/\text{scal}_{j,2}$

$(x\text{boundary}_{j,4}, y\text{boundary}_{j,4}) \leftarrow \text{TransfiniteQuadMap}(\Gamma_j, -1, \text{GLnodes}_j)$

$(X_\xi, X_\eta, Y_\xi, Y_\eta) \leftarrow \text{TransfiniteQuadMetrics}(\Gamma_j, -1, \text{GLnodes}_j)$

$J \leftarrow (X_\xi * Y_\eta) - (X_\eta * Y_\xi)$

$\text{scal}_{j,4} \leftarrow \sqrt{Y_\eta^2 + X_\eta^2}$

$\hat{n}_{j,4} \leftarrow -\text{Sign(J)} * (Y_\eta - X_\eta)/\text{scal}_{j,4}$

end for

for `i = 0 to N` **do**

$(x\text{boundary}_{i,1}, y\text{boundary}_{i,1}) \leftarrow \text{TransfiniteQuadMap}(\Gamma_j, \text{GLnodes}_i, -1)$

$(X_\xi, X_\eta, Y_\xi, Y_\eta) \leftarrow \text{TransfiniteQuadMetrics}(\Gamma_j, \text{GLnodes}_i, -1)$

$J \leftarrow (X_\xi * Y_\eta) - (X_\eta * Y_\xi)$

$scal_{i,1} \leftarrow \sqrt{Y_\xi^2 + X_\xi^2}$

$\hat{n}_{i,1} \leftarrow -\text{Sign}(J) * (-Y_\xi + X_\xi)/\text{scal}_{i,1}$

$(x\text{boundary}_{i,3}, y\text{boundary}_{i,3}) \leftarrow \text{TransfiniteQuadMap}(\Gamma_j, \text{GLnodes}_i, 1)$

$(X_\xi, X_\eta, Y_\xi, Y_\eta) \leftarrow \text{TransfiniteQuadMetrics}(\Gamma_j, \text{GLnodes}_i, 1)$

$J \leftarrow (X_\xi * Y_\eta) - (X_\eta * Y_\xi)$

$\text{scal}_{i,3} \leftarrow \sqrt{Y_\xi^2 + X_\xi^2}$

$\hat{n}_{i,3} \leftarrow \text{Sign}(J) * (-Y_\xi + X_\xi)/\text{scal}_{i,3}$

end for

2.6 Implementation of the Transfinite Mapping into the Code

This section describes how the algorithms presented in Section 2.5 are integrated into the existing solver to implement the transfinite mapping. Each element possesses an object of the MappedGeometry class. Next, when the solution is initialized, four CurveInterpolant objects are constructed to set the desired mapped geometry. After the four CurveInterpolant objects are constructed, the constructor of the MappedGeometry class is called.

The main modification to the code is made in the procedure for obtaining the DG time derivative. The first step of interpolating the solutions to the boundaries remains the same. The interpolation to the boundaries is required since we are using Gauss-Legendre nodes (no nodes are present on element boundaries). However, we multiply the output of the Riemann solver by the scaling factor to obtain the appropriate numerical fluxes. When computing the spatial derivative of the fluxes, we multiply the internal fluxes with the appropriate metric terms. Finally, when calculating the final portion of the time derivative, we divide by the Jacobian. The block diagram below shows the modifications that were made to the time derivative routine.

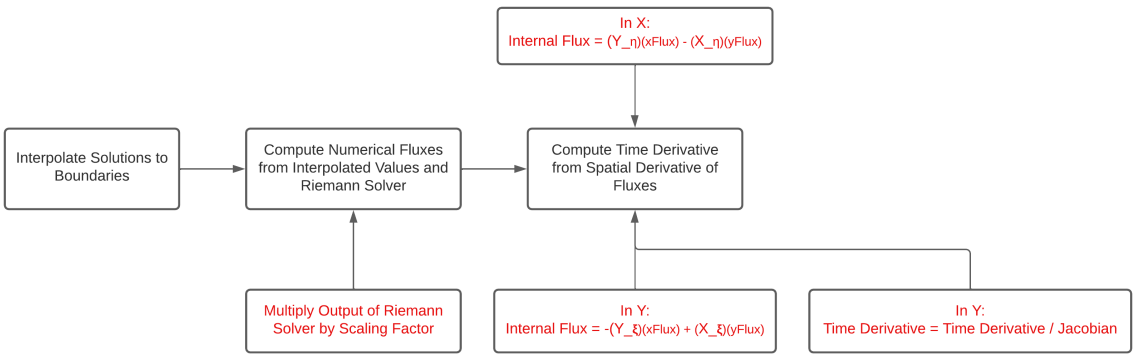


Figure 2.6: Modifications made (in red) to the DG time derivative routine to implement the transfinite mapping.

2.7 Simulations on Curvilinear Geometries

2.7.1 Benchmark Case: Gaussian Plane Wave Propagation in a Half-Cylindrical Domain

In this section, we demonstrate the transfinite mapping used to model curvilinear geometries on a test case which consists of a Gaussian plane wave propagation in a half-cylindrical domain. The geometry is defined by the following expressions:

$$\begin{aligned}
\Gamma_1(\xi) &= 0.5 + 4.5(\xi+1)/2\hat{x} + 0\hat{y}, \\
\Gamma_2(\eta) &= 5\cos(\pi(\eta+1)/2)\hat{x} + 5\sin(\pi(\eta+1)/2)\hat{y}, \\
\Gamma_3(\xi) &= -0.5 - 4.5(\xi+1)/2\hat{x} + 0\hat{y}, \\
\Gamma_4(\eta) &= 0.5\cos(\pi(\eta+1)/2)\hat{x} + 0.5\sin(\pi(\eta+1)/2)\hat{y},
\end{aligned} \tag{2.35}$$

and is illustrated in Figure 2.7. The initial and boundary conditions are given by the exact solution to a Gaussian plane acoustic wave propagating through the domain:

$$\begin{bmatrix} P \\ u \\ v \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{k_x}{c} \\ \frac{k_y}{c} \end{bmatrix} e^{-\frac{\left[k_x(x-x_0)+k_y(y-y_0)-ct\right]^2}{d^2}}, \tag{2.36}$$

where k_x and k_y are the components of the wave vector and are set to $k_x = k_y = \frac{\sqrt{2}}{2}$ such that the wave propagates towards the top right. We set the parameter $d = \frac{0.2}{2\sqrt{log(2)}}$ and the wavespeed $c = 1$. We show the initial condition in the domain in Figure 2.7.

We perform a polynomial convergence test of the calculation of the wave propagating in the domain. The grid is discretized with 32 elements in both the radial and azimuthal directions and a fixed small timestep $\Delta t = 5 \times 10^{-5}$ is used in accordance with the stability limits of the CFL condition. We calculate the L_2 error with respect to the exact solution in Equation (2.36), where the L_2 error is defined as $\sqrt{\int_\Omega (u - u_{\text{exact}})^2 d\Omega}$. The results of the polynomial convergence test are shown in Figure 2.8. We observe

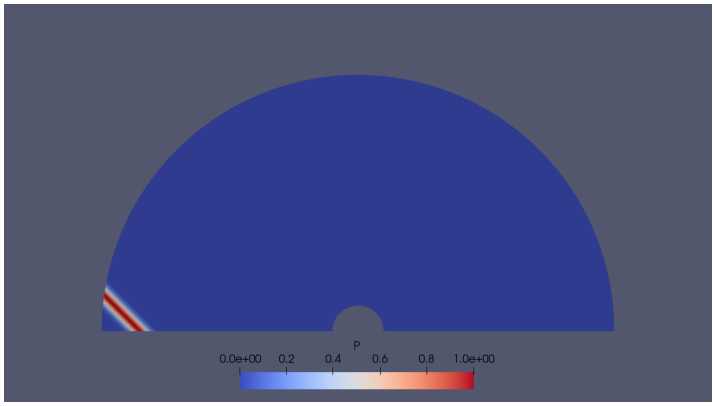


Figure 2.7: Initial condition for the Gaussian plane wave propagating in the half circular cylinder domain.

exponential convergence is still obtained even on a very deformed geometry.

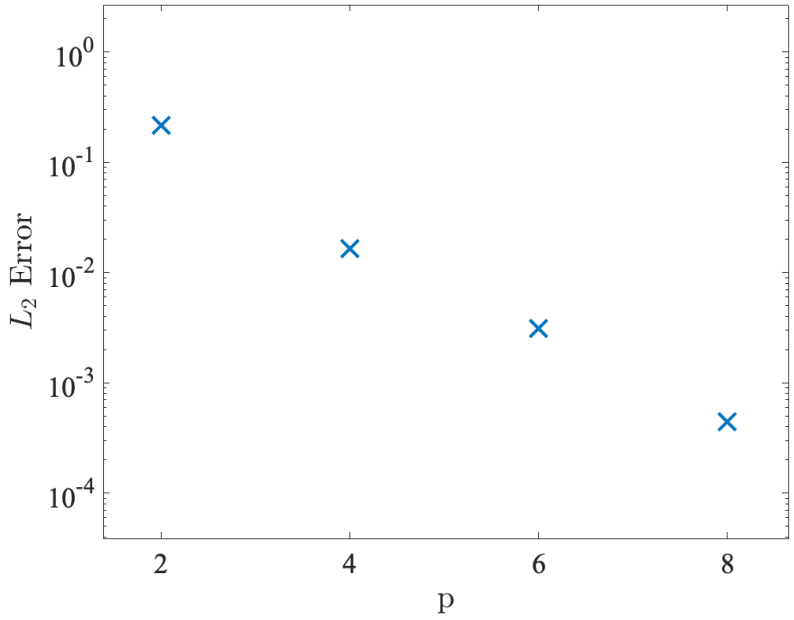


Figure 2.8: Polynomial convergence test of a Gaussian plane wave propagating in the half circular cylinder domain. Exact solution given in Equation (2.36). Exponential convergence is obtained as the polynomial order p is increased.

2.7.2 Acoustic Scattering off a Half Circular Cylinder

The next simulation is of an acoustic scattering off a half circular cylinder, which can act as a simplified model of the acoustic loading on the surface of an aircraft (half circular cylinder) from an engine noise source (initial condition) [23]. This case is used

for the strong scaling tests. The geometry is the same as the previous test case and is defined in Equation (2.35). We use the following initial conditions for a pressure perturbation:

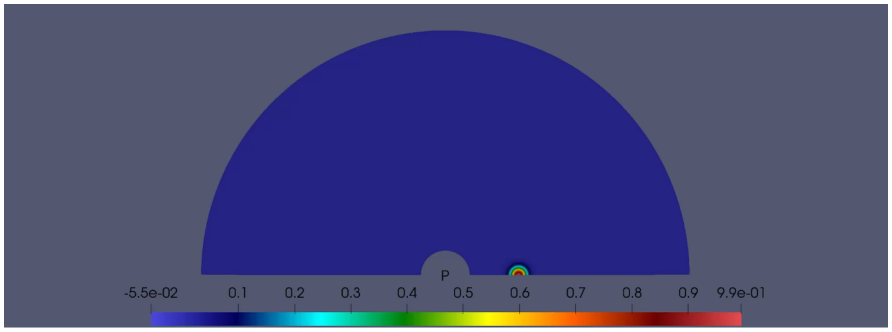
$$\begin{aligned} P(x,y,0) &= e^{-ln(2)(\frac{(x-1.5)^2+y^2}{0.125^2})} \\ u(x,y,0) &= 0 \\ v(x,y,0) &= 0. \end{aligned} \tag{2.37}$$

We impose wall boundary conditions on the bottom walls and the inner circular boundary. Radiation boundary conditions are imposed on the outer circular boundary. We show a sample simulation in Figures 2.9 and 2.10. The domain starts with 16 elements in both the radial and azimuthal directions with a polynomial order of $p = 5$ to sufficiently resolve the initial condition. We allow for two levels of h-refinement and p-refinement up to $p = 11$. We observe the hp-adaptation following the profile of the wave very well. There is mostly h-refinement accompanied with one level of p-refinement. Furthermore, we observe additional p-refinement near the location of the initial pressure perturbation.

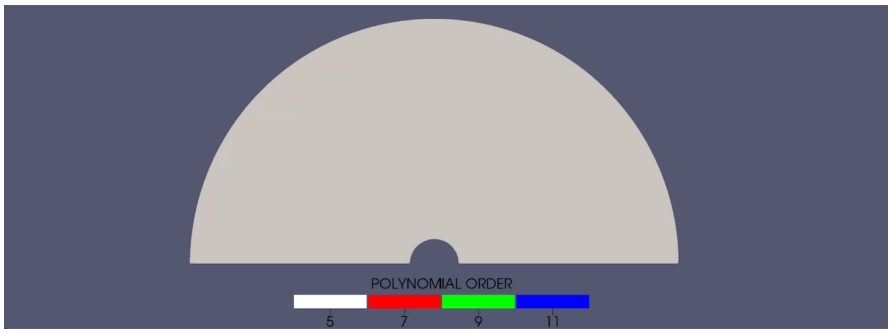
2.7.3 Gaussian Wave Propagation in Curved Channel

Next, we simulate a Gaussian plane wave propagating in a curved channel. The geometry is defined by the following expressions:

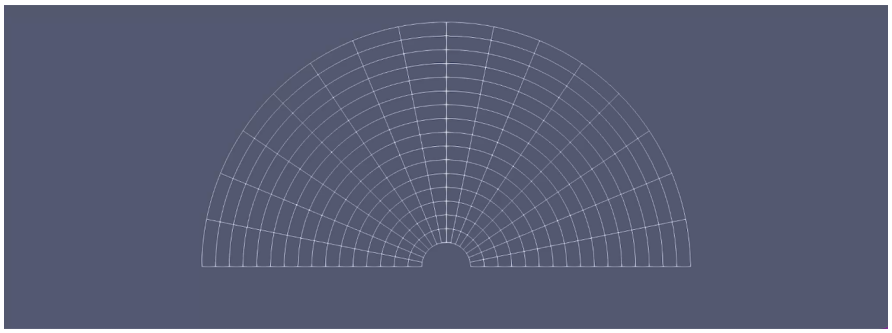
$$\begin{aligned} \Gamma_1(\xi) &= 3\xi/2\hat{x} + -(0.3 + 0.35(\tanh(2\xi) + 1)\hat{y} \\ \Gamma_2(\eta) &= 3/2\hat{x} + (0.289 + 0.35(\tanh(3) + 1))\eta\hat{y} \\ \Gamma_3(\xi) &= 3\xi/2\hat{x} + (0.3 + 0.35(\tanh(2\xi) + 1)\hat{y} \\ \Gamma_4(\eta) &= -3/2\hat{x} + (0.155(\tanh(3) + 1))\eta\hat{y}. \end{aligned} \tag{2.38}$$



(a)



(b)



(c)

Figure 2.9: Half circular cylinder acoustic scattering: initial pressure perturbation. (a): pressure, (b): polynomial order, (c): grid.

The initial conditions used are

$$\begin{aligned} P(x,y,0) &= e^{-(\frac{(\frac{\sqrt{2}}{2}(x+1)}{(\frac{0.2}{2\sqrt{log(2)}})^2})} \\ u(x,y,0) &= e^{-(\frac{(\frac{\sqrt{2}}{2}(x+1)}{(\frac{0.2}{2\sqrt{log(2)}})^2})} \\ v(x,y,0) &= 0. \end{aligned} \tag{2.39}$$

The boundary conditions are given by the exact solution of the Gaussian plane wave propagating undisturbed through the domain. We show a sample hp-adaptive simula-

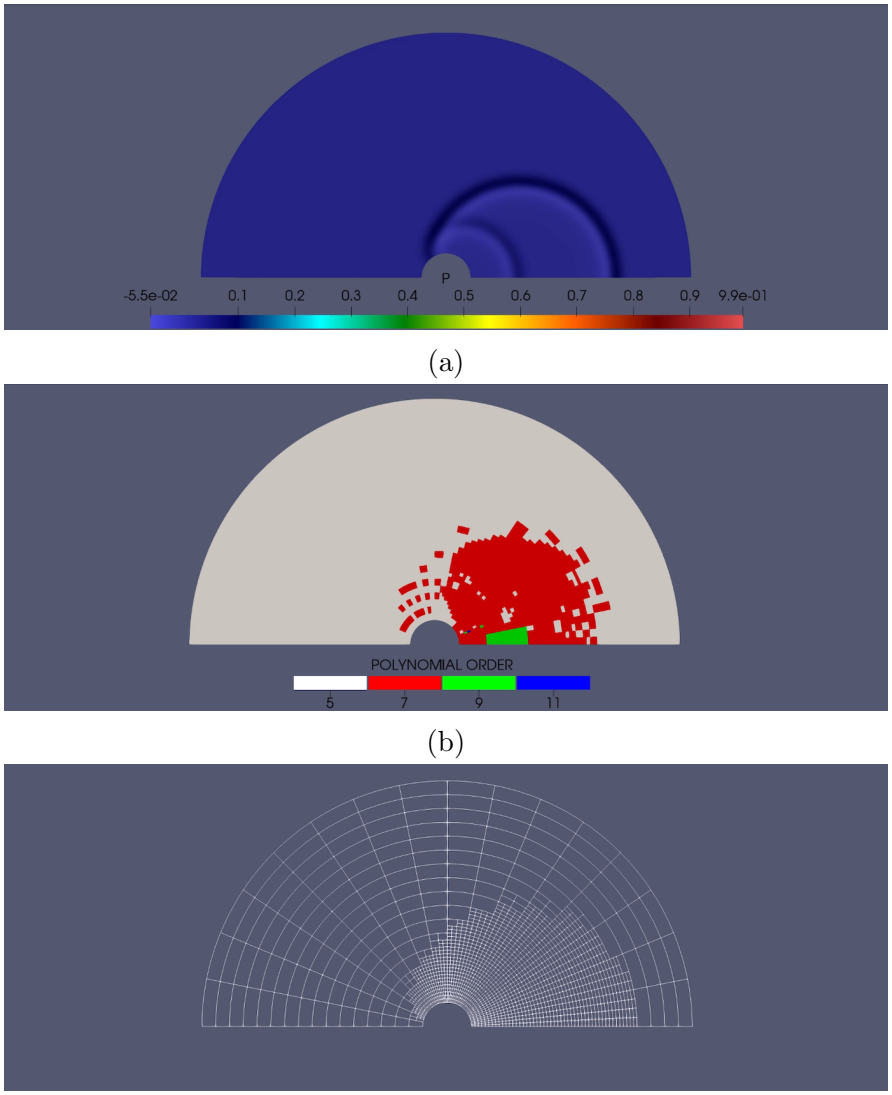


Figure 2.10: Half circular cylinder acoustic scattering: propagation of initial perturbation and wave reflection off half circular cylinder using hp-adaptation. Grid starts with 16×16 elements with polynomial order $p_{initial} = 5$. Two levels of h-refinement used and p-refinement up to $p = 11$. (a): pressure, (b): polynomial order, (c): grid.

tion using this geometry in Figures 2.11 and 2.12. The domain starts with 8 elements in each direction with a polynomial order of $p = 5$ to sufficiently resolve the initial wave. We allow for 2 levels of h-refinement and p-refinement up to $p = 11$. The hp-adaptation follows the profile of the propagating Gaussian wave well. There is mostly one level of p-refinement, with additional p-refinement occurring near the location of the initial wave position.

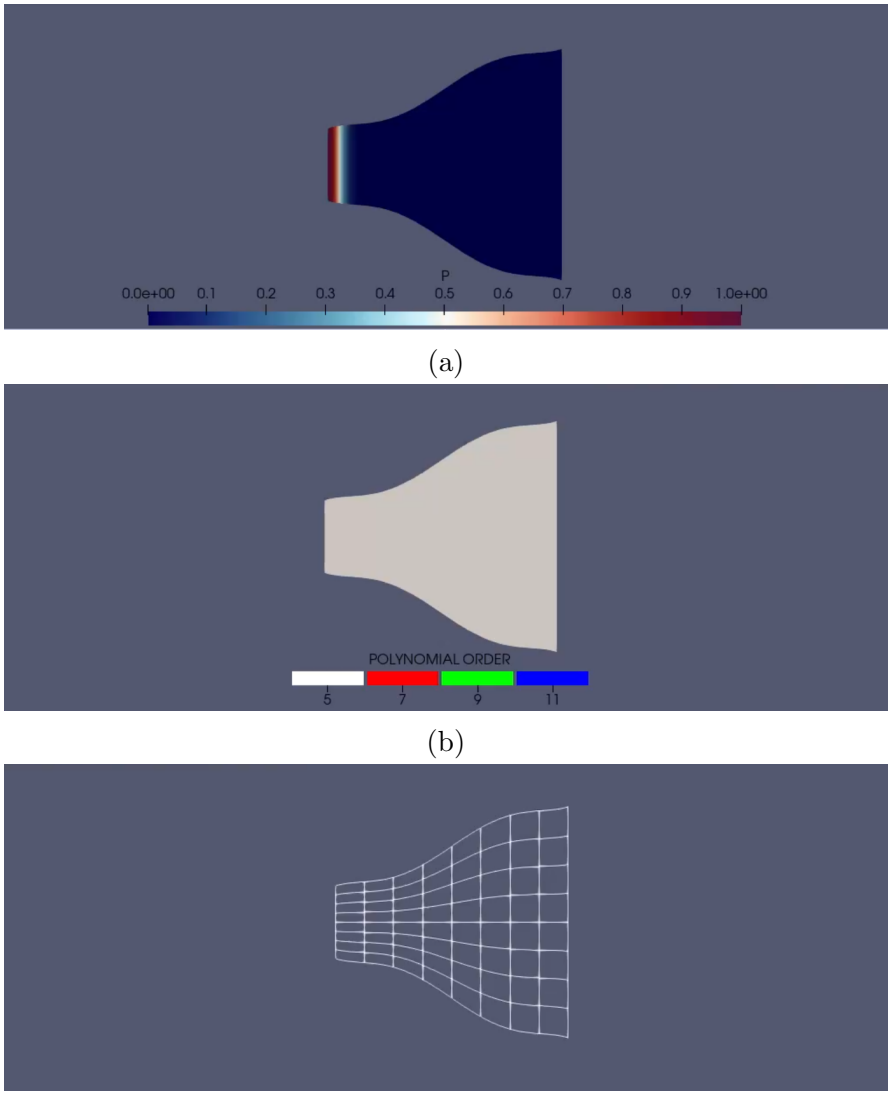


Figure 2.11: Initial condition for the curved channel case. (a): pressure, (b): polynomial order, (c): grid.

2.8 Scaling Tests

In this section, we test the parallel performance of the transfinite mapped code and compare it to that of the Cartesian code by He [20]. We perform two types of scaling tests, weak scaling and strong scaling, both using transfinite mapped grids. In the weak scaling tests, the problem size scales with the computational resources. Whereas in the strong scaling tests, the problem size stays constant and the computational resources used varied. We use the Tuning and Analysis Utilities (TAU) profiling library to measure the runtime and memory usage of the code during the scaling tests. TAU

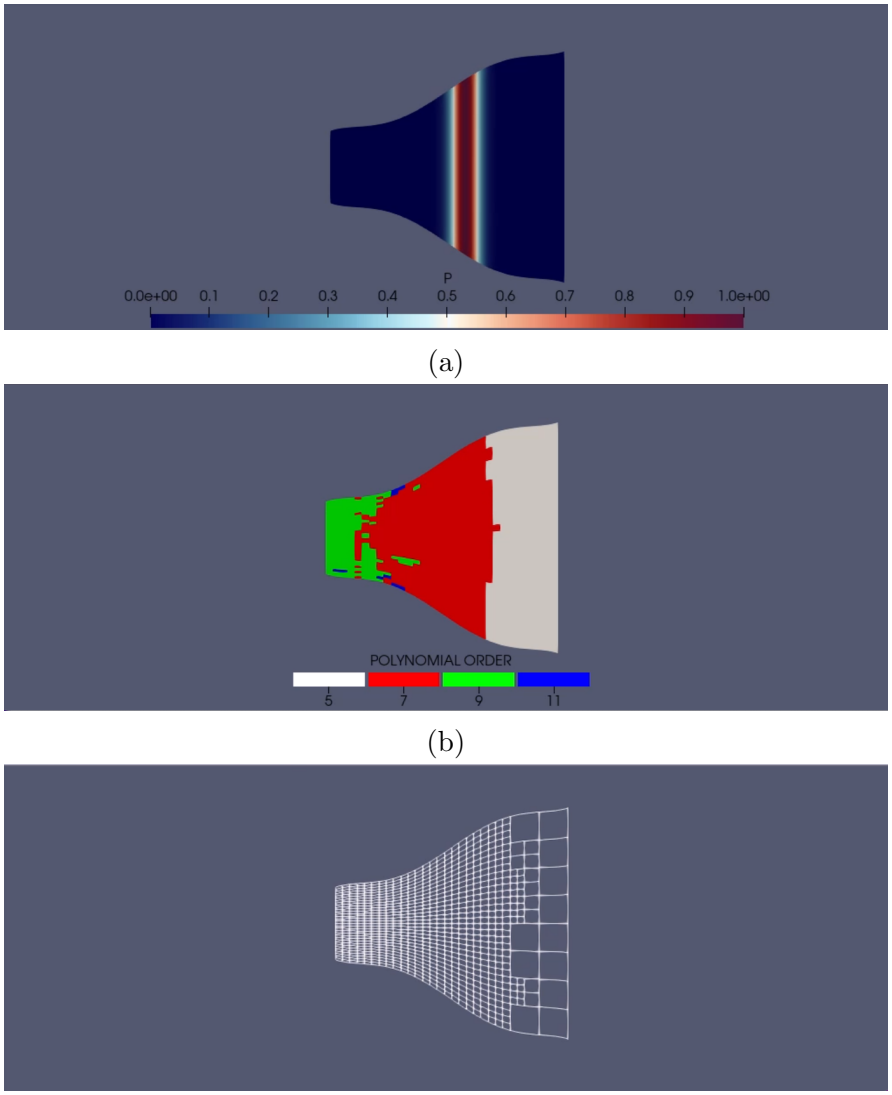


Figure 2.12: Gaussian plane wave propagation towards the right in curved channel. Grid starts with 8×8 elements with polynomial order $p_{initial} = 5$. Two levels of h-refinement used and p-refinement up to $p = 11$. (a): pressure, (b): polynomial order, (c): grid.

is employed since it was previously used by He to measure the performance of the Cartesian code [20].

2.8.1 High Performance Computing Platforms

The high performance computing platforms were provided by the Digital Research Alliance of Canada (formerly Compute Canada). We use the Niagara cluster. The specifications of which will now be presented.

Niagara

The Niagara cluster possesses 40 cores per node with 202 Gb of memory per node. It uses a Intel Skylake architecture, running at 2.4 GHz. Niagara uses an EDR Dragonfly+ network connection, with a bandwidth of 100 Gb/s. Niagara is located at the University of Toronto and operated by SciNet. Ideally, the Graham cluster could have been used for the scaling tests to match the platform used by He. However, due to technical issues with Graham this year, we were forced to seek alternatives and we used Niagara for the scaling tests.

2.8.2 Weak Scaling Test

For the weak scaling test, we use the case presented in Section 2.7.3 of the Gaussian plane wave propagating in the curved channel. We use a non-adaptive grid with a polynomial order of $p = 6$, however the resources used scale with the problem size. The parameters for the weak scaling test are presented in Table 2.1. We also simulated equivalent cases of a Gaussian plane wave propagating in a square domain using the Cartesian code of He. The results of the weak scaling test are presented in Figure 2.13 and compared to He's results.

Table 2.1: Weak scaling test parameters.

Elements in Each Direction	**Processors**
64×64	80
128×128	320
256×256	1280
512×512	5120
1024×1024	20480

The results show the transfinite mapped code scales similarly to the Cartesian code. However, as expected, there is a slight increase in runtime. For the last three data points, the transfinite mapped code has an approximately 24 % increase in runtime. The increase in runtime can be attributed to the extra overhead incurred by the transfinite mapping procedure. Overall, the transfinite mapped code shows a similar trend to the

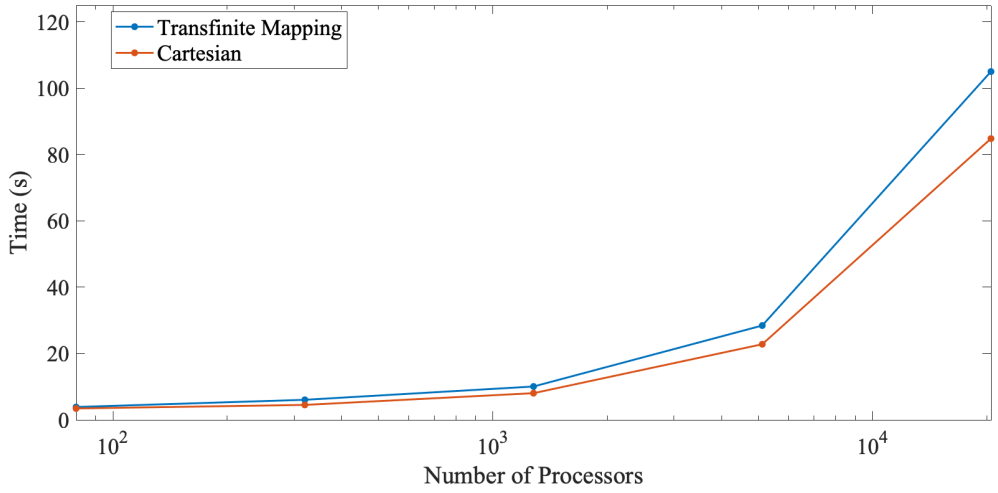


Figure 2.13: Weak scaling runtime results comparing transfinite mapped code (blue) with Cartesian code (orange).

Cartesian code.

2.8.3 Strong Scaling Test

For the strong scaling test, we use the case presented in Section 2.7.2 of the acoustic scattering of a half circular cylinder. The domain is discretized with 256 elements in both the radial and azimuthal directions. We start with a polynomial order of $p = 4$ and allow p-refinement up to $p = 8$. Furthermore, we allow for two levels of h-refinement. We run the simulation for 20 time steps, adapting every five time steps. The simulation ends with 69,115 elements. The runtime results and memory usage results are presented in Figures 2.14 and 2.16 respectively. The results are compared to a strong scaling test performed by He in Figures 2.15 and 2.17 respectively.

The runtime results in Figure 2.14 show an initial decrease as more computational resources are used, however the execution time does eventually increase slightly. Overall, the runtime remains relatively in the same order of magnitude. Comparing to He's results in Figure 2.15, the transfinite mapped code unfortunately does not scale as well. However, this is expected as the transfinite mapping procedure requires more computational work. Next we observe the memory usage results in Figure 2.16. The memory usage remains overall constant, with the memory plateauing after the second

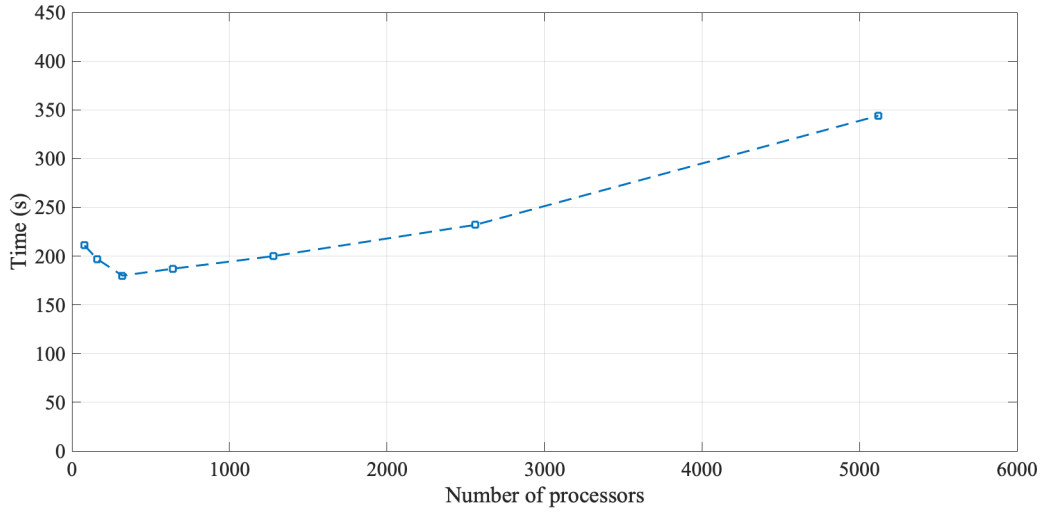


Figure 2.14: Strong scaling runtime results.

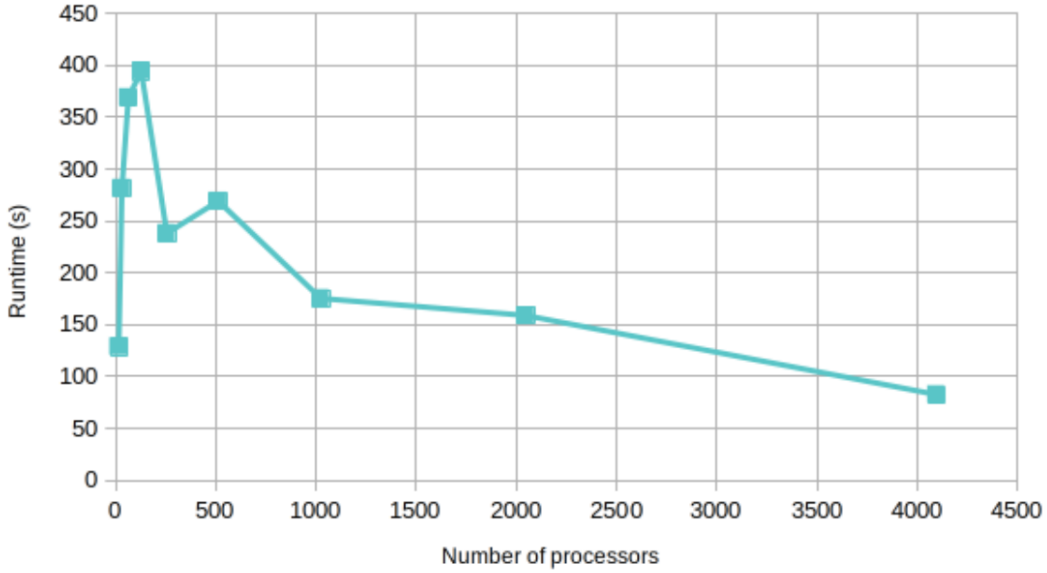


Figure 2.15: Cartesian code strong scaling runtime results, reprinted from [20].

case. Comparing to He's results in Figure 2.17 we observe a similar trend of the memory remaining overall constant, with a plateau after. However, we do note that the memory usage in the transfinite mapped case is significantly larger. This is because during the dynamic load balancing procedure, the mapped geometry class needs to also be sent in order to maintain the transfinite mapping.

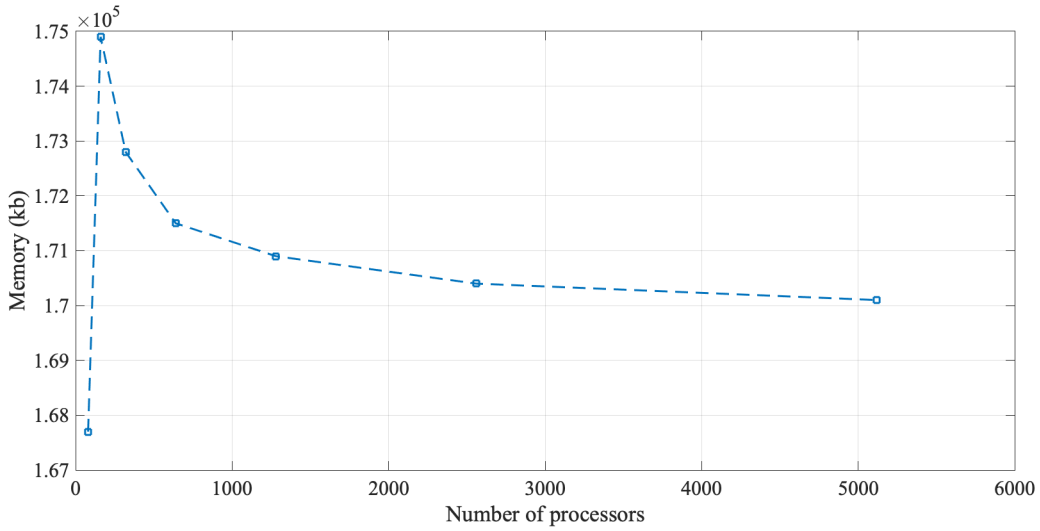


Figure 2.16: Strong scaling memory results.

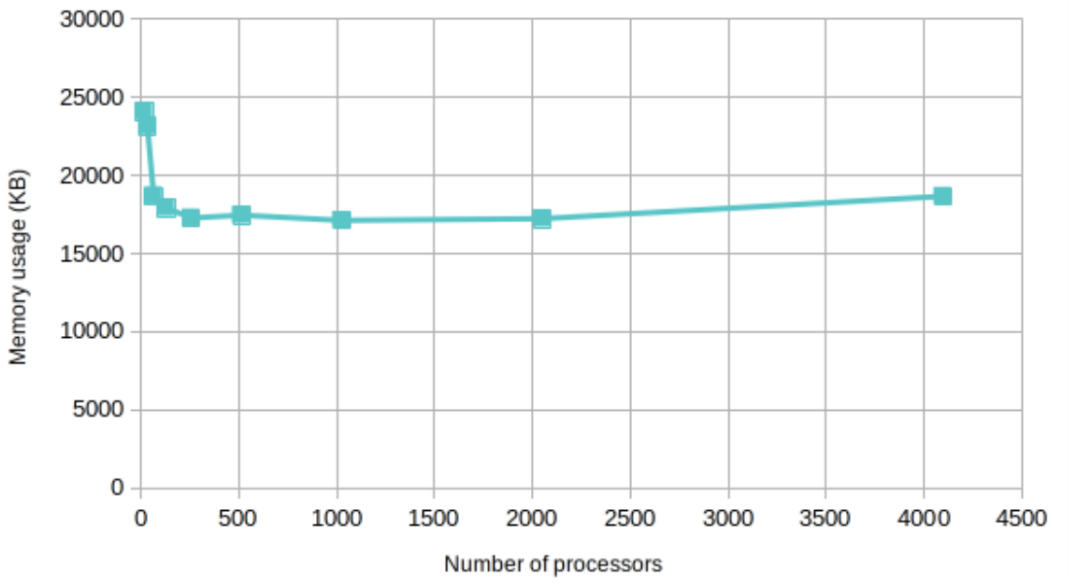


Figure 2.17: Cartesian code strong scaling memory results, reprinted from [20].

2.8.4 Discussion and Future Directions

The results of the scaling tests will now be discussed. For the weak scaling test, we observe a similar trend in runtime compared to the Cartesian code. However, as expected, there is more computational effort required for the transfinite mapping which translates to increased run times. Unfortunately for the strong scaling tests, we observe for the run times the transfinite mapping does not show the same scaling trend as the Cartesian code. The runtime slightly increases but overall remains relatively constant.

Furthermore we observe a significant increase in memory usage. This extra runtime and memory can be attributed to the use of the mapped geometry class, shown in Figure 2.18.

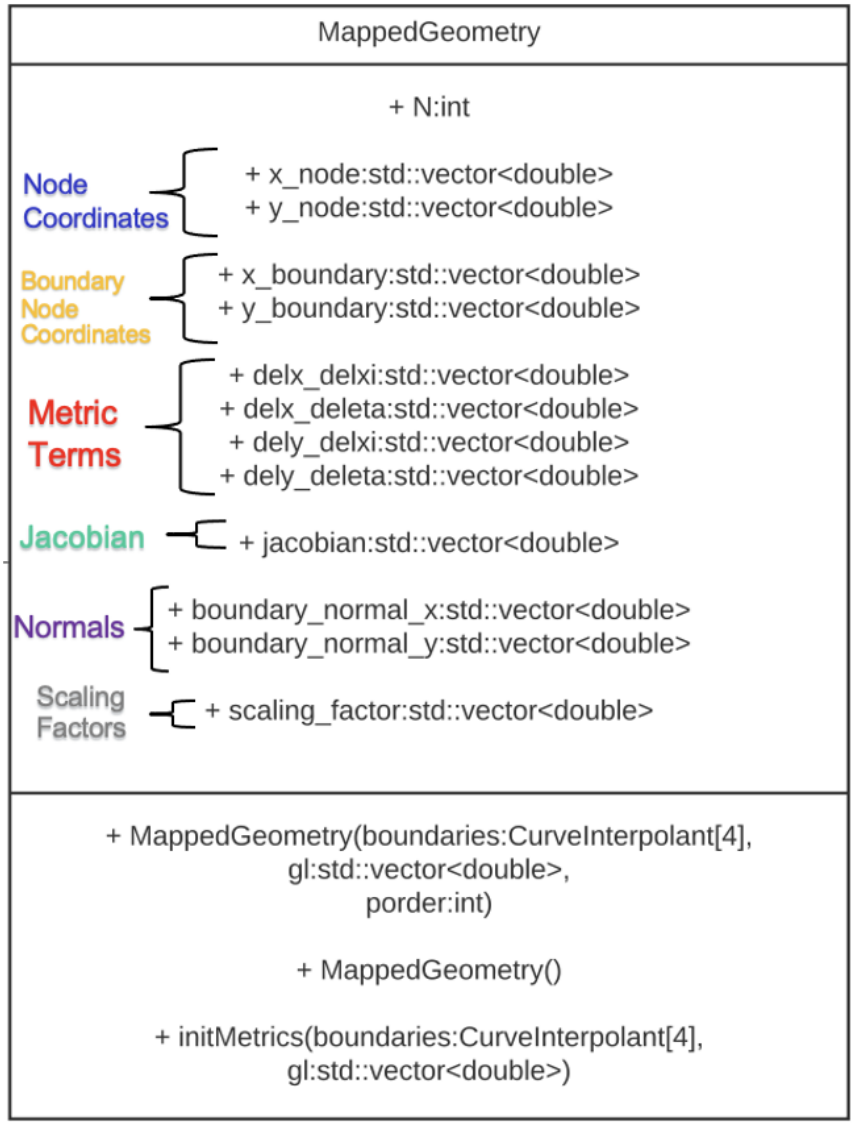


Figure 2.18: Mapped geometry class showing attributes and methods.

During the load-balancing procedure, in addition to sending the solution $O(Np^2)$, we also send each of the attributes in Figure 2.18 including the four metric term arrays $O(Np^2)$, the Jacobian array $O(Np^2)$, the four boundary normal vector arrays $O(Np)$, the four scaling factor arrays $O(Np)$, the two node coordinate arrays $O(Np^2)$ and the four boundary node coordinate arrays $O(Np)$. Comparing what is required for the

Cartesian code, the metric terms, Jacobian, and scaling factors are constant and can be calculated on the fly. The normals are trivial either -1 or +1. The node coordinates can also be obtained very easily given simply the four corners of the element. However, for curvilinear geometries these simplifications cannot be made. The increase in time and memory is due to sending the data for the mapped geometry class.

The transfinite mapping shows promising potential. Despite the additional runtime, the transfinite mapping does allow for the modeling of more complicated geometries. This is desirable as most engineering applications are not simply composed of rectangular geometries. In the future, different implementation approaches will be explored to find something more optimized for parallel computing. Furthermore, a compromise can be obtained by using straight sided elements and unstructured grids similarly to [26]. The DGSEM formulation is well suited for unstructured grids. In this way, the data associated with the transfinite mapping is drastically simplified, and complex geometries can still be modeled.

Another approach which is explored in the next chapter of this thesis is the immersed boundary method. In this method, we can model complex geometries using solely Cartesian grids. Instead of the boundary of the obstacle conforming to the grid, the obstacle is modeled by other means without conforming to the underlying grid. By doing so one can ideally avoid the calculations and storage associated with the mapping. However, the immersed boundary method does come with numerical issues such as a loss of accuracy or oscillations. We aim to address these issues in the next chapter.

Chapter 3

Immersed Boundaries in the Discontinuous Galerkin Spectral Element Method through hp-Adaptivity

The previous chapter modeled complex geometries using a transfinite mapping. However, in this chapter, we take an alternative approach to model complex geometries by using the immersed boundary method with simple Cartesian grids. This paper was submitted to the Computers and Fluids journal, and has been revised based on the reviewers' comments.

Contributions

The author wrote this paper in collaboration with the co-author. The author implemented the immersed boundary method and performed all of the simulations.

Nomenclature

$\tilde{a}_n$	Modal coefficients
c	Wave speed
$\mathfrak{F}$	Flux
F^*	Numerical flux
$\mathbf{f}_{\boldsymbol{x}}, \mathbf{g}_{\boldsymbol{y}}$	Spatial derivatives of flux
J	Temporary variable
k_x, k_y	Wave components
$L_n(x)$	Legendre polynomial
N	Number of elements
$\hat{n}$	Boundary normal
P	Pressure
p	Polynomial order
$\boldsymbol{q}$	Vector of conserved variables
r	Radial coordinate
S_n	Surface
t	Time
u, v	Velocity components
$w^{+,L}, w^{-,R}$	Left and right going waves
$X(x, y)$	Masking function
x, y	Physical space coordinates
$\hat{x}, \hat{y}$	Unit vectors
ϵ	Error
θ	Angle
ξ, η	Computational space coordinates
σ	Modal decay rate
ϕ	Porosity parameter
Ω_n	Domain

Immersed Boundaries in the Discontinuous Galerkin Spectral Element Method through hp-Adaptivity

Amit Nayak and Catherine Mavriplis

[a] *University of Ottawa, Department of Mechanical Engineering, 161 Louis-Pasteur, Ottawa, K1N 6N5, Canada*

Abstract

The immersed boundary method is a promising numerical technique that allows for modeling of complex geometries without the need for body conforming meshes. However, immersed boundary methods present a significant reduction in accuracy. In this paper, we implement the volume penalty method in an hp-adaptive discontinuous Galerkin spectral method framework to solve the two-dimensional acoustic wave equation with immersed boundaries. We demonstrate that combining low porosity, which represents the immersed boundary, with hp-adaptivity reduces oscillations, localizes error to the vicinity of the immersed boundary and improves the overall accuracy. A variety of test cases are presented to show that the implementation is capable of modeling wave propagation in complex geometries with simple Cartesian grids.

Keywords: Discontinuous Galerkin Spectral Element Method, Immersed Boundaries, Volume Penalization, hp-Adaptive Mesh Refinement

Email address: anaya085@uottawa.ca, Catherine.Mavriplis@uottawa.ca (Amit Nayak and Catherine Mavriplis)

1. Introduction

Computational mechanics have drastically risen in popularity due to the recent rise in computing power. Tools such as computational fluid dynamics (CFD), computational acoustics and computational solid mechanics are routinely used for design and analysis of complex engineering systems. Despite the ubiquity of these tools, the meshing process, i.e. the discretization of the computational domain, can still consume a significant amount of user time, often measured in weeks, to set up a reliable calculation [1, 2]. A variety of approaches have been developed over the past few decades to automate and/or alleviate the task, including: unstructured grids e.g. [3, 1, 4]; overset grids e.g. [5]; and body-fitted structured grids e.g. [6]. Despite the variety of meshing techniques, grid generation remains a time-intensive task and therefore a vigorous area of research and development. The CFD Vision 2030 report stated:

"Mesh generation and adaptivity continue to be significant bottlenecks in the CFD workflow, and very little government investment has been targeted in these areas. As more capable HPC [High Performance Computing] hardware enables higher resolution simulations, fast, reliable mesh generation and adaptivity will become more problematic. Additionally, adaptive mesh techniques offer great potential, but have not seen widespread use due to issues related to software complexity, inadequate error estimation capabilities, and complex geometries." [1]

Beyond finding a suitable starting grid for a calculation for which the aim is firstly, to accurately represent the body or boundaries, and secondly, to provide enough resolution to model the initial conditions, grid adaptivity or

adaptive mesh refinement (AMR) [7] [8] can help simplify the grid generation task. Error estimators or indicators can guide a calculation to automatically increase refinement in areas of increasing complexity of the physical phenomena to be modeled, while coarsening or reducing the mesh density in areas where the physical phenomena are relatively simpler.

Many engineering applications involve complex geometries and physical phenomena with a wide range of length scales, particularly in fluid dynamics where turbulent flows are characterized by an energy cascade from larger to smaller eddies over a range of several orders of magnitude. While resolution of these scales can be accomplished by grid refinement, the number of degrees of freedom required scales with Reynolds number, Re, as: $Re^{9/4}$. With Reynolds numbers reaching 10^6 and beyond in practical applications, Direct Numerical Simulation (DNS) by full resolution of all the scales was impractical for many decades and still is for industrial applications. In order to accurately and efficiently resolve this wide range of scales, high order methods were introduced and developed over the last few decades. As an example, spectral methods [9] provide high accuracy through low dissipation and dispersion errors with exponential convergence for smooth solutions. The spectral element method (SEM) [10] was introduced in 1984 to combine the advantages of high-order ("p") methods with the geometric flexibility of finite element ("h") methods in order to handle complex geometries. Through the continuous development of the SEM over the last decades and through the use of Graphical Processing Units (GPUs), SEM methods now routinely calculate DNS in complex geometries and are approaching exascale performance [11].

Nevertheless, gridding complex geometries for high order methods such as the SEM remains a difficult and time-intensive task. Even fewer tools exist for high order methods grids than for low order methods ones. High order method grids use bigger elements than traditional (lower order) finite element, finite difference or finite volume methods since higher order basis functions are used within each element, typically polynomials of degree 5 to 12. Hence coarser elemental meshes are used and either smaller straight-sided elements define the complex boundaries or simple curvature (e.g. circular arc) elements are used to create curved side boundaries. In principle, geometries can be defined by elemental polynomials of the same, or up to the same, degree as the solution through an isoparametric mapping that preserves the exponential convergence of the SEM [12]. This is rarely done, notably because the extra metric terms within the integrals of the variational Galerkin formulation then incur quadrature errors unless they are "overintegrated" which requires extra work.

Grid adaptivity through h-refinement by subdividing (or agglomeration of) elements, e.g. [13] [14], and p-refinement by increasing (or decreasing) spectral (polynomial) order can further increase the flexibility of the high order SEM, e.g. [15, 16] but, as yet, is still not robust enough for big calculations and industrial applications. For ease of programming and, in particular, for HPC applications, many adaptive codes simplify the refinement process to mostly isotropic h-refinement: e.g. splitting elements equally in each direction. Such choices simplify the data handling structures and allow for scaling to hundreds of thousands of processors, as in the case of the popular p4est AMR software library [17], which has been adopted in [13]. Others

use strictly p-refinement to simplify this data structure problem, e.g. [18]. Certainly full hp-refinement is a data structure headache, especially in HPC environments, where dynamically changing grids can wreak havoc with load balancing. In [19], we implemented an effective Hilbert space filling curve approach to dynamic load balancing for full hp-adaptivity for the discontinuous Galerkin SEM. The Hilbert curve preserves locality which is advantageous in grid adaptive procedures.

While the SEM relies on inter-element C^0 continuity, the discontinuous Galerkin SEM (DGSEM) allows for discontinuities through a flux formulation. Since hp-adaptivity incurs inter-element discontinuities due to geometric and/or functional nonconformities, it fits nicely into the DG framework. Nonconformities are handled through the mortar approach which preserves spectral accuracy [20]. The DG approach presents advantages for convection-dominated flows through flux-enforced conservation and for parallel calculations on HPC platforms due to the local nature of the method [21] [22] [23] [19]. Many works have applied DGSEM to compressible fluid flow [24] [25], incompressible fluid flow [26] [22], the shallow water equations [27] [28], and electromagnetics [29]. In this work, we solve the wave equation as a model equation for fluid dynamics and acoustics through an hp-adaptive DGSEM.

Given the above-mentioned challenges to grid generation, we chose to investigate the use of simpler Cartesian grids with immersed boundaries to handle complex geometries in the context of the DGSEM. The immersed boundary method (IBM) was developed by Peskin [30] where the method was applied to simulate the fluid-structure interaction between blood and heart valves. In the IBM, the fluid is discretized using an Eulerian formula-

tion while the immersed object is modeled with a set of Lagrangian points which do not necessarily conform to the computational grid where the boundary forces are applied. Over the years, many different approaches have been developed. The Brinkman penalty method [31] which was originally used for incompressible flow, models obstacles as porous media with low porosity. In the feedback forcing method [32] [33], body forcing is determined using a feedback approach to satisfy the no-slip condition. The projection method [34] algebraically follows the fractional step method [35] [36] where, similarly to the pressure, the boundary force is used as a Lagrange multiplier, making it well suited for simulating incompressible flows. The sharp interface method [37] uses ghost cells inside the immersed boundary to enforce boundary conditions on the surface, showing success for simulating higher Reynolds number flows. In this, our first work in immersed boundaries, we were inspired by the Brinkman penalty method [31] [38] due to its simplicity in software implementation and robustness.

The Brinkman penalty method has been used in low order schemes, such as finite volumes, e.g. [39]. It has also been used in high-order methods such as high-order finite differences [40] [41], wavelet collocation [42] [43], flux reconstruction (FR) [44], and discontinuous Galerkin methods (DG) [38] [45] [46] for a variety of fluid flow regimes, producing very promising results.

In this paper, we explore the combination of the above-mentioned methods to simplify grid generation while providing higher accuracy results for complex geometry simulations. Namely, we introduce a volume penalization immersed boundary method as an alternative to meshing complex geometries in a discontinuous Galerkin spectral element method framework, while

utilizing both h- and p-adaptivity to address loss of accuracy and oscillatory behaviour, without the use of limiters, filters or smoothers.

The remainder of this paper is organized as follows. First, the governing equations and discretization scheme are presented. Next, the hp-adaptation strategy is discussed. The IBM implementation via the volume penalty method for the acoustic wave equation follows. Finally, we demonstrate the behaviour of the method through a variety of numerical examples and discuss our findings.

2. Governing Equations and Discretization

2.1. Acoustic Wave Equation

We consider the two-dimensional acoustic wave equation

$$P_{tt} - c^2 \nabla^2 P = 0 \tag{1}$$

$$u_t = -P_x \tag{2}$$

$$v_t = -P_y \tag{3}$$

where P is pressure, c is the wave speed or speed of sound and u, v are the velocity components. Next, we integrate Eq. (1) with respect to time with proper initial conditions

$$P_t + c^2 \left(u_x + v_y \right) = 0. \tag{4}$$

Writing this as a system of equations,

$$\begin{bmatrix} P \\ u \\ v \end{bmatrix}_t + \begin{bmatrix} 0 & c^2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P \\ u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & 0 & c^2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P \\ u \\ v \end{bmatrix}_y = 0. \tag{5}$$

Since the matrices are constant, Eq. (5) can be written as

$$\mathbf{q}_t + \mathbf{f}_x + \mathbf{g}_y = 0 \tag{6}$$

where $\mathbf{q}$ is the vector of conserved variables $\mathbf{q} = [P\ u\ v]^T$ and $\mathbf{f}_x = [c^2 u_x\ P_x\ 0]^T$, $\mathbf{g}_y = [c^2 v_y\ 0\ P_y]^T$ are the components of the spatial derivative of the flux $\mathfrak{F}$. Finally, we can write Eq. (6) as a conservation law:

$$\mathbf{q}_t + \bigtriangledown \cdot \mathfrak{F} = 0. \tag{7}$$

2.2. Discontinuous Galerkin Spectral Element Method

We use the nodal discontinuous Galerkin spectral element method (DGSEM) to discretize the conservation law (7). First the spatial discretization used to obtain the semi-discrete system will be briefly described, followed by the temporal discretization. Further details on the method can be found in the work of Kopriva [47].

2.2.1. Spatial Discretization

We discretize the domain $\Omega \subset \mathbb{R}^2$ into N non-overlapping two-dimensional quadrilateral elements Ω_n. We consider straight-sided quadrilateral elements in the x-y plane and map from the reference square element $(\xi, \eta) \in [-1; 1]^2$ as shown in Fig 1. The mapping for the four vertices $\mathbf{x}_i$ is:

$$\mathbf{x} = \frac{1}{4}[\mathbf{x}_1(1-\xi)(1-\eta)+\mathbf{x}_2(1+\xi)(1-\eta)+\mathbf{x}_3(1-\xi)(1+\eta)+\mathbf{x}_4(1+\xi)(1+\eta)]. \tag{8}$$

We present the discretization in terms of a reference element Ω_n. The solution is represented as tensor products of Legendre polynomials of degree p in both

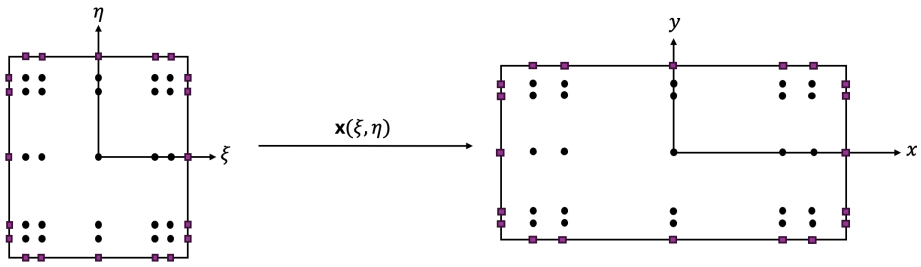


Figure 1: 2D quadrilateral element (right) mapped from reference element (left) with polynomial order $p = 4$. Black dots denote Gauss-Legendre quadrature nodes. Purple squares denote the location of the boundary flux calculations.

the x and y directions in Lagrange form, l_p, for the conserved variables $\mathbf{q}$ and the flux $\mathfrak{F}$ as:

$$\mathbf{q} \approx \mathbf{Q} = \sum_{i=0}^{p} \sum_{j=0}^{p} \mathbf{Q}_{i,j} l_i(x) l_j(y) \tag{9}$$

$$\mathfrak{F} \approx \mathbf{F} = \sum_{i=0}^{p} \sum_{j=0}^{p} \left(\mathbf{F}_{i,j} \widehat{x} + \mathbf{G}_{i,j} \widehat{y} \right) l_i(x) l_j(y). \tag{10}$$

We use Gauss-Legendre nodes as our solution nodes and quadrature nodes. We multiply the conservation law of Eq. (7) by test functions $\phi_{i,j}$ and integrate by parts to give

$$(\mathbf{Q}_t, \phi_{i,j}) + \oint_{S_n} \phi_{i,j} \mathbf{F}^* \cdot \widehat{n} dS - \int \int_{\Omega_n} \mathbf{F} \cdot \bigtriangledown \phi_{i,j} dx dy = 0 \tag{11}$$

where $\mathbf{F}^*$ is the numerical flux and $\widehat{n}$ is the normal vector (normal to the surface S_n). In a Galerkin formulation the test functions are taken from the same function space as the solution. The integrals are calculated by Gauss-Legendre quadrature. Since the DGSEM formulation uses a discontinuous

function space, we must resolve the numerical (boundary) fluxes at each element boundary. For the choice of numerical flux, we utilize the exact Riemann solver given in [47],

$$\mathbf{F}^*(\mathbf{Q}^L, \mathbf{Q}^R, \hat{n}) = \begin{bmatrix} \frac{c}{2}(w^{+,L} - w^{-,R}) \\ \frac{n_x}{2}(w^{+,L} + w^{-,R}) \\ \frac{n_y}{2}(w^{+,L} + w^{-,R}) \end{bmatrix} \tag{12}$$

where $w^{+,L} = Q_1^L + c(n_x Q_2^L + n_y Q_3^L)$, $w^{-,R} = Q_1^R - c(n_x Q_2^R + n_y Q_3^R)$, R refers to the right i.e. the exterior of an element surface as shown in Fig. 2, L to the left, i.e. the interior, n_x and n_y are the x and y components of the normal vector and the subscripts 1, 2, and 3 refer to the components of the vector $\mathbf{Q}$. To implement traditional wall boundary conditions, we enforce $un_x + vn_y = 0$ resulting in the following external state to be used in the Riemann solver,

$$\mathbf{q}^{\text{wall}} = \begin{bmatrix} P \\ (n_y^2 - n_x^2)u - 2n_x n_y v \\ -2n_x n_y u + (n_x^2 - n_y^2)v \end{bmatrix}, \tag{13}$$

where the pressure and velocities are taken from the interior of the domain boundary. To implement radiation boundary conditions, we use a zero external state. The inter-element communication is ensured through the fluxes calculated by the Riemann solver. Since all operations are performed element-wise, the DGSEM formulation is perfectly suited to high-performance computing where portions of the grid are allocated to different processors and communication between processors needs to be minimized to maximize parallel performance.

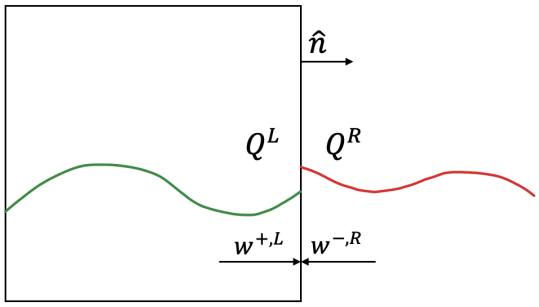


Figure 2: Riemann problem at the right element interface for a 2D quadrilateral element. $\boldsymbol{Q^L}$ is the interior solution interpolated to the element boundary and $\boldsymbol{Q^R}$ is the exterior solution, either from a neighbouring element or the boundary condition. The normal vector is $\hat{n}$ and the right and left going waves are $w^{+,L}$ and $w^{-,R}$, respectively. The green curve represents the solution in the element and the red curve is the solution exterior to the element, i.e. in a neighbouring element.

2.2.2. Temporal Discretization

After spatially discretizing, we employ the explicit third-order low-storage Runge-Kutta scheme (RK3) to integrate in time. The process is as follows. The semi-discrete system produces the ordinary differential equation $\dot{u} = H(u,t)$. The time step is Δt and u^n indicates the solution at the nth time step. Then, to move to the next time step from u^n to u^{n+1}

$$
\begin{aligned}
&u \leftarrow u^n, \\
&J \leftarrow H(u, t_n), \\
&u \leftarrow u + \frac{1}{3}\Delta t J, \\
&J = -\frac{5}{9}J + H\left(u, t_n + \frac{1}{3}\Delta t\right), \\
&u \leftarrow u + \frac{15}{16}\Delta t J, \\
&J \leftarrow -\frac{153}{128}J + H\left(u, t_n + \frac{3}{4}\Delta t\right), \\
&u^{n+1} \leftarrow u + \frac{8}{15}\Delta t J.
\end{aligned}
\tag{14}
$$

The J in Eq. (14) is used as a temporary variable. The table of the coefficients for the time integration method is presented in Table 1.

m	a_m	b_m	g_m
0	0	0	1/3
1	-5/9	1/3	15/16
2	-153/128	3/4	8/15

Table 1: Coefficients of the explicit third-order low storage Runge-Kutta method.

2.3. hp-Adaptivity

DGSEM schemes are well suited to adaptive mesh refinement since they are already discontinuous in nature. In our work, we utilize both h- and p-adaptivity. When h-refining, we split the quadrilateral element into four smaller elements as shown in Fig. 3. Conversely, when coarsening four elements are combined into one larger element. When p-refining, we increase the

polynomial order of the element by two, and when p-coarsening we decrease the polynomial order by two.

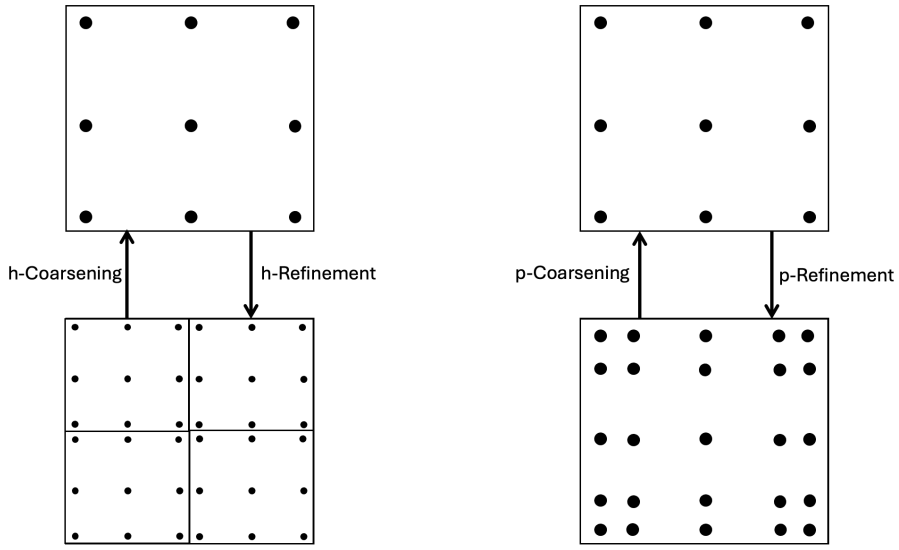


Figure 3: Illustrations of hp-adaptivity for a 2D quadrilateral element with polynomial order $p = 2$. Left: h-refinement (the element is split into four elements) and h-coarsening (four elements are combined into one). Right: p-refinement (polynomial order is increased from $p = 2$ to $p = 4$) and p-coarsening (polynomial order is decreased from $p = 4$ to $p = 2$).

The core driver for the hp-adaptation is the error estimator and smoothness indicator. In 1D, following the approach in [15], we first obtain the modal approximation of the solution

$$u(x) \approx \sum_{n=0}^{p} \widetilde{a}_n L_n(x), \tag{15}$$

where $\widetilde{a}_n$ are the modal coefficients or spectrum and L_n is the nth degree Legendre polynomial. Given the coefficients, we estimate the error as

$$\epsilon_{est} \approx \left(\sum_{n=p+1}^{\infty} \frac{\widetilde{a}_n^2}{\frac{2n+1}{2}} \right)^{1/2}. \tag{16}$$

If the error is above a user-defined upper bound we flag the element for refinement and, conversely, if the error is below a user defined lower bound we flag the element for coarsening. In this work, the error estimator was applied on the pressure solution. The decision to h- or p-refine is based on the following decision process. After obtaining the modal coefficients, we fit the spectrum to an exponential decay,

$$\widetilde{a}_n \approx Ce^{-\sigma n}. \tag{17}$$

If the decay rate $\sigma > 1$ in Eq. (17), the spectrum has an exponential decay rate indicating that the solution is smooth and converging well, and we therefore prescribe p-refinement. However, if $\sigma \leq 1$ then the solution is not properly resolved, i.e. we have not yet locked into the exponential convergence range, and we prescribe h-refinement.

The timestep is chosen for each case such that the CFL number is respected for the smallest possible element (maximum h-level) with the highest possible polynomial order (maximum p-level). The hp-adaptation produces non-conforming interfaces of two categories: geometric (neighbours have different element sizes) and functional (neighbours have different polynomial orders). An example of geometric non-conforming interfaces is presented in Fig. 4. We utilize the mortar element method to handle these non-conforming interfaces, i.e. to preserve the exponential convergence properties of spectral methods, keeping the errors arising from the mismatches to the same order

as, or below, the spectral approximation error. The full details are explained in [20, 48]; here, we briefly describe the technique. The solution $u(s)$ and the

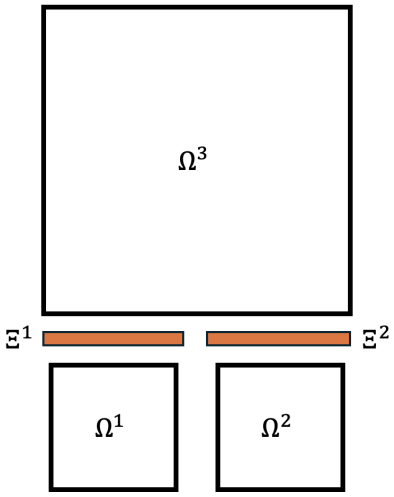


Figure 4: Geometrically non-conforming interfaces caused by h-adaptation. Mortars (orange) are used to preserve the accuracy of the method. Ω denotes the elements and Ξ denotes the mortars.

projection of the solution onto the mortar $\Psi(z)$ are represented as

$$u(s) = \sum_{k=0}^{p} u(s_k) l_k(s), \tag{18}$$

$$\Psi(z) = \sum_{k=0}^{J} \Psi(z_k) l_k(z), \tag{19}$$

where s is the element side coordinate and z is the mortar side coordinate defined with the reference space , $z \in [-1, 1]$. The corresponding coordinate on the element can be computed as $s = a + bz$, where a is the offset and b is a scaling factor. J is taken as the maximum polynomial order between two elements of the non-conforming interface, $J = \max(p^L, p^R)$. We enforce the

following for the projection onto the mortar Ξ

$$\int_{-1}^{1} \left(\Psi(z) - u(a+bz)\right) l_j^{\Xi} dz = 0, \quad j = 0, ..., J. \tag{20}$$

After substituting Equations (18) and (19) we obtain

$$\int_{-1}^{1} \sum_{m=0}^{J} \Psi(z_m) l_m^{\Xi}(z) l_j^{\Xi}(z) dz = \int_{-1}^{1} \sum_{k=0}^{P} u(a+bz_k) l_k^{\Omega}(a+bz) l_j^{\Xi}(z) dz, \quad j = 0, ..., J, \tag{21}$$

where Ω denotes the element. Defining operators M and S as

$$M = \int_{-1}^{1} l_m^{\Xi}(z) l_j^{\Xi}(z) dz, \quad j, m = 0, 1, ..., J, \tag{22}$$

$$S = \int_{-1}^{1} l_k^{\Omega}(a+bz) l_j^{\Xi}(z) dz, \quad j = 0, 1, ..., J, \; k = 0, 1, ..., P, \tag{23}$$

where M is a square matrix of size $(J+1)^2$ and S is a matrix of size $(J+1) \times (P+1)$. Substituting these operators in Equation (21) results in

$$M\Psi = Su. \tag{24}$$

To obtain Ψ, we define the projection matrix $P^{\Omega \to \Xi} = M^{-1}S$ thus resulting in the L_2 projection from the element to the mortar as

$$\Psi = P^{\Omega \to \Xi} u. \tag{25}$$

For the projection of the numerical flux back from the mortar to the element we follow a similar procedure. Taking an example of two mortars communicating with one element we require the following condition

$$\int_{-1}^{o'} \left(F(s) - \Phi(\frac{s-a}{b}) \right) l_j^{\Omega}(s) ds + \int_{o'}^{1} \left(F(s) - \Phi(\frac{s-a}{b}) \right) l_j^{\Omega}(s) ds = 0, \quad j = 0...p, \tag{26}$$

where Φ is the numerical flux calculated on each mortar and $F(s)$ is the numerical flux that we map from the mortar to the element. o' is the s value where the two mortars join. The corresponding coordinate on the element interface can be computed as $z = \frac{s-a}{b}$. Simplifying Equation (26) results in

$$\int_{-1}^{1} F(s) l_j^{\Omega}(s) ds = \int_{-1}^{o'} \Phi(\frac{s-a}{b}) l_j^{\Omega}(s) ds + \int_{o'}^{1} \Phi(\frac{s-a}{b}) l_j^{\Omega}(s) ds. \tag{27}$$

We represent the fluxes as

$$F(s) = \sum_{m=0}^{N} F(s_m) l_m^{\Omega}(s), \tag{28}$$

$$\Phi(\frac{s-a}{b}) = \sum_{m=0}^{J} \Phi(\frac{s_m-a}{b}) l_m^{\Xi}(\frac{s-a}{b}), \tag{29}$$

and substitute into Equation (27) to obtain the expression in operator form:

$$MF = S\Phi. \tag{30}$$

The projection matrix from the mortar to the element is $P^{\Xi\to\Omega} = M^{-1}S$. Thus the L_2 projection of the numerical flux from the mortar to the element is

$$F = P^{\Xi\to\Omega}\Phi. \tag{31}$$

The operator M is the same as in the projection from the element to the mortar

$$M = \int_{-1}^{1} l_m^{\Xi}(z) l_j^{\Xi}(z) dz, \quad j, m = 0, 1, ..., J. \tag{32}$$

The operator S requires a change of variable

$$\int_{-1}^{o'} \Phi(\frac{s-a}{b}) l_j^{\Omega}(s) ds = b \int_{-1}^{1} \sum_{m=0}^{J} \Phi(z_m) l_m^{\Xi} l_j^{\Omega}(a+bz) dz. \tag{33}$$

Note that in this case $S^{\Xi\to\Omega} = b\left(S^{\Omega\to\Xi}\right)^T$. Now we have all the operators for the mortar element method to deal with the non-conforming interfaces.

For our hp-adaptive DGSEM implementation, we extend upon the code by He [19] to implement the immersed boundary method. In that work, a dynamic load balancing algorithm was developed to ensure that the adaptive code scales well on HPC platforms, using a Hilbert space filling curve; further details can be found in [19].

2.4. Immersed Boundary Method

The IBM implementation that we use is a volume penalty method inspired by the Brinkman approach. We define a masking function $X(x, y)$ that has a value of 1 if a node is inside the immersed boundary, or a value of 0 if a node is outside the immersed boundary. An example of this is shown in the left of Fig. 5, where the red dots indicate the nodes within the immersed boundary based on the masking function. Specifically in our hp-adaptive framework, at the beginning of a simulation we flag any elements that have a node within the immersed boundary for refinement, regardless of whether the error estimator deems it unnecessary or not. This is to help capture the immersed boundary geometry with more precision and to mitigate errors and any ensuing oscillatory effects caused by the immersed boundary. As the grid is refined through the hp-adaptive procedure, all new nodes within or on the immersed boundaries are given a masking function value of 1, as shown in red in Fig. 5 in the right image.

We introduce a porosity parameter ϕ to model walls as porous obstacles. As $\phi \to 0$, this would imply an object that has no porosity, essentially an idealized wall. Thus, for walls, the porosity ϕ must be very small. Specifically

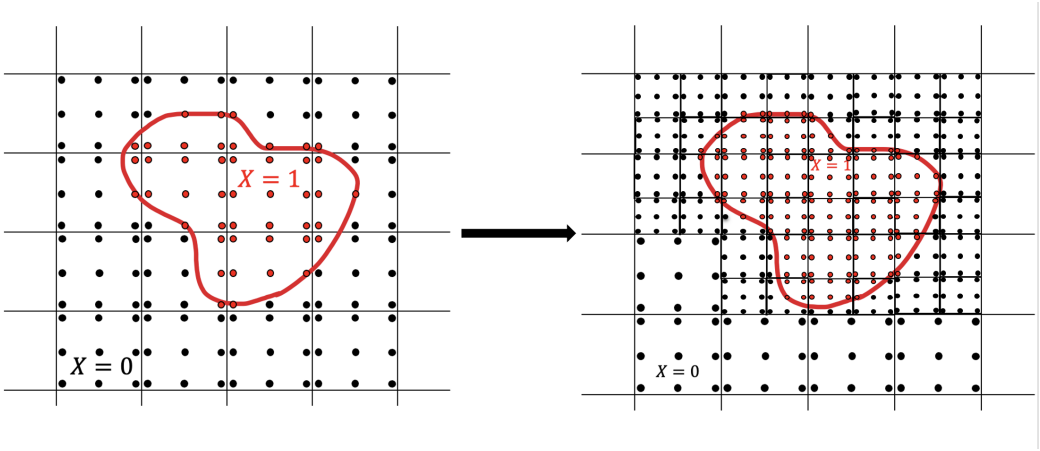


Figure 5: Masking function values for the immersed boundary technique. The red curve represents the immersed boundary. The red dots represent the Gauss-Legendre quadrature nodes on, or inside, the immersed boundary, while the black ones are outside the immersed boundary. Left: Current grid. Right: Adapted grid.

for the acoustic wave equation, we penalize the gradient of the pressure which is equal to the time derivative of the velocity components ($u_t = -P_x$, $v_t = -P_y$) during the temporal integration

$$u_t(1 + \frac{X\Delta t}{\phi}) = -P_x, \tag{34}$$

$$v_t(1 + \frac{X\Delta t}{\phi}) = -P_y. \tag{35}$$

While the Brinkman approach for the Navier-Stokes equations of [38] penalizes the velocity in the momentum equations (with a permeability factor) and the momentum in the mass conservation equation (with a porosity factor), here we penalize the gradient of pressure in the wave equation with only one parameter, which we call the porosity. In this sense, our approach is closer to that of Paccou et al. [49] who penalize pressure itself, with a porosity in the wave equation. As mentioned previously, we employ a low-storage

explicit RK3 for the temporal integration. However, for the sake of clarity, we demonstrate this below on a Euler forward (explicit) time step:

$$u(t+\Delta t) = u(t) + \Delta t \frac{-P_x}{1+\frac{X\Delta t}{\phi}}, \tag{36}$$

$$v(t+\Delta t) = v(t) + \Delta t \frac{-P_y}{1+\frac{X\Delta t}{\phi}}. \tag{37}$$

When the masking function $X = 0$, we recover the standard time integration procedure. However, when the masking function $X = 1$, the volume terms are penalized, the level of which is controlled by the porosity value ϕ. Note that in Eqs.(36) the porosity parameter ϕ is scaled with Δt since a lower porosity value can be used as the time step decreases (i.e. with more accurate time integration), whereas low porosity values should not be used with larger time steps in order to avoid instabilities.

The overall hp-adaptive immersed boundary DGSEM algorithm is given in Fig. 6.

3. Numerical Results

3.1. Wave Reflection Off a Plane Wall

In the following test case, we model a plane wall using the immersed boundary implementation. The domain is the unit square $\Omega = [-1, 1]^2$, with the right half of the square $x \geq 0$ as the immersed boundary. The initial conditions are a Gaussian plane wave propagating towards the wall at the right side of the domain:

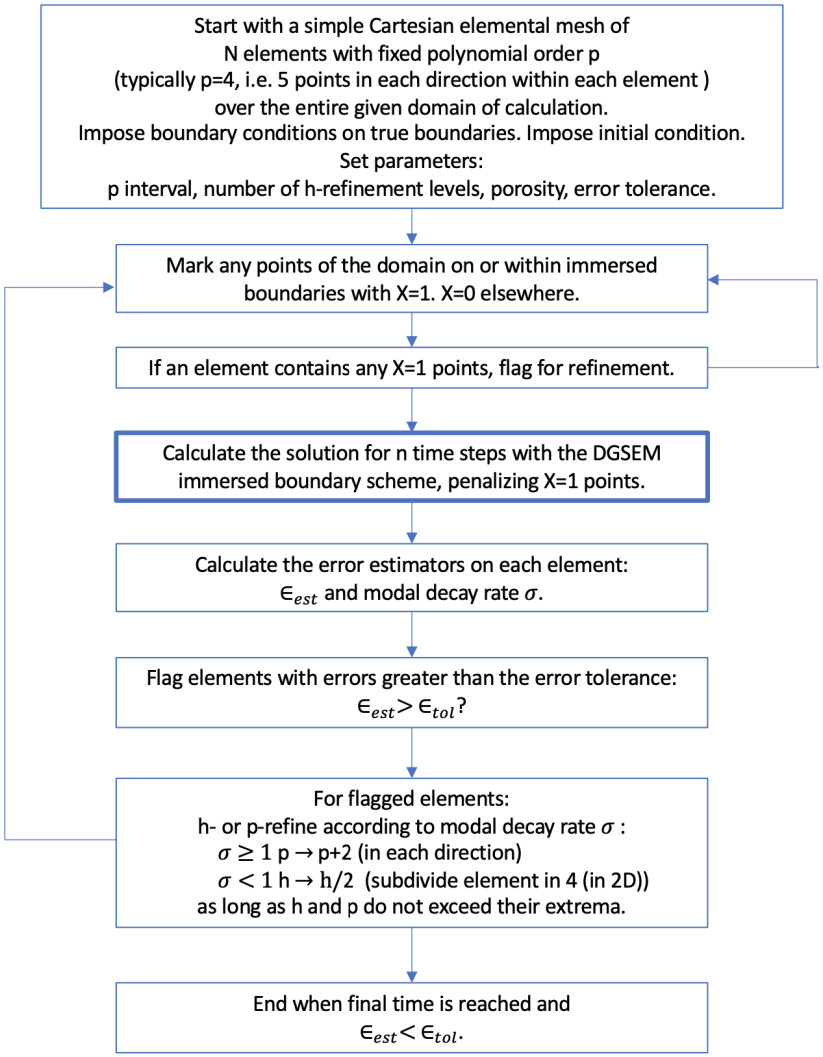


Figure 6: Flowchart for the hp-adaptive immersed boundary DGSEM algorithm.

$$\begin{aligned} P(x,y,0) &= e^{-ln(2)(\frac{(x+0.2)^2}{0.05^2})} \\ u(x,y,0) &= e^{-ln(2)(\frac{(x+0.2)^2}{0.05^2})} \\ v(x,y,0) &= 0. \end{aligned} \tag{38}$$

Wall boundary conditions are used for the left, top and bottom boundaries. The domain is discretized with $N = 32 \times 32 = 1024$ elements with a polynomial order of $p = 4$. For this and all cases, we set the minimum polynomial order to 4, as we need to sufficiently resolve the initial condition wave and its passing through the domain. Furthermore, the error estimators require at least 4 modes to estimate the error and calculate the slope of the modal decay. The simulation is run with a fixed time step of $\Delta t = 4 \times 10^{-5}$ until a final time of $t = 0.4$, such that the Gaussian plane wave returns to its original position. We choose fixed time steps that obey the CFL condition for all tests in order to isolate the spatial errors. First, we vary the porosity value $\phi \in [10^{-4}, 10^{-5}, 10^{-6}, 10^{-8}, 10^{-10}]$ and observe the effects. The pressure profiles along the horizontal centerline of $y = 0$ are shown in Fig. 7 for the varying porosity values up till $\phi = 10^{-8}$.

We observe that as the porosity value is decreased, the amplitude of the reflection approaches the exact solution. Thus, a lower porosity value is desired to better approximate a solid wall boundary. However, we also observe a trend of increased oscillations within the vicinity of the immersed boundary as we lower the porosity value. The variation of the L_2 error with respect to the exact solution is plotted in Fig. 8 as a function of porosity. While the error decreases due to the better approximation of the solid wall, undesirable artifacts/oscillations near the immersed boundary remain problematic. It should also be noted that the reduction in error plateaus beyond $\phi = 10^{-8}$, as was reported in [38], [39] and [49]. From our numerical experiments, we observe a convergence rate of 0.26 with respect to ϕ, whereas [38], [39] and [49] show a theoretical convergence of 0.5 (square root of permeability).

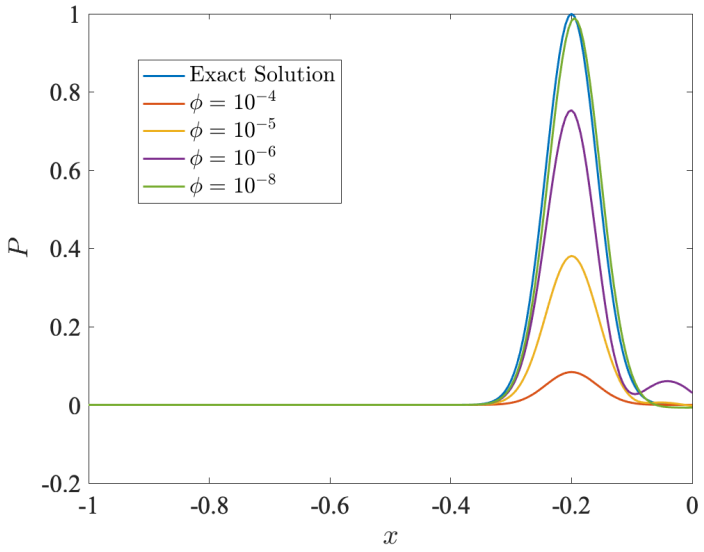


Figure 7: Pressure profiles for the plane wall reflection case at $t = 0.4$ with varying porosity values using the non-adaptive DGSEM immersed boundary method. $N = 32 \times 32 = 1024$ elements, $p = 4$, $\Delta t = 4 \times 10^{-5}$.

Next, we perform an h-refinement spatial convergence test. We use a polynomial order of $p = 4$ and start with the coarsest element size of $h = 0.0625$ ($N = 32 \times 32 = 1024$ elements). We use a porosity value of $\phi = 10^{-6}$. The L_2 error is computed with respect to the exact solution and plotted in Fig. 9. The results illustrate the disadvantage of immersed boundaries: the accuracy does not follow the theoretical accuracy of the spatial discretization scheme. Next, we test the effect of changing the polynomial order. The domain is discretized with $N = 32 \times 32 = 1024$ elements and the polynomial order is varied: $p \in [5, 6, 7, 8]$. We again use a porosity value of $\phi = 10^{-6}$. Figure 10 shows a negligible change in the reflected wave amplitude from increasing the polynomial order. We note that the oscillations are localized

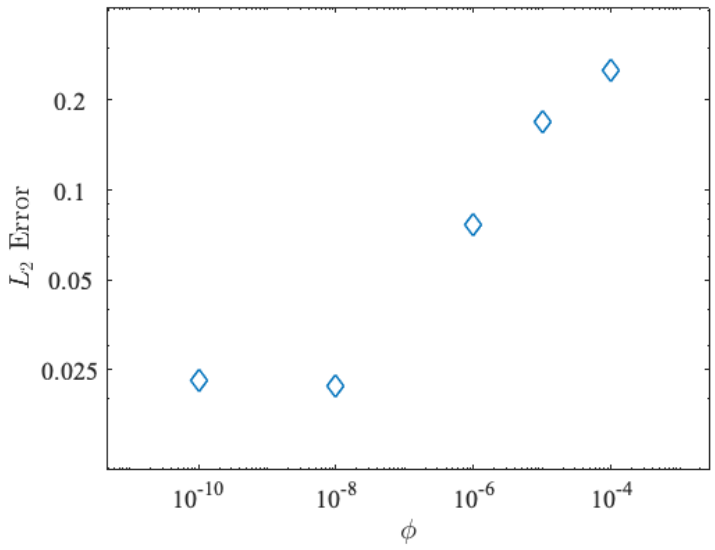


Figure 8: Variation of L_2 error with decreasing porosity value for the plane wall reflection case. Parameters as in Fig. 7.

to the vicinity of the immersed boundary, limiting corruption of the solution beyond the reflected wave. The L_2 error is computed with respect to the exact solution and is plotted in Fig. 11. There is a negligible change in error despite increasing the polynomial order.

To remedy these issues, we utilize hp-adaptivity and decrease the porosity value. The rationale behind the approach is to lower the porosity value to produce more accurate wall reflections, while using hp-adaptivity to reduce the oscillations near the immersed boundary and provide more resolution where required. We run an adaptive simulation starting with $N = 32 \times 32 = 1024$ elements with a polynomial order of $p = 4$. The polynomial order can be refined up to $p = 8$, we allow for two levels of h-refinement, and the error tolerance is set to $\epsilon = 10^{-6}$. We use a porosity value of $\phi = 10^{-8}$. We

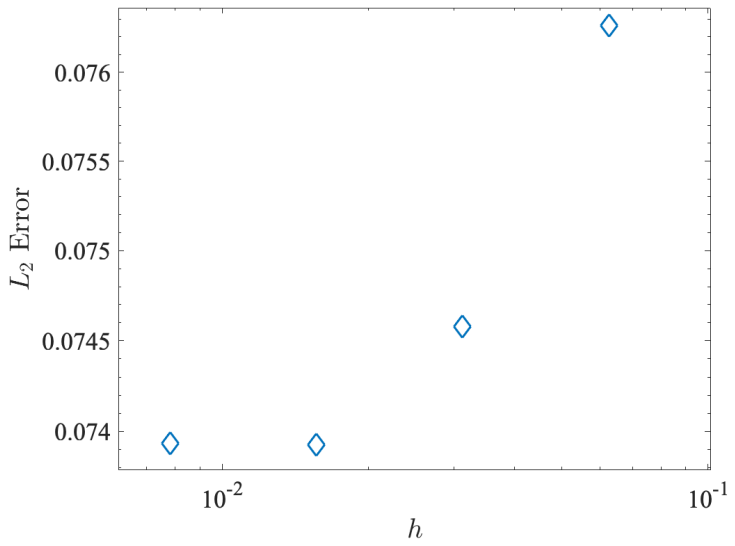


Figure 9: Uniform h-refinement spatial convergence test for the immersed boundary plane wall reflection case. $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p = 4$, $\Delta t = 4 \times 10^{-5}$.

compare results in Fig. 12 with a non-adaptive case using $p = 4, \phi = 10^{-6}$, and the element size is the same as the smallest element size in the adaptive case ($N = 128 \times 128 = 16{,}384$).

Fig. 12 demonstrates that the combination of decreasing the porosity value and using hp-adaptivity is successful in increasing the accuracy of the immersed boundary approach. We obtain the benefits of using a very low porosity value while mitigating the trade-off of spurious oscillations in the vicinity of the immersed boundary. The h-adapted grid of the adaptive case with a porosity of $\phi = 10^{-8}$ is shown in Figure 13.

We also present the non-adaptive cases with a porosity of $\phi = 10^{-8}$ for polynomial orders of $p = 4$ and $p = 6$, and the adaptive case with a higher porosity of $\phi = 10^{-6}$ for comparison purposes. The errors are computed

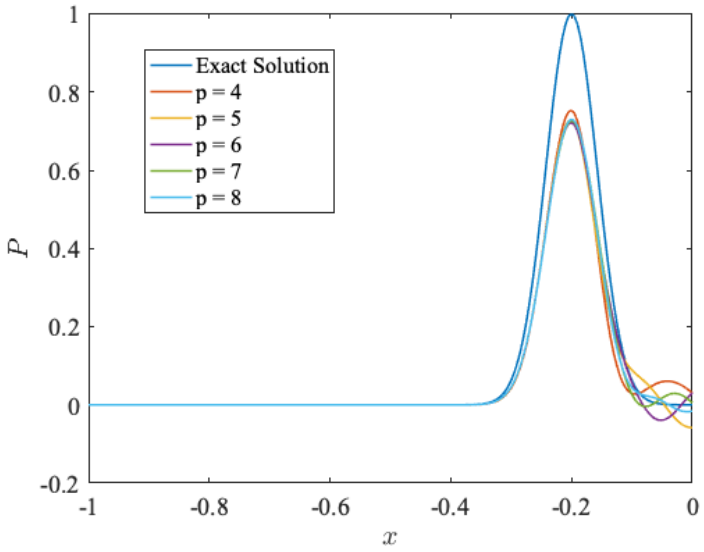


Figure 10: Pressure profiles at $t = 0.4$ for the immersed boundary plane wall reflection case for increasing polynomial orders. $N = 32 \times 32 = 1024$ elements, $p \in [5, 6, 7, 8]$, $\Delta t = 4 \times 10^{-5}$.

and presented in Table 2. CPU times for each of the cases and relevant performance metrics from the work of [50] are given in Table 3. The hp-adaptive case with a porosity of $\phi = 10^{-8}$ produces the lowest error. It should be noted that all of the cases with the porosity value of $\phi = 10^{-8}$ have an error an order of magnitude lower than all of the cases with a porosity of $\phi = 10^{-6}$. However, Table 3 shows the benefits of using hp-adaptivity. Despite the fact that non-adaptive cases with a porosity of $\phi = 10^{-8}$ produce an error roughly the same as the adaptive case with the same porosity, the CPU time for the adaptive case is much lower (a savings of 88%). Thus, hp-adaptivity improves the accuracy of the immersed boundary in an efficient manner. This can be further quantified when looking at the ratio of CPU

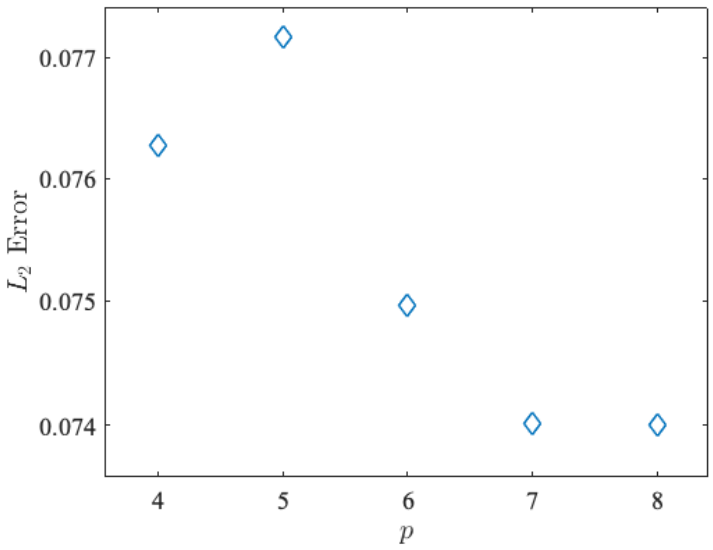


Figure 11: p-Refinement spatial convergence test for the immersed boundary plane wall reflection case. Parameters as in Fig. 10.

Table 2: L_2 error for the planar wall reflection immersed boundary case, comparing hp-adaptive to non-adaptive cases with low and moderate porosity.

Case			**Degrees of Freedom**	**L_2 Error**
# of Elements	hp-Adaptive	Porosity ϕ		
$N_{\text{fixed}} = 16384$	No, $p = 4$	$\phi = 10^{-6}$	409600	7.39×10^{-2}
$N_{\text{fixed}} = 16384$	No, $p = 4$	$\phi = 10^{-8}$	409600	7.49×10^{-3}
$N_{\text{fixed}} = 16384$	No, $p = 6$	$\phi = 10^{-8}$	802816	8.82×10^{-3}
$N_{\text{initial}} = 1024$	Yes, $p \in [4, 6, 8]$	$\phi = 10^{-6}$	567238	7.39×10^{-2}
$N_{\text{initial}} = 1024$	Yes, $p \in [4, 6, 8]$	$\phi = 10^{-8}$	553108	7.33×10^{-3}

time to error. Comparing this metric for cases with the same porosity for non-adaptive vs hp-adaptive cases, we observe a lower time to error ratio

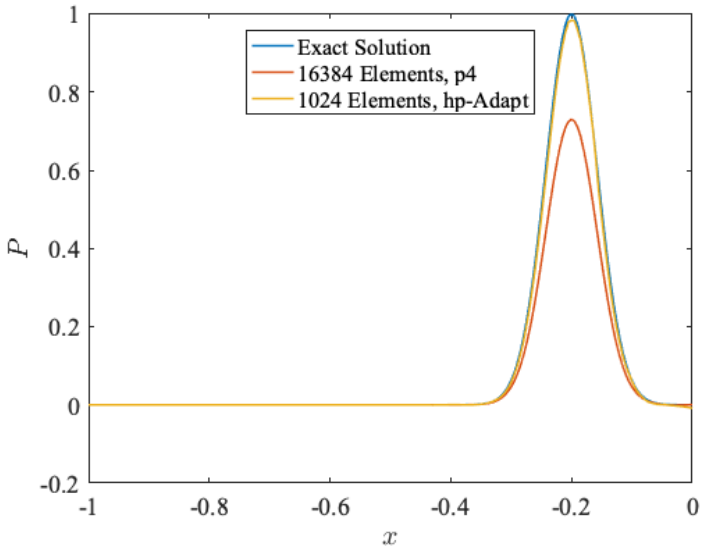


Figure 12: Pressure profiles at $t = 0.4$ for the hp-adaptive (yellow) immersed boundary and the non-adaptive (red) immersed boundary solutions for the plane wall reflection case. Non-adaptive: $N = 128 \times 128 = 16{,}384$ elements, $p = 4$, $\Delta t = 4 \times 10^{-5}$, $\phi = 10^{-6}$. Adaptive: $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p \in [4, 6, 8]$, $\Delta t = 4 \times 10^{-5}$, $\phi = 10^{-8}$. Exact solution in blue.

(one order of magnitude lower). Thus, to reach the same level of error, hp-adaptivity is much more efficient. The overall efficiency of the hp-adaptive simulations can also be demonstrated from the time per degree of freedom ratios (one order of magnitude lower). The hp-adaptive simulations are thus more efficient per degree of freedom. Finally, the error per degree of freedom ratio is an order of magnitude lower when the porosity is $\phi = 10^{-8}$ compared to $\phi = 10^{-6}$. This trend is consistent with the fact that a lower porosity value allows for a better representation of a solid wall.

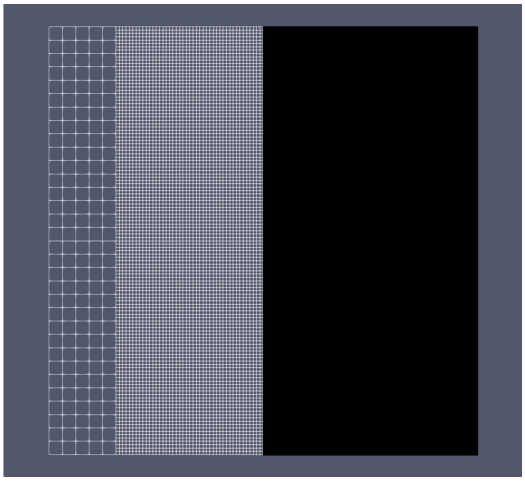


Figure 13: Adapted grid at $t = 0.4$ for the hp-adaptive immersed boundary (black is the immersed boundary): $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p \in [4, 6, 8]$, $\Delta t = 4 \times 10^{-5}$, $\phi = 10^{-8}$.

3.2. Acoustic Scattering off a Circular Cylinder

The following case tests curved immersed boundaries. Kopriva [47] suggests this case as a model for the acoustic pressure loading on an aircraft fuselage (the cylinder) from a nearby engine source (on the wing). The domain is rectangular: $\Omega = [-5, 5] \times [0, 5]$ with a half-circle wall of radius r_0 of 0.5 centered on the bottom wall (at x=0, y=0) (see Fig. 14). The cylinder and the source are symmetric about the x axis, so only the upper half plane is simulated and the lower horizontal boundary has a symmetry boundary condition. In the farfield we simply set the state to zero to simulate a radiation boundary condition and stop the calculation before the perturbation reaches it. The initial conditions are a pressure perturbation centered at

Table 3: CPU times and relevant performance metrics for the adaptive and non-adaptive immersed boundary plane wall reflection cases.

Case		**CPU Time**	$\frac{\textbf{Error}}{\textbf{DOF}}$	$\frac{\textbf{Time}}{\textbf{Error}}$	$\frac{\textbf{Time}}{\textbf{DOF}}$
hp-Adaptive	Porosity ϕ	**(Core Hours)**			
No, $p = 4$	$\phi = 10^{-6}$	933	1.81×10^{-7}	1.26×10^{4}	2.28×10^{-3}
No, $p = 4$	$\phi = 10^{-8}$	930	1.83×10^{-8}	1.24×10^{5}	2.28×10^{-3}
No, $p = 6$	$\phi = 10^{-8}$	1627	1.10×10^{-8}	1.84×10^{5}	2.03×10^{-3}
Yes, $p \in [4, 6, 8]$	$\phi = 10^{-6}$	205	1.30×10^{-7}	2.77×10^{3}	3.61×10^{-4}
Yes, $p \in [4, 6, 8]$	$\phi = 10^{-8}$	198	1.33×10^{-8}	2.70×10^{4}	3.58×10^{-4}

$x = 1.5, y = 0$:

$$\begin{aligned} P(x, y, 0) &= e^{-ln(2)(\frac{(x-1.5)^2+y^2}{0.125^2})} \\ u(x, y, 0) &= 0 \\ v(x, y, 0) &= 0. \end{aligned} \tag{39}$$

A high resolution solution is run on a transfinite mapped grid implementation following the procedure of [47] (see details in [51], also based on the code of He [19]) with N = 32 (in r) x 32 (in θ) = 1024 elements and polynomial order $p = 8$. r and θ are the cylindrical coordinates centred at the centre of the circular cylinder, with θ increasing counter-clockwise from the x axis. The respective grids are visualized in Fig. 14. Pressure profiles along the line $0.5 \leq r \leq 5$, $\theta = 0^\circ$ are used for the comparison. The simulation is run until a final time of $t = 3$.

First, we vary the porosity value $\phi \in [10^{-4}, 10^{-5}, 10^{-6}]$. The domain is discretized with $N = 32 \times 32 = 1024$ elements with a polynomial order of $p =$

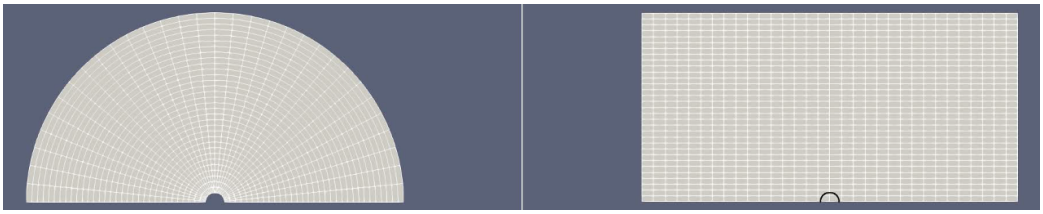


Figure 14: Comparison of the immersed boundary grid (right) and the transfinite mapped grid (left) for the acoustic scattering off a circular cylinder. The black line on the right figure indicates the immersed boundary for the circular cylinder.

4 and the simulation is run with a time step of $\Delta t = 5 \times 10^{-5}$. The L_2 error decreases as the porosity decreases, as shown in Fig. 15. However, decreasing the porosity comes with a trade off as shown in Fig. 16: the solution recovers some of the amplitude of the reflected wave but exhibits more oscillations in proximity to the immersed boundary. The error converges with porosity with an experimental convergence rate of 0.15. As before in the plane wall reflection case, the error plateaus beyond $\phi = 10^{-6}$.

Next, we fix the porosity value at $\phi = 10^{-6}$ with a polynomial order of $p = 4$ and uniformly refine the element size starting with a grid size of $N = 16 \times 16 = 256$ elements. In Figure 17, the immersed boundary implementation displays the common issue where the order of accuracy does not match the spatial accuracy of the h-refinement scheme.

To demonstrate the benefit of using adaptivity, we run the case with h- and p-adaptive mesh refinement. The grid initially has $N = 32 \times 32 = 1024$ elements and a polynomial order of 4. We allow two levels of h-refinement and p-refinement up to $p = 8$ with a error tolerance of $\epsilon = 10^{-6}$. The solution is compared to a non-adaptive simulation with the same starting grid in Fig. 18. Both simulations use a porosity value of $\phi = 10^{-6}$.

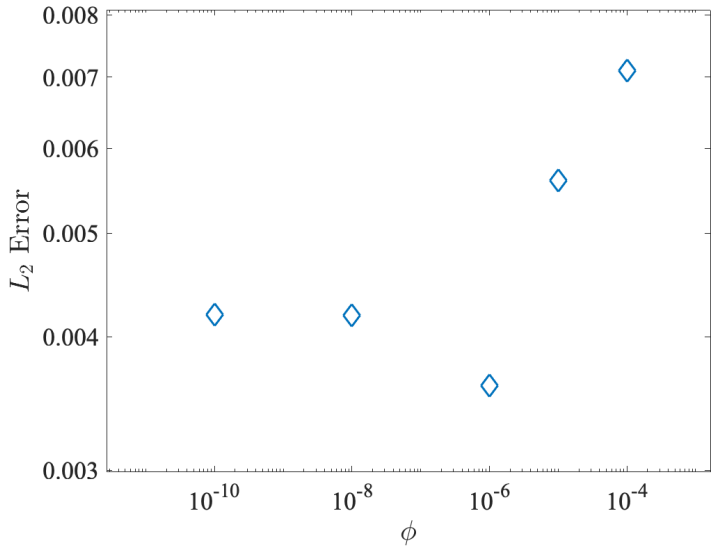


Figure 15: Variation of L_2 error with decreasing porosity value for the acoustic scattering off a circular cylinder case. $N = 32 \times 32 = 1024$ elements, $p = 4$, $\Delta t = 5 \times 10^{-5}$.

We observe that adaptivity significantly reduces the oscillations upstream of the reflected wave. However, it should be noted that the reflected wave in the adaptive solution has a lower amplitude. Thus, hp-adaptivity allows for the use of lower porosity values, and hence a more accurate representation of the wall. As in the plane wall case, we lower the porosity value to $\phi = 10^{-8}$ to more accurately model the half-circle wall boundary, while utilizing hp-adaptivity to reduce the oscillations. The simulation starts with $N = 32 \times 32 = 1024$ elements and polynomial order $p = 4$. We adapt every 50 time steps, with a error tolerance of $\epsilon = 10^{-6}$, two levels of h-refinement and p-refinement up to $p = 8$. The simulation run with hp-adaptivity and $\phi = 10^{-8}$ is compared to a case where the element size is the same as the smallest element in the adaptive case $N = 128 \times 128 = 16{,}384$ elements with

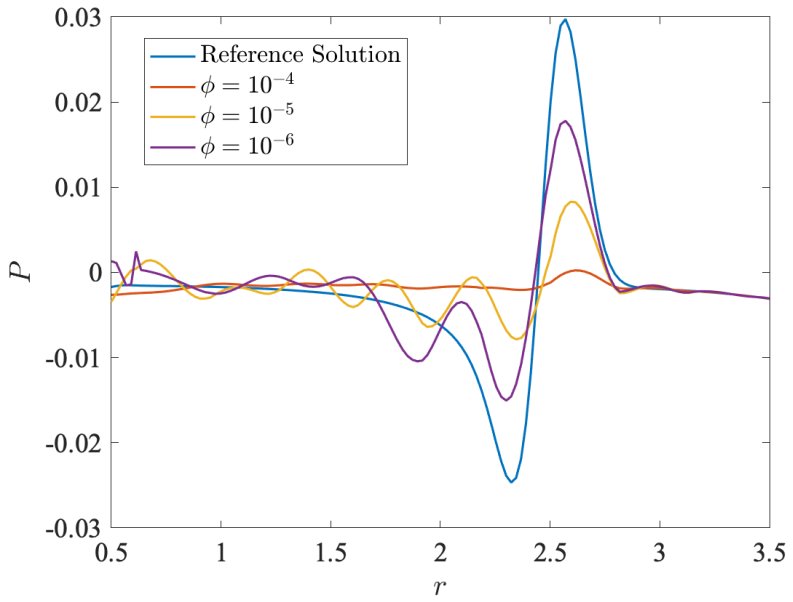


Figure 16: Closeup of the reflected wave for varying porosity values at $t = 3$, $\theta = 0$, $0.5 \leq r \leq 3.5$. High resolution transfinite mapping (reference) solution in blue. Parameters as in Figure 15.

a polynomial order $p = 4$ and a porosity of $\phi = 10^{-6}$. The hp-adaptive case shows a drastic improvement in Fig. 19. The oscillations upstream of the reflected wave are heavily diminished for the adaptive simulation. Combining the adaptivity with a low porosity value allows the reflected wave amplitude to match much more closely with the reference solution, as compared to the non-adaptive case.

As in the plane wall case, we run non-adaptive cases with a porosity of $\phi = 10^{-8}$ with a polynomial order of $p = 4$ and $p = 6$, and the adaptive case with a larger porosity of $\phi = 10^{-6}$. The smallest element size in the adaptive case is the same element size in the non-adaptive case. The errors are presented in Table 4 with the CPU times and relevant performance metrics in Table 5.

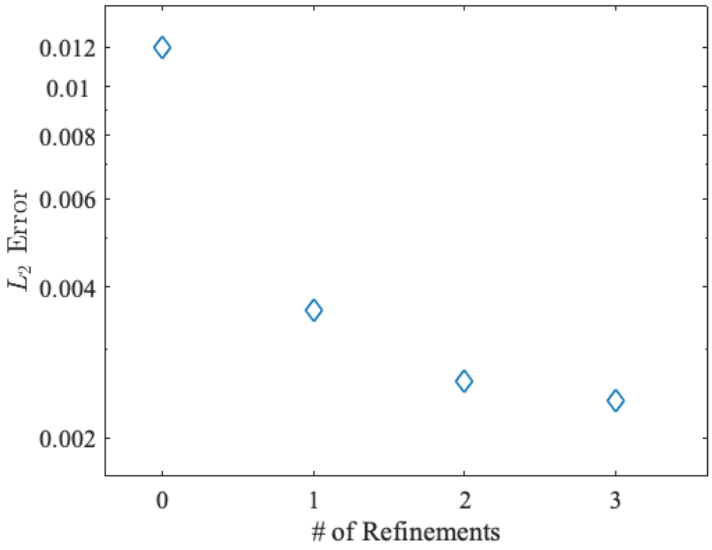


Figure 17: Uniform h-refinement spatial convergence test for the acoustic scattering of cylinder case. $N_{\text{initial}} = 16 \times 16 = 256$ elements, $p = 4$, $\Delta t = 5 \times 10^{-5}$, $\phi = 10^{-6}$.

Table 4: L_2 error for the acoustic scattering off a circular cylinder case, comparing low porosity with hp-adaptivity to moderate porosity and no adaptivity.

Case			**Degrees of Freedom**	**L_2 Error**
# of Elements	hp-Adaptive	Porosity ϕ		
$N_{\text{fixed}} = 16384$	No, $p = 4$	$\phi = 10^{-6}$	409600	2.4×10^{-3}
$N_{\text{fixed}} = 16384$	No, $p = 4$	$\phi = 10^{-8}$	409600	1.4×10^{-3}
$N_{\text{fixed}} = 16384$	No, $p = 6$	$\phi = 10^{-8}$	802816	9.0×10^{-4}
$N_{\text{initial}} = 1024$	Yes, $p \in [4, 6, 8]$	$\phi = 10^{-6}$	497452	2.39×10^{-3}
$N_{\text{initial}} = 1024$	Yes, $p \in [4, 6, 8]$	$\phi = 10^{-8}$	496165	6.4×10^{-4}

As in the plane wall case, the hp-adaptive case with a porosity of $\phi = 10^{-8}$ produces the lowest error. In Table 5, we observe that the CPU times for

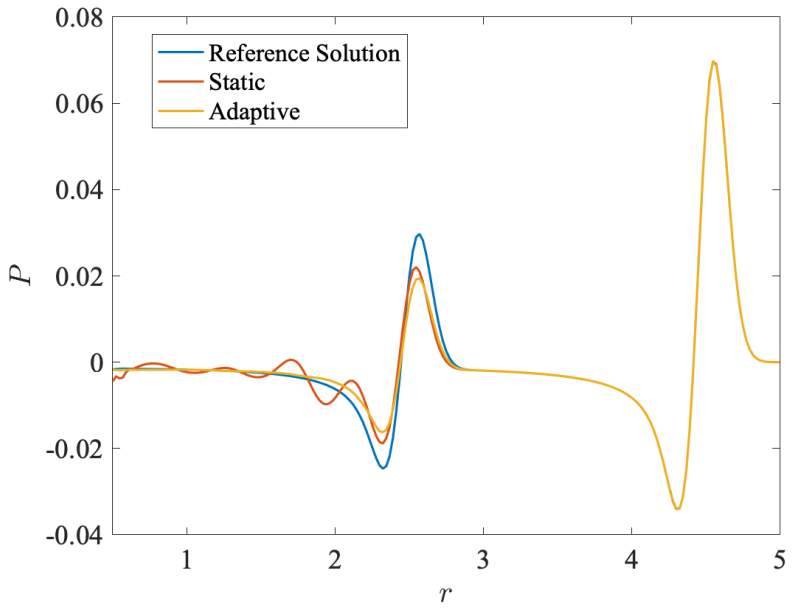


Figure 18: Comparison of pressure profiles at $0.5 \leq r \leq 5$, $\theta = 0$, t = 3, in the acoustic scattering off a circular cylinder case with (yellow) and without (red, "Static") hp-adaptivity, both with a porosity of $\phi = 10^{-6}$. Reference solution in blue.

the adaptive cases are much lower (a savings of 83% for the lowest error). Comparing the ratio of CPU time to error for cases with the same porosity for non-adaptive vs. hp-adaptive cases, we observe a lower time to error ratio. Therefore, the same trends as the plane wall case hold here in terms of both the increase in accuracy and improved efficiency when using hp-adaptivity. The hp-adaptive simulations appear to be more efficient per degree of freedom with lower time to degree of freedom ratios. The error to degree of freedom ratio is again lower when the porosity is $\phi = 10^{-8}$ compared to $\phi = 10^{-6}$. Overall, the same trends hold as in the previous case, however the gains are not as large. This is to be expected as this case is a more challenging problem.

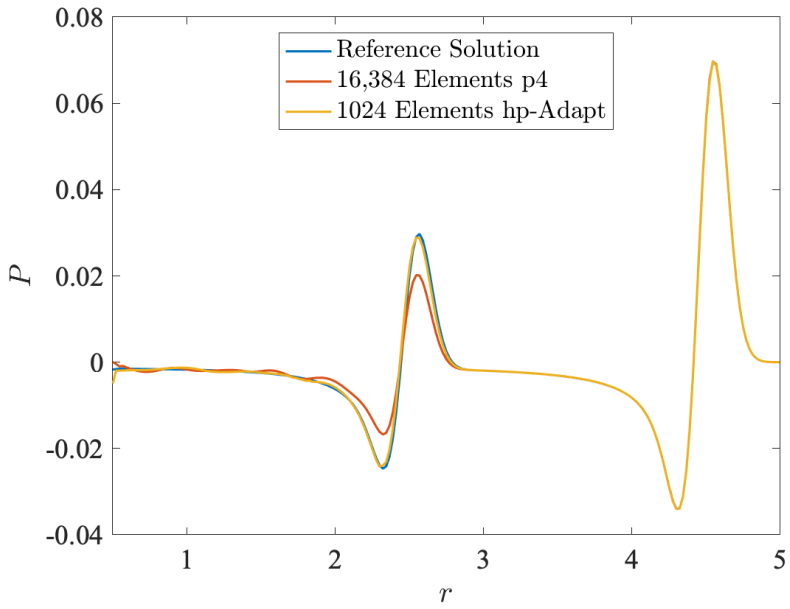


Figure 19: Pressure profile comparison at $0.5 \leq r \leq 5$, $\theta = 0$, t = 3, for the hp-adaptive (yellow) immersed boundary and non-adaptive (red) immersed boundary solutions for the acoustic scattering off a circular cylinder case. Reference solution in blue. Non-adaptive: $N = 128 \times 128 = 16{,}384$ elements, $p = 4$, $\Delta t = 5 \times 10^{-5}$, $\phi = 10^{-6}$. Adaptive: $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p \in [4, 6, 8]$, $\Delta t = 5 \times 10^{-5}$, $\phi = 10^{-8}$.

The plots of the pressure fields for both the transfinite mapped grid calculation and the hp-adapted immersed boundary calculation are presented in Figs. 20 and 21 respectively. The immersed boundary implementation shows strong qualitative agreement with the reference transfinite mapping solution. The final adapted grid for the immersed boundary case is shown in Figs. 22 (polynomial order distribution) and 23 (h-refined elemental grid). We observe that the grid adaptation follows the path of the wave propagation well, with more p-refinement occurring downstream of the initial outgoing wave.

Table 5: CPU times and relevant performance metrics for the adaptive and non-adaptive immersed boundary circular cylinder case.

Case		**CPU Time**	$\frac{\textbf{Error}}{\textbf{DOF}}$	$\frac{\textbf{Time}}{\textbf{Error}}$	$\frac{\textbf{Time}}{\textbf{DOF}}$
hp-Adaptive	Porosity ϕ	**(Core Hours)**			
No, $p = 4$	$\phi = 10^{-6}$	1162	5.86×10^{-9}	4.84×10^{5}	2.84×10^{-3}
No, $p = 4$	$\phi = 10^{-8}$	1184	3.42×10^{-9}	8.46×10^{5}	2.89×10^{-3}
No, $p = 6$	$\phi = 10^{-8}$	2720	1.12×10^{-9}	3.02×10^{6}	3.39×10^{-3}
Yes, $p \in [4, 6, 8]$	$\phi = 10^{-6}$	459	4.80×10^{-9}	1.92×10^{5}	9.23×10^{-4}
Yes, $p \in [4, 6, 8]$	$\phi = 10^{-8}$	463	1.29×10^{-9}	7.23×10^{5}	9.33×10^{-4}

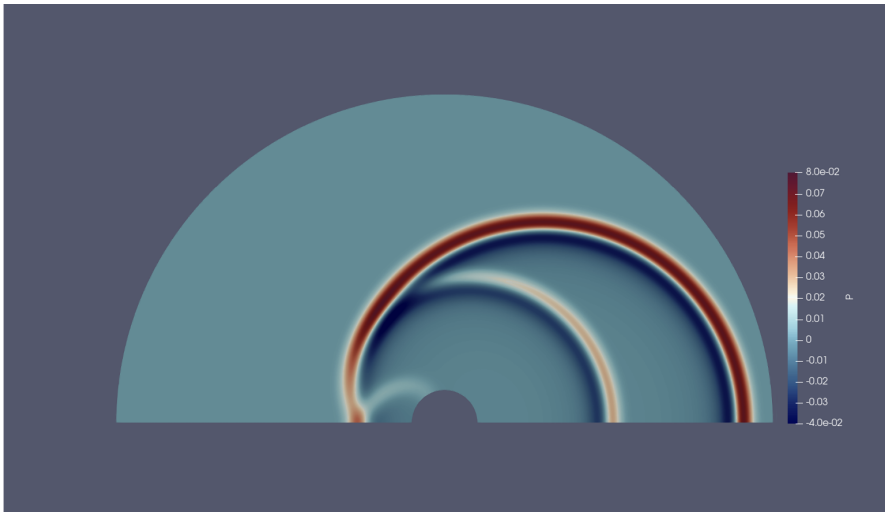


Figure 20: Pressure at $t = 3$ for the acoustic scattering off a circular cylinder case using the transfinite mapping implementation. $N = 32 \times 32 = 1024$ elements, $p = 8$.

3.3. Acoustic Scattering off a NACA 0012 Airfoil

Now that the benefits of combining hp-adaptivity with low porosity values have been demonstrated, we apply our immersed boundary implementation to more complicated geometries. In this test case, we model a Gaussian plane

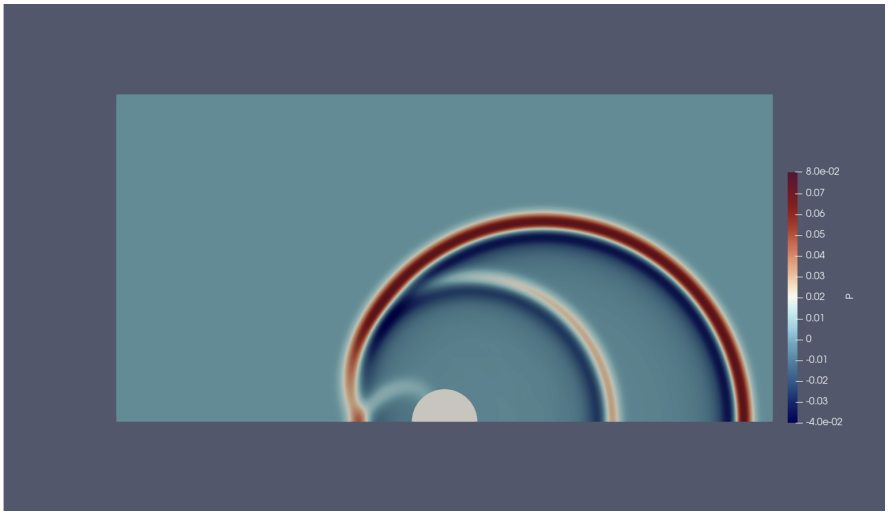


Figure 21: Pressure at $t = 3$ for the acoustic scattering off a circular cylinder case using the hp-adaptive immersed boundary implementation. Parameters as in Fig. 19.

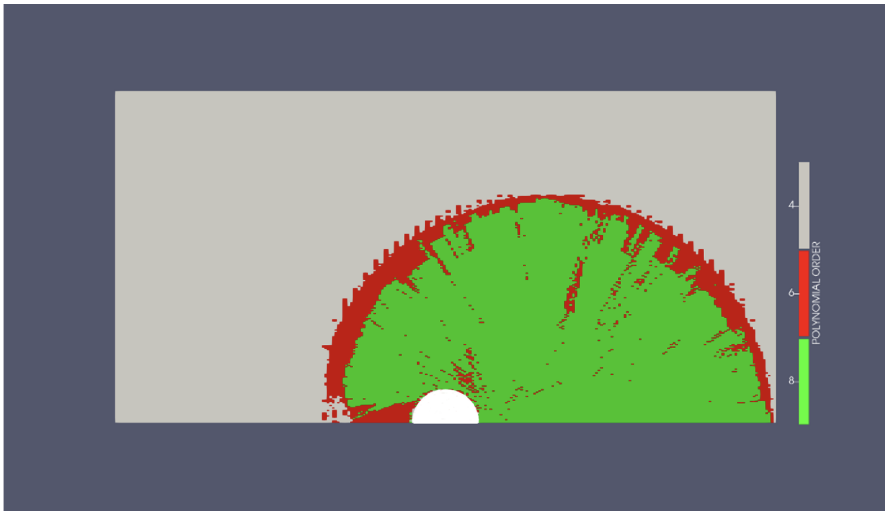


Figure 22: Polynomial order distribution in the adapted grid at $t = 3$ for the acoustic scattering off a cylinder case using the hp-adaptive immersed boundary implementation. Parameters as in Fig. 19.

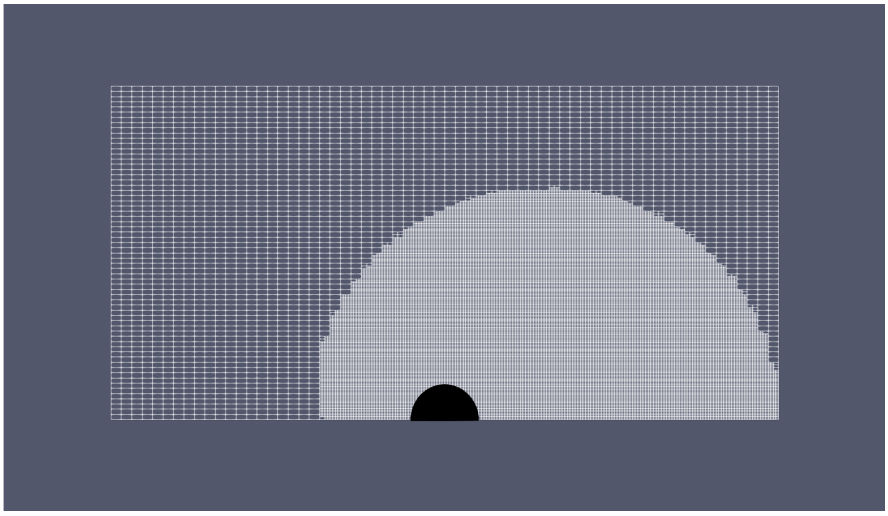


Figure 23: Adapted grid at $t = 3$ for the acoustic scattering off a cylinder using the hp-adaptive immersed boundary implementation. Parameters as in Fig. 19.

wave reflection off a NACA 0012 airfoil. The domain is a Cartesian domain $\Omega = [-2, 2]^2$ discretized with $N = 64 \times 64 = 4096$ elements with $p = 4$. We use hp-adaptivity allowing for two levels of h-refinement, p-refinement up to $p = 8$ with a error tolerance of $\epsilon = 10^{-6}$. We use a porosity value of $\phi = 10^{-8}$. The domain and the immersed boundary are presented in Fig. 24. We compare to a similarly run hp-adaptive GPU-based DGSEM case calculated by Tousignant [52] where an unstructured body-fitted grid was used. The initial grid is shown in Fig. 25.

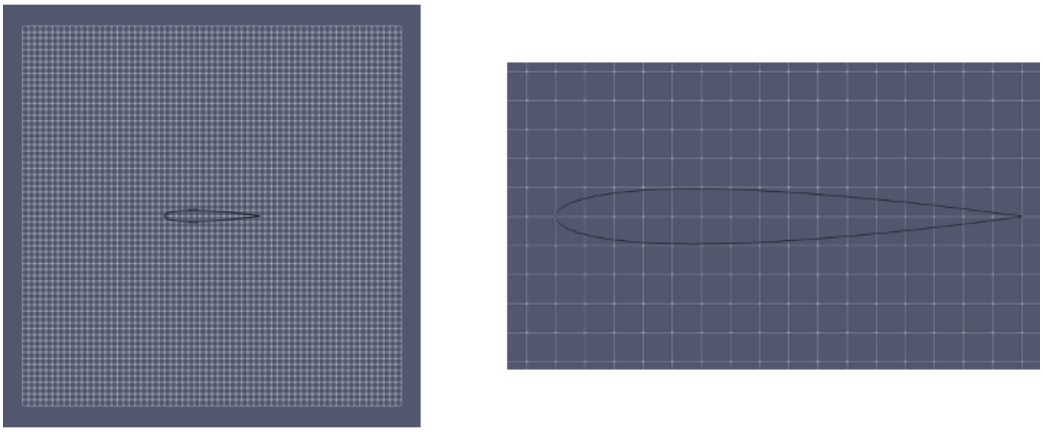
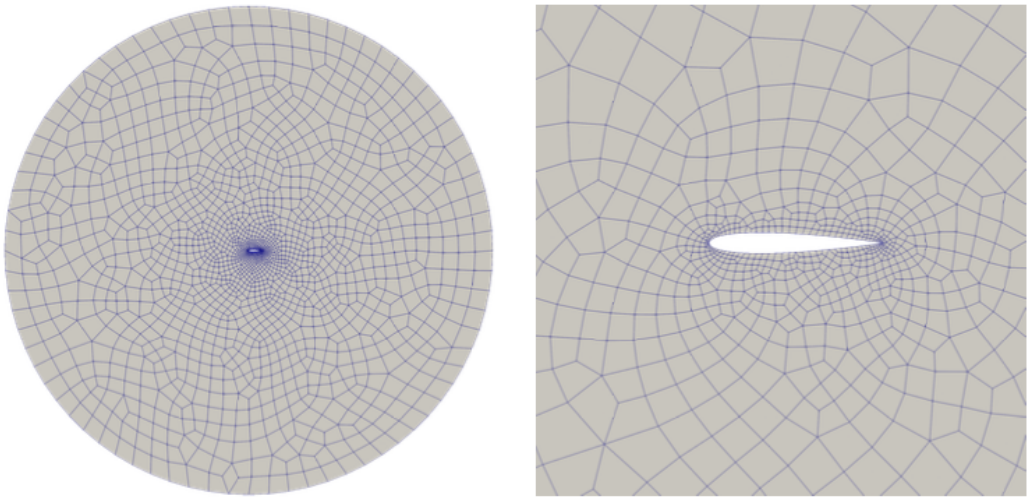


Figure 24: Grid for the immersed boundary NACA 0012 airfoil case. Overall grid (left), closeup near the immersed boundary (right).

The initial conditions are for a Gaussian plane wave:

$$\begin{aligned} P(x,y,0) &= e^{-\left(\frac{\left(\frac{\sqrt{2}}{2}(x+0.75)+\frac{\sqrt{2}}{2}(y+0.5)\right)^2}{\left(\frac{0.2}{2\sqrt{log(2)}}\right)^2}\right)} \\ u(x,y,0) &= \frac{\sqrt{2}}{2} e^{-\left(\frac{\left(\frac{\sqrt{2}}{2}(x+0.75)+\frac{\sqrt{2}}{2}(y+0.5)\right)^2}{\left(\frac{0.2}{2\sqrt{log(2)}}\right)^2}\right)} \\ v(x,y,0) &= \frac{\sqrt{2}}{2} e^{-\left(\frac{\left(\frac{\sqrt{2}}{2}(x+0.75)+\frac{\sqrt{2}}{2}(y+0.5)\right)^2}{\left(\frac{0.2}{2\sqrt{log(2)}}\right)^2}\right)} \end{aligned} \tag{40}$$

plotted in Fig. 26. The boundary conditions are given by the undisturbed solution for the Gaussian plane wave propagating through the domain. The simulation is run until a final time of $t = 3$ with a time step of $\Delta t = 2 \times 10^{-5}$. The simulation ends with a final total number of elements of $N = 65{,}137$ and a total number of degrees of freedom of 5,223,017. To visualize the case, we plot the pressure at $t = 2.1$ in Fig. 27.



(a) Circular domain (b) Airfoil

Figure 25: Unstructured grid for the NACA 0012 reference calculation by Tousignant. Overall grid (left), closeup near the airfoil (right). [52]

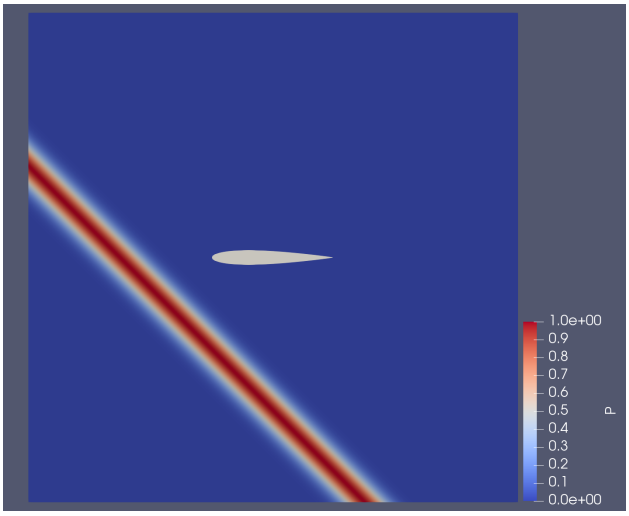


Figure 26: Initial pressure condition for the Gaussian plane wave reflecting off an immersed boundary NACA 0012 case. $N_{\text{initial}} = 64 \times 64 = 4096$ elements, $p = 4$.

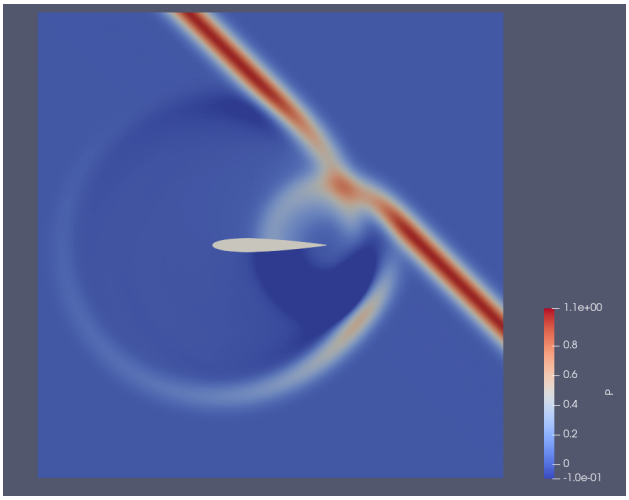


Figure 27: Pressure profile at $t = 2.1$ for the Gaussian plane wave reflecting off an immersed boundary NACA 0012 case. $N_{\text{initial}} = 64 \times 64 = 4096$ elements, $p \in [4, 6, 8]$, $\Delta t = 2 \times 10^{-5}$, $\phi = 10^{-8}$.

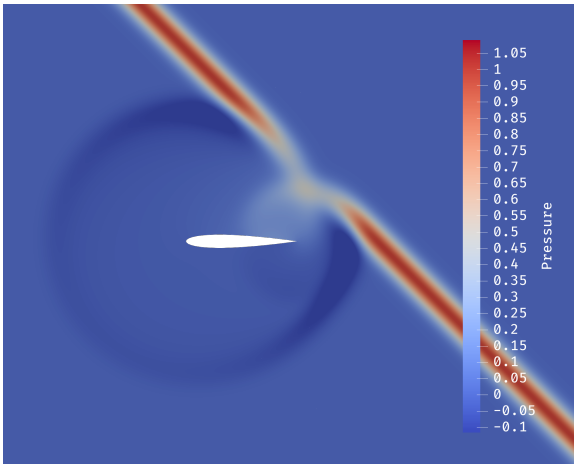


Figure 28: Pressure profile for the Gaussian plane wave reflecting off a NACA 0012 using a body-fitted unstructured grid. [52]

We observe that the immersed boundary implementation solution exhibits structures in the pressure field similar to those in Fig. 28 which uses a

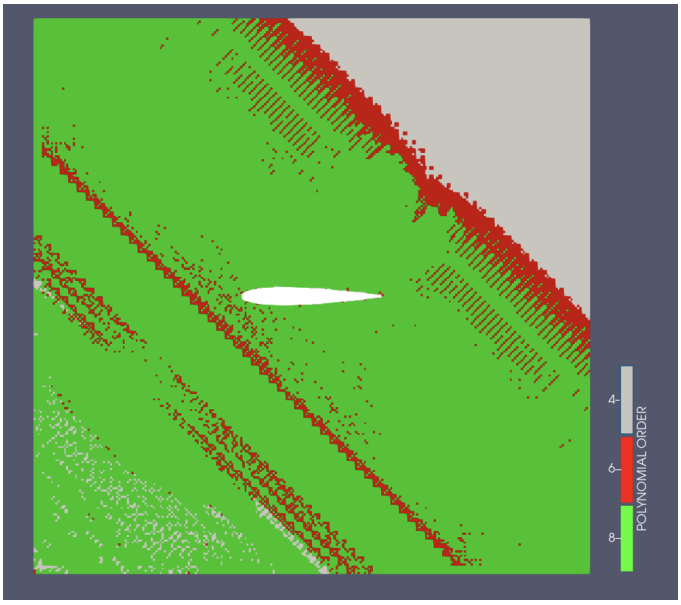


Figure 29: Polynomial order distribution in the adapted grid for the immersed boundary Gaussian plane wave reflecting off a NACA 0012 case. Parameters as in Fig. 27.

traditional wall boundary condition on the airfoil. (Note that the comparison is for illustration purposes only: Fig. 28 results are obtained with a much larger domain, a larger number of elements, three levels of h-refinement and polynomials ranging from 4 to 16. They are also plotted at a slightly different time.) The hp-adapted grid for the immersed boundary case is shown in Figures 29 (polynomial distribution) and 30 (h-refined elemental grid). The grid adaptation appears to follow the path of the Gaussian plane wave. With this test we show that the immersed boundary implementation is capable of modeling more complex geometries.

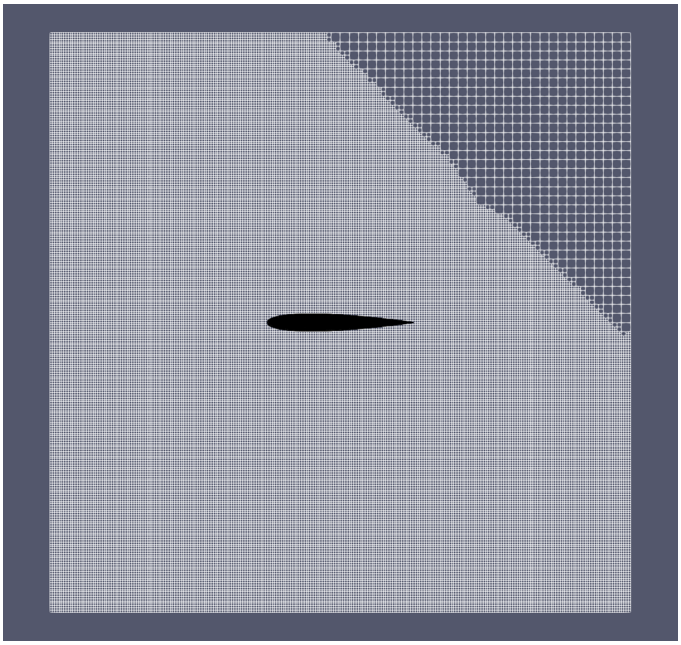


Figure 30: Adapted grid for the immersed boundary Gaussian plane wave reflecting off a NACA 0012 case. Parameters as in Fig. 27.

3.4. Irregular Surfaces

In this test case, we employ our immersed boundary implementation to model walls with smaller length scales. We simulate acoustic reflections on a sine wave wall and compare with a reference solution using a transfinite mapped grid. We also simulate reflections off a square wave wall as a contrast. The closeup views of the immersed boundary profiles for each wall are in Fig. 31, where the sine wave wall is the red curve and the square wave wall is the black curve. The sine-wave wall has a profile of $y = 0.05\,\sin(5\pi(x+1)) - 0.95$ and the square-wave wall has a square-wave profile with five periods equally spaced within $[-1, 1]$ with an amplitude of 0.1. The domain is the square $\Omega = [-1, 1]^2$ initially discretized with $N = 32 \times 32 = 1024$ elements and

polynomial order $p = 4$. The initial conditions for all the cases are visualized in Fig. 32 and are:

$$\begin{aligned} P(x,y,0) &= e^{-ln(2)(\frac{(x+1)^2+(y+0.6)^2}{0.075^2})} \\ u(x,y,0) &= 0 \\ v(x,y,0) &= 0. \end{aligned} \tag{41}$$

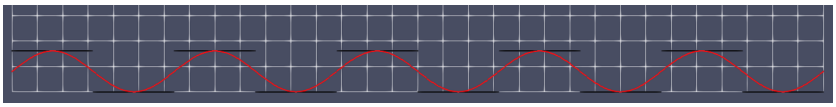


Figure 31: Closeup of the sine wave wall (red) and the square wave wall (black) immersed boundaries. Both immersed boundaries do not conform to the underlying Cartesian grid.

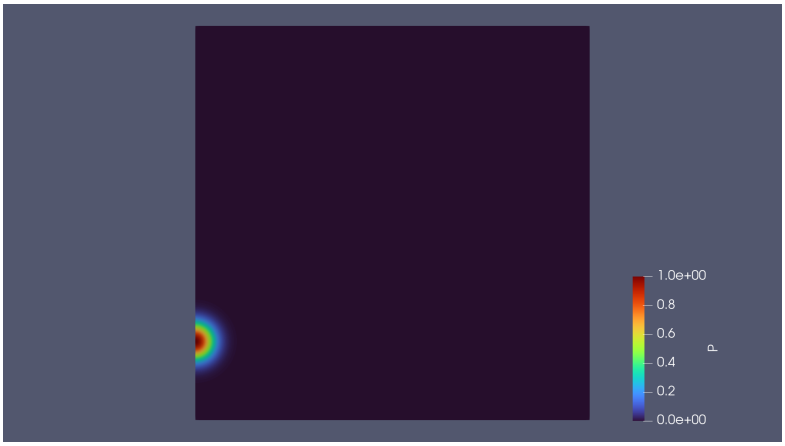


Figure 32: Initial pressure condition for both wavy wall immersed boundary cases.

Wall boundary conditions are imposed for the non-immersed boundaries. We use a time step of $\Delta t = 2 \times 10^{-5}$ and run until a final time of $t = 2$. For the transfinite reference simulation, we discretize with $N = 16 \times 16 = 256$ elements, use a polynomial order of $p = 21$ and a timestep of $\Delta t = 1 \times 10^{-5}$.

All the immersed boundary simulations are hp-adaptive, allowing for two levels of h-refinement and polynomial refinement up to $p = 8$, with a error tolerance of $\epsilon = 10^{-6}$, adapting every 10 time steps. The porosity value for both cases is $\phi = 10^{-8}$. The sine-wave wall case ends with a final total number of elements of $N = 16{,}105$ with a total number of degrees of freedom of 1,251,657. The square-wave wall case ends with a final total number of elements of $N = 16{,}111$ with a total number of degrees of freedom of 1,255,423.

We visualize the pressure fields for all the cases at an intermediate time of $t = 1$. First we present the reference case in Figure 33 and compare it to the immersed boundary case in Figure 34. The immersed boundary simulation shows strong visual agreement with the transfinite mapped simulation. The polynomial order distribution and adapted grid for the sine-wave wall are also presented in Figs. 35 (polynomial order distribution) and 36 (h-refined elemental grid). The hp-adaptation appears to follow the wave well. The pressure profile of the square wall case is plotted in Fig. 37. We observe that the shape of the wall does have a significant influence on the reflection pattern. The sine-wave wall, as expected, appears to produce a smoother reflection pattern. On the other hand, due to the sharp corners in the square-wave wall, this case appears to exhibit a more fragmented reflection pattern. The differences in the reflection patterns further validate that the immersed boundary implementation is successful in modeling objects with small length scales.

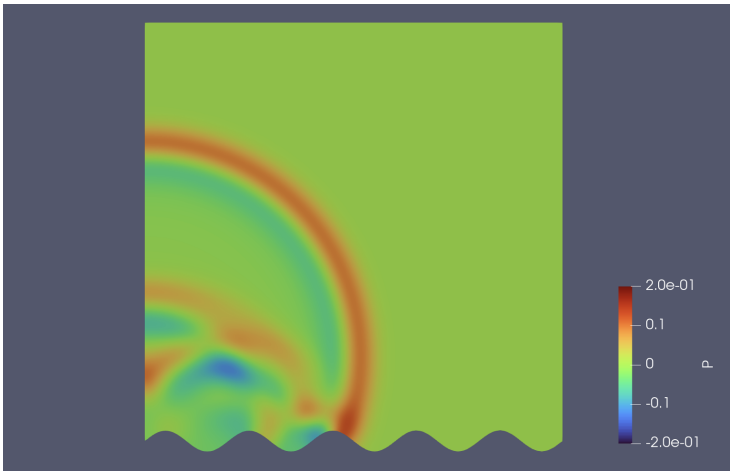


Figure 33: Pressure profile at $t = 1$ for the sine wave wall using a transfinite mapped grid. $N = 16 \times 16 = 256$ elements, $p = 21$, $\Delta t = 1 \times 10^{-5}$.

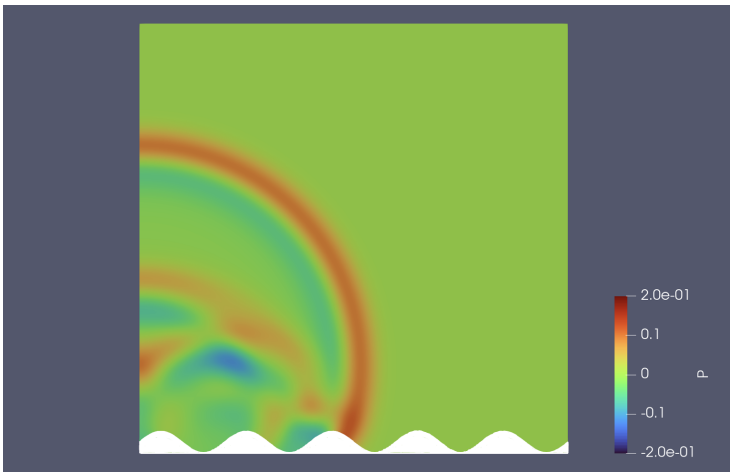


Figure 34: Pressure profile at $t = 1$ for the sine wave wall immersed boundary case. $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p \in [4, 6, 8]$, $\Delta t = 2 \times 10^{-5}$, $\phi = 10^{-8}$.

4. Conclusions

The immersed boundary method enables the modeling of complex geometries using simple Cartesian grids. This is a promising approach to drastically

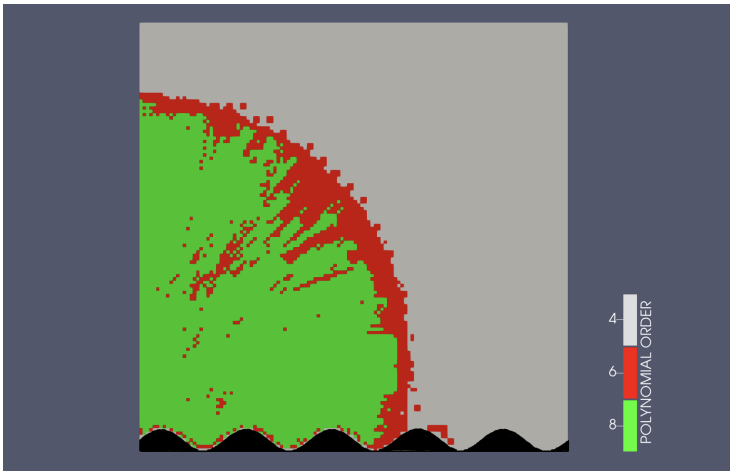


Figure 35: Polynomial order distribution at $t = 1$ for the sine wave wall immersed boundary case. Parameters as in Figure 34

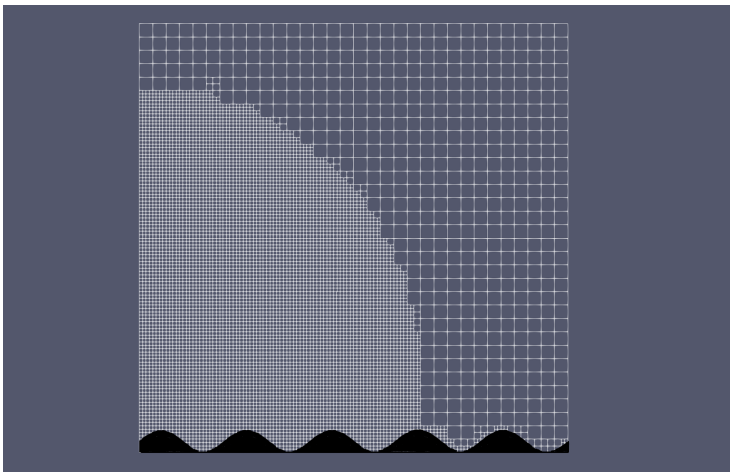


Figure 36: Adapted grid at $t = 1$ for the sine wave wall immersed boundary case. Parameters as in Figure 34

reduce user time spent in meshing complex geometries. However, it induces errors and can cause instabilities if not implemented with care. In this paper, the immersed boundary method was implemented in an hp-adaptive discon-

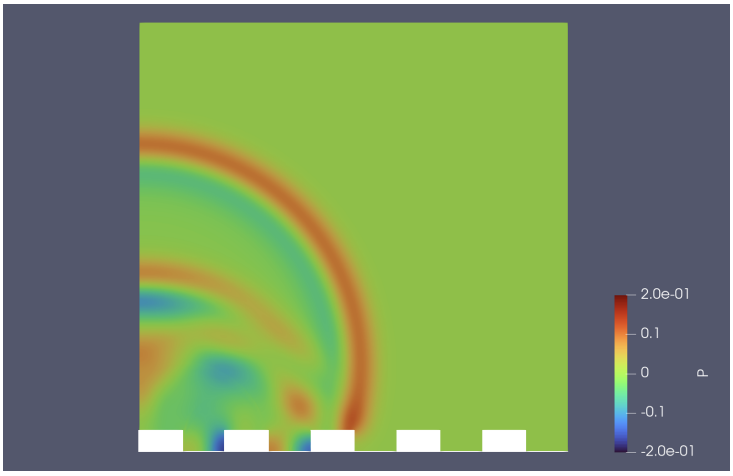


Figure 37: Pressure profile at $t = 1$ for the square wave wall immersed boundary case. $N_{\text{initial}} = 32 \times 32 = 1024$ elements, $p \in [4, 6, 8]$, $\Delta t = 2 \times 10^{-5}$, $\phi = 10^{-8}$.

tinuous Galerkin spectral element framework to solve the two-dimensional acoustic wave equation using the volume penalization method. We show that using hp-adaptivity driven by a modal decay indicator is successful in reducing oscillations, localizing the error to the vicinity of the immersed boundary, while also allowing for the use of smaller porosity values, ultimately improving accuracy. Furthermore, we also demonstrate that using an hp-adaptive approach is computationally more efficient compared to uniformly refining the grid to improve accuracy. We tested our implementation on a variety of cases, including complex geometries problems, showing that the hp-adaptive immersed boundary implementation is capable of modeling boundaries with different length scales using simple Cartesian grids.

Our work provides further flexibility to high-order methods for engineers, as carefully constructed, time-consuming, high-order body fitted meshes can be avoided. Furthermore, if higher accuracy is desired, body fitted meshes

can be constructed from prototype meshes devised by our hp-adaptive immersed boundary implementation. The method can determine where and which (h- or p-) resolution is needed in the computational domain to capture the physics with sufficient accuracy, providing a much better starting point for mesh generation.

We have tested the adaptive immersed boundary approach on the acoustic wave equation to provide a proof of concept, showing that dispersion and dissipation errors can be controlled with the method. We expect that adding viscosity or diffusion to our model equations in future work will pose even less problems as viscous solutions are less prone to oscillations. We will extend our implementation to the full incompressible Navier-Stokes equations to provide more efficient modeling capabilities for Direct Numerical Simulations of flows in complex geometries. Extensions to three dimensions are also planned. Future work will also include parallel scaling tests to ensure proper performance on HPC platforms and performance comparisons to using transfinite mapped or other body fitted grids.

Acknowledgments

The authors acknowledge support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Digital Research Alliance of Canada for computing resources.

In memoriam

This paper is dedicated to the memory of Prof. Arturo Hidalgo López (*July 03$^{\text{rd}}$ 1966 - †August 26$^{\text{th}}$ 2024) of the Universidad Politecnica de Madrid,

organizer of HONOM 2019 and active participant in many other editions of HONOM. Our thoughts and wishes go to his wife Lourdes and his sister María Jesús, whom he left behind.

References

[1] J. P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. J. Mavriplis, CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences, NASA CR-2014-218178, Langley Research Center (2014). doi:2060/20140003093.

[2] W. Chan, R. Gomez, S. Rogers, P. Buning, Best Practices in Overset Grid Generation, in: 32nd AIAA Fluid Dynamics Conference and Exhibit, AIAA Paper 2002-3191, American Institute of Aeronautics and Astronautics, St. Louis, Missouri, 2002. URL: https://arc.aiaa.org/doi/10.2514/6.2002-3191. doi:10.2514/6.2002-3191.

[3] D. J. Mavriplis, UNSTRUCTURED GRID TECHNIQUES, Annual Review of Fluid Mechanics 29 (1997) 473–514. URL: https://www.annualreviews.org/doi/10.1146/annurev.fluid.29.1.473. doi:10.1146/annurev.fluid.29.1.473.

[4] M. A. Park, A. Loseille, J. Krakos, T. R. Michal, J. J. Alonso, Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Towards CFD 2030, in: 46th AIAA Fluid Dynamics Conference, AIAA Paper 2016-3323, American Institute of Aeronautics and Astronautics, Washington, D.C., 2016. URL: https://arc.aiaa.org/doi/10.2514/6.2016-3323. doi:10.2514/6.2016-3323.

[5] W. M. Chan, Overset grid technology development at NASA Ames Research Center, Computers & Fluids 38 (2009) 496–503. URL: https:

//linkinghub.elsevier.com/retrieve/pii/S0045793008001953. doi:10.1016/j.compfluid.2008.06.009.

[6] J. K. Hodge, S. A. Leone, R. L. McCarty, Noniterative parabolic grid generation for parabolized equations, AIAA Journal 25 (1987) 542–549. URL: https://arc.aiaa.org/doi/10.2514/3.9661. doi:10.2514/3.9661.

[7] M. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, Journal of Computational Physics 82 (1989) 64–84. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999189900351. doi:10.1016/0021-9991(89)90035-1.

[8] P. R. Eiseman, Adaptive grid generation, Computer Methods in Applied Mechanics and Engineering 64 (1987) 321–376. URL: https://linkinghub.elsevier.com/retrieve/pii/0045782587900466. doi:10.1016/0045-7825(87)90046-6.

[9] D. Gottlieb, S. A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS-NSF Regional Conference Series in Applied Mathematics, Soc. for Industrial and Applied Mathematics, Philadelphia, Pa, 2009.

[10] A. T. Patera, A spectral element method for fluid dynamics: Laminar flow in a channel expansion, Journal of Computational Physics 54 (1984) 468–488. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999184901281. doi:10.1016/0021-9991(84)90128-1.

[11] M. Min, Y.-H. Lan, P. Fischer, T. Rathnayake, J. Holmen, Nek5000/RS performance on advanced GPU architectures, Frontiers in High Performance Computing 2 (2025) 1303358. URL: https://www.frontiersin.org/articles/10.3389/fhpcp.2024.1303358/full. doi:10.3389/fhpcp.2024.1303358.

[12] K. Z. Korczak, A. T. Patera, An isoparametric spectral element method for solution of the Navier-Stokes equations in complex geometry, Journal of Computational Physics 62 (1986) 361–382. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999186901348. doi:10.1016/0021-9991(86)90134-8.

[13] N. Offermans, D. Massaro, A. Peplinski, P. Schlatter, Error-driven adaptive mesh refinement for unsteady turbulent flows in spectral-element simulations, Computers & Fluids 251 (2023) 105736. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793022003280. doi:10.1016/j.compfluid.2022.105736.

[14] F. Basile, J.-B. Chapelier, M. De La Llave Plata, R. Laraufie, P. Frey, Unstructured h- and hp-adaptive strategies for discontinuous Galerkin methods based on a posteriori error estimation for compressible flows, Computers Fluids 233 (2022) 105245. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793021003509. doi:10.1016/j.compfluid.2021.105245.

[15] C. Mavriplis, Adaptive mesh strategies for the spectral element method, Computer Methods in Applied Mechanics and Engineering 116

(1994) 77–86. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045782594800103. doi:10.1016/S0045-7825(94)80010-3.

[16] H. Feng, C. Mavriplis, Adaptive Spectral Element Simulations of Thin Premixed Flame Sheet Deformations, Journal of Scientific Computing 17 (2002) 385–395. doi:10.1023/A:1015137722700.

[17] C. Burstedde, L. C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (2011) 1103–1133. doi:10.1137/100791634.

[18] D. Moxey, C. D. Cantwell, G. Mengaldo, D. Serson, D. Ekelschot, J. Peiró, S. J. Sherwin, R. M. Kirby, Towards p-adaptive spectral/hp element methods for modelling industrial flows, in: M. Bittencourt, N. A Doumont, J. S Hesthaven (Eds.), Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016, volume 119 of *Lecture Notes in Computational Science and Engineering*, Springer Nature, 2017, pp. 63–79.

[19] S. He, Dynamic Load Balancing for a hp-adaptive Discontinuous Galerkin Wave Equation Solver via Spacing-Filling Curve and Advanced Data Structure, Master of Applied Science thesis, University of Ottawa, Ottawa, Canada, 2021.

[20] Y. Maday, C. Mavriplis, A. Patera, Nonconforming mortar element methods: Application to spectral discretizations, in: T.F. Chan, R. Glowinski, J. Periaux, O.B. Widlund (Eds.), Domain Decomposition Methods, SIAM Philadelphia, 1989, p. 392–418.

[21] D. A. Kopriva, J. H. Kolias, A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows, Journal of Computational Physics 125 (1996) 244–261. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999196900911. doi:10.1006/jcph.1996.0091.

[22] N. Chalmers, G. Agbaglah, M. Chrust, C. Mavriplis, A parallel hp-adaptive high order discontinuous Galerkin method for the incompressible Navier-Stokes equations, Journal of Computational Physics: X 2 (2019) 100023. URL: https://linkinghub.elsevier.com/retrieve/pii/S2590055219300393. doi:10.1016/j.jcpx.2019.100023.

[23] P. Mossier, P. Oestringer, J. Keim, C. Mavriplis, A. D. Beck, C.-D. Munz, Tackling Compressible Turbulent Multi-Component Flows with Dynamic hp-Adaptation (2025). URL: http://arxiv.org/abs/2502.19342. doi:10.48550/arXiv.2502.19342, arXiv:2502.19342.

[24] A. D. Beck, T. Bolemann, D. Flad, H. Frank, G. J. Gassner, F. Hindenlang, C. Munz, High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations, International Journal for Numerical Methods in Fluids 76 (2014) 522–548. URL: https://onlinelibrary.wiley.com/doi/10.1002/fld.3943. doi:10.1002/fld.3943.

[25] F. Renac, Entropy stable, robust and high-order DGSEM for the compressible multicomponent Euler equations, Journal of Computational Physics 445 (2021) 110584. URL: https:

//linkinghub.elsevier.com/retrieve/pii/S0021999121004794. doi:10.1016/j.jcp.2021.110584.

[26] J. Manzanero, C. Redondo, M. Chávez-Módena, G. Rubio, E. Valero, S. Gómez-Álvarez, Rivero-Jiménez, High-order discontinuous Galerkin approximation for a three-phase incompressible Navier–Stokes/Cahn–Hilliard model, Computers & Fluids 244 (2022) 105545. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793022001700. doi:10.1016/j.compfluid.2022.105545.

[27] G. J. Gassner, A. R. Winters, D. A. Kopriva, A well balanced and entropy conservative discontinuous Galerkin spectral element method for the shallow water equations, Applied Mathematics and Computation 272 (2016) 291–308. URL: https://linkinghub.elsevier.com/retrieve/pii/S0096300315009261. doi:10.1016/j.amc.2015.07.014.

[28] N. Wintermeyer, A. R. Winters, G. J. Gassner, D. A. Kopriva, An entropy stable nodal discontinuous Galerkin method for the two dimensional shallow water equations on unstructured curvilinear meshes with discontinuous bathymetry, Journal of Computational Physics 340 (2017) 200–242. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999117302310. doi:10.1016/j.jcp.2017.03.036.

[29] D. A. Kopriva, S. L. Woodruff, M. Y. Hussaini, Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method, International Journal for Numerical Methods in Engi-

neering 53 (2002) 105–122. URL: https://onlinelibrary.wiley.com/doi/10.1002/nme.394. doi:10.1002/nme.394.

[30] C. S. Peskin, Flow patterns around heart valves: A numerical method, Journal of Computational Physics 10 (1972) 252–271. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999172900654. doi:10.1016/0021-9991(72)90065-4.

[31] E. Arquis, J. Caltagirone, Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide – milieu poreux: application à la convection naturelle, Comptes Rendus de l'Académie des Sciences. Série 2. b. Mecanique 299 (1984) 1–4.

[32] D. Goldstein, R. Handler, L. Sirovich, Modeling a No-Slip Flow Boundary with an External Force Field, Journal of Computational Physics 105 (1993) 354–366. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999183710818. doi:10.1006/jcph.1993.1081.

[33] E. Saiki, S. Biringen, Numerical Simulation of a Cylinder in Uniform Flow: Application of a Virtual Boundary Method, Journal of Computational Physics 123 (1996) 450–465. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999196900364. doi:10.1006/jcph.1996.0036.

[34] K. Taira, T. Colonius, The immersed boundary method: A projection approach, Journal of Computational Physics 225 (2007) 2118–2137. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999107001234. doi:10.1016/j.jcp.2007.03.005.

[35] A. J. Chorin, Numerical solution of the Navier-Stokes equations, Mathematics of Computation 22 (1968) 745–762. URL: https://www.ams.org/mcom/1968-22-104/S0025-5718-1968-0242392-2/. doi:10.1090/S0025-5718-1968-0242392-2.

[36] J. Perot, An Analysis of the Fractional Step Method, Journal of Computational Physics 108 (1993) 51–58. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999183711629. doi:10.1006/jcph.1993.1162.

[37] R. Ghias, R. Mittal, H. Dong, A sharp interface immersed boundary method for compressible viscous flows, Journal of Computational Physics 225 (2007) 528–553. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999106006048. doi:10.1016/j.jcp.2006.12.007.

[38] N. Anand, N. Ebrahimi Pour, H. Klimach, S. Roller, Utilization of the Brinkman Penalization to Represent Geometries in a High-Order Discontinuous Galerkin Scheme on Octree Meshes, Symmetry 11 (2019) 1126. URL: https://www.mdpi.com/2073-8994/11/9/1126. doi:10.3390/sym11091126.

[39] E. L. Sharaborin, O. A. Rogozin, A. R. Kasimov, The Coupled Volume of Fluid and Brinkman Penalization Methods for Simulation of Incompressible Multiphase Flows, Fluids 6 (2021) 334. URL: https://www.mdpi.com/2311-5521/6/9/334. doi:10.3390/fluids6090334.

[40] Y. Bae, Y. J. Moon, On the use of Brinkman penalization method for

computation of acoustic scattering from complex boundaries, Computers & Fluids 55 (2012) 48–56. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793011003215. doi:10.1016/j.compfluid.2011.10.015.

[41] A. Piquet, O. Roussel, A. Hadjadj, A comparative study of Brinkman penalization and direct-forcing immersed boundary methods for compressible viscous flows, Computers & Fluids 136 (2016) 272–284. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793016301888. doi:10.1016/j.compfluid.2016.06.001.

[42] Q. Liu, O. V. Vasilyev, A Brinkman penalization method for compressible flows in complex geometries, Journal of Computational Physics 227 (2007) 946–966. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999107003403. doi:10.1016/j.jcp.2007.07.037.

[43] N. K.-R. Kevlahan, T. Dubos, M. Aechtner, Adaptive wavelet simulation of global ocean dynamics using a new Brinkman volume penalization, Geoscientific Model Development 8 (2015) 3891–3909. URL: https://gmd.copernicus.org/articles/8/3891/2015/. doi:10.5194/gmd-8-3891-2015.

[44] J. Kou, S. Joshi, A. Hurtado-de Mendoza, K. Puri, C. Hirsch, E. Ferrer, Immersed boundary method for high-order flux reconstruction based on volume penalization, Journal of Computational Physics 448 (2022) 110721. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999121006161. doi:10.1016/j.jcp.2021.110721.

[45] N. Ebrahimi Pour, N. Anand, H. Klimach, S. Roller, Compressible flow simulation with moving geometries using the Brinkman penalization in high-order Discontinuous Galerkin, Advanced Modeling and Simulation in Engineering Sciences 8 (2021) 10. URL: https://amses-journal.springeropen.com/articles/10.1186/s40323-021-00195-4. doi:10.1186/s40323-021-00195-4.

[46] V. J. Llorente, J. Kou, E. Valero, E. Ferrer, A modified equation analysis for immersed boundary methods based on volume penalization: Applications to linear advection–diffusion equations and high-order discontinuous Galerkin schemes, Computers & Fluids 257 (2023) 105869. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793023000944. doi:10.1016/j.compfluid.2023.105869.

[47] D. A. Kopriva, Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers, Scientific Computation, Springer Netherlands, Dordrecht, 2009. URL: http://link.springer.com/10.1007/978-90-481-2261-5. doi:10.1007/978-90-481-2261-5.

[48] D. A. Kopriva, A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows. II. A Semi-Structured Method, Journal of Computational Physics 128 (1996) 475–488. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999196902259. doi:10.1006/jcph.1996.0225.

[49] A. Paccou, G. Chiavassa, J. Liandrat, K. Schneider, A penalization method applied to the wave equation, Comptes Rendus. Mécanique 333

(2004) 79–85. URL: https://comptes-rendus.academie-sciences.fr/mecanique/articles/10.1016/j.crme.2004.09.019/. doi:10.1016/j.crme.2004.09.019.

[50] N. Beisiegel, C. E. Castro, J. Behrens, Metrics for performance quantification of adaptive mesh refinement, Journal of Scientific Computing 87 (2021) 36. URL: http://link.springer.com/10.1007/s10915-021-01423-0. doi:10.1007/s10915-021-01423-0.

[51] A. Nayak, Improving Efficiency in the Discontinuous Galerkin Spectral Element Method for Complex Geometry Physics, Master of Applied Science thesis, University of Ottawa, Ottawa, Canada, 2025.

[52] G. Tousignant, A Graphics Processing Unit Based Discontinuous Galerkin Wave Equation Solver with hp-Adaptivity and Load Balancing, Master of Applied Science thesis, University of Ottawa, Ottawa, Canada, 2022.

Chapter 4

Half-Closed Discontinuous Galerkin Methods for Incompressible Flow Problems

This chapter focuses on the underlying discretization to improve efficiency by using so-called "half-closed" nodes for more efficient operator construction and better sparsity patterns. Complex geometries are modeled using unstructured grids. The following is an edited and reprinted conference paper published in the proceedings of AIAA SciTech 2025.

Contributions

The author wrote this paper in collaboration with the other co-authors. The author implemented the half-closed discontinuous Galerkin method and performed all of the simulations.

Half-Closed Discontinuous Galerkin Methods For Incompressible Flow Problems

Amit Nayak*
University of Ottawa, Ottawa, ON K1N 6N5, Canada

Yulong Pan†
University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A.

Per-Olof Persson‡
University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A.

We utilize the new approach of using "half-closed" nodes for nodal Discontinuous Galerkin (DG) methods for incompressible flow problems. Closed nodes in DG have element nodes on each boundary interface. However, half-closed nodes weakly satisfy this constraint by placing element nodes on a subset of boundaries in each element. This enables more efficient construction of DG operators due to the increased flexibility in picking nodes, while preserving the underlying properties of DG schemes and obtaining good sparsity patterns. The method is applied here to a mix of problems ranging from low to moderate Reynolds numbers.

I. Nomenclature

C_D	=	Boundary stabilization parameter
D	=	Divergence
d	=	Dimension
$F(u)$	=	Flux
$\hat{F}(u)$	=	Numerical flux
f	=	Forcing function
G	=	Gradient
g_D, g_N	=	Dirichlet and Neumann boundary value
K_n	=	Element
L	=	Laplacian
M	=	Mass matrix
P	=	Pressure
p	=	Polynomial order
q	=	Geometry order
Re	=	Reynolds number
S_n^m	=	Switch function
t	=	Time
u, v	=	Velocity
α	=	Angle of attack
ϕ	=	Basis function
Ω	=	Domain

*Affiliate, Mathematics Group, Lawrence Berkeley National Laboratory. Master's Student, Department of Mechanical Engineering, University of Ottawa. Email: anaya085@uottawa.ca. AIAA Student Member

†Graduate Student Researcher, Mathematics Group, Lawrence Berkeley National Laboratory. Ph.D Student, Department of Mathematics, University of California, Berkeley. Email: yllpan@berkeley.edu. AIAA Student Member

‡Faculty Scientist, Mathematics Group, Lawrence Berkeley National Laboratory. Professor, Department of Mathematics, University of California, Berkeley. E-mail: persson@berkeley.edu. AIAA Senior Member.

II. Introduction

High-order methods for computational fluid dynamics (CFD) have gained popularity in recent years [1]. One such example is the discontinuous Galerkin (DG) method. Some advantages of DG include geometric flexibility to support unstructured grids, arbitrarily high order accuracy, capability of local hp-refinement and that it is highly parallelizable [2] [3]. These advantages make it promising for use in simulations of turbulent flows. However, there is still room for improvement before DG methods are competitive for use in aerospace applications.

One of the largest criticisms of DG is the computational cost. Much research has been conducted to improve the efficiency resulting in methods such as the Hybridized Discontinuous Galerkin method [4], the Discontinuous Galerkin Spectral Element Method (DGSEM) [5], and the Flux Reconstruction Method [6] to name a few. In this work, we aim to address this issue by using the Half-closed discontinuous Galerkin (HCDG) method introduced in [7]. In HCDG, the DG formulation is not modified, rather node placement enabling the use of "half-closed" nodes is used to improve efficiency.

III. Overview of the Discontinuous Galerkin Method

Since HCDG does not modify the DG formulation, an overview of the DG method will now be presented. The domain Ω is triangulated into elements $T_h = \{K_n : \bigcup K_n = \Omega\}$. In this work, we only consider quadrilateral elements. Then, an element-wise function space V_h is imposed as

$$V_h = \{v_h|_K \in V(K_n)\}. \tag{1}$$

For $V(K_n)$, we utilize d-dimensional outer products of one-dimensional polynomials of degree at most p. Functions have local support per element and are discontinuous along element boundaries $\partial\Omega$. Thus, we utilize numerical fluxes $\hat{F}$ on the boundaries. We utilize the discontinuous Galerkin discretization for a first-order system of conservation laws

$$\frac{\partial u}{\partial t} + \nabla \cdot \boldsymbol{F}(u) = 0, \tag{2}$$

where $\boldsymbol{F}(u)$ is the flux function. We seek an approximation of $u \in V_h$ by multiplying Eq. 2 by a test function $v \in V_h$ and integrating by parts,

$$\frac{\partial}{\partial t}\int_{K_n} uv dx = \int_{K_n} \nabla v \cdot \boldsymbol{F}(u) dx - \int_{\partial K_n} v\hat{\boldsymbol{F}}(u) \cdot \boldsymbol{n} ds, \quad \forall v \in V_h, \tag{3}$$

where $\boldsymbol{n}$ is the boundary normal. Popular choices of $\hat{\boldsymbol{F}}$ include Godunov, Lax-Fredrichs or HLLC (Harten-Lax-van Leer-Contact) fluxes, and in this work we utilize the Godunov flux, which for the incompressible Navier-Stokes equations simplifies to be the upwind flux. We utilize the nodal discontinuous Galerkin method, where the solution u and the numerical fluxes are represented as a linear combination of the basis functions ϕ_j

$$u = \sum_j \phi_j u_j, \quad \phi_j \in V_h. \tag{4}$$

The test function ϕ_i is chosen as an arbitrary basis function $\phi_i \in V_h$. This results in

$$\sum_n \int_{K_n} \phi_i \phi_j dx \cdot \frac{\partial}{\partial t} u_j = \sum_d \sum_n \int_{K_n} \frac{\partial \phi_i}{\partial x_d} \phi_j dx \cdot F_d(u_j) - \int_{\partial K_n} \phi_i \hat{F}_d(u^{\text{int}}, u^{\text{ext}}) \cdot n_d ds, \tag{5}$$

where u^{int} and u^{ext} are the solution in the interior and exterior of the element K_n, respectively, and F_d has been introduced to denote the d-th spatial component of the flux $\boldsymbol{F}(u)$. The operator on the left side of Eq. 5 is known as the mass matrix with entries defined as

$$M_{ij} = \sum_n \int_{K_n} \phi_i \phi_j dx, \tag{6}$$

and the operator on the right hand side of the equation is the discrete divergence which acts on the flux $\boldsymbol{F}(u)$ as

$$D^d F_d = \sum_n \int_{K_n} \frac{\partial \phi_i}{\partial x_d} \phi_j F_d(u_j) dx - \int_{\partial K_n} \phi_i \hat{F}_d(u^{\text{int}}, u^{\text{ext}}) \cdot n_d ds. \tag{7}$$

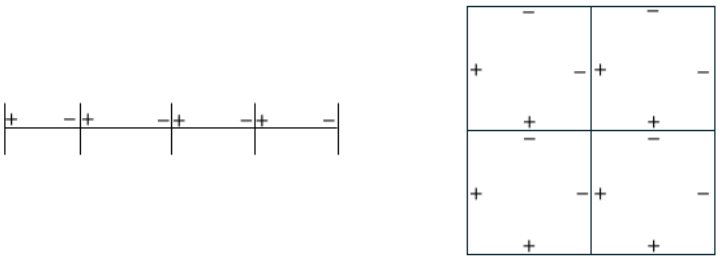


Fig. 1 Examples of switch functions in 1D and in 2D for quadrilateral elements.

Overall we obtain the semi-discrete system

$$M\dot{u} = \sum_d D^d \boldsymbol{F}_d. \tag{8}$$

For second order partial differential equations such as Poisson's equation

$$-\nabla^2 u = f, \tag{9}$$

we split into first order equations and utilize the local discontinuous Galerkin (LDG) method [8]. First, $\boldsymbol{q}$ is defined as ∇u. Substituting this into Eq. 9 we obtain

$$-\nabla \cdot \boldsymbol{q} = f, \tag{10}$$

$$\boldsymbol{q} = \nabla u. \tag{11}$$

Similarly for first order equations, we multiply by test functions v and τ and integrate by parts.

$$\int_{K_n} \boldsymbol{q} \cdot \nabla v dx - \int_{\partial K_n} v\hat{\boldsymbol{q}} \cdot \boldsymbol{n} ds = \int_{K_n} f v dx \quad \forall v \in V_h, \tag{12}$$

$$\int_{K_n} (\boldsymbol{q} + u\nabla) \cdot \boldsymbol{\tau} dx - \int_{\partial K_n} \hat{u}\boldsymbol{\tau} \cdot \boldsymbol{n} ds = 0 \quad \forall \boldsymbol{\tau} \in V_h^d. \tag{13}$$

On each element boundary, a switch function is defined to specify the fluxes $\hat{\boldsymbol{q}}$ and $\hat{u}$. An example of a valid switch function is shown in Fig. 1. For the boundary ∂K_n between element K_n and neighbor element K_m, the switch function S_n^m results in a value of -1 or +1 with the constraint that $S_n^m + S_m^n = 0$. We use the upwind-downwind flux also known as the minimal dissipation LDG flux based on the switch function such that

$$\hat{\boldsymbol{q}} = \begin{cases} \boldsymbol{q}|_{K_n} & \text{if } S_n^m < 0, \\ \boldsymbol{q}|_{K_m} & \text{if } S_m^n > 0, \end{cases} \tag{14}$$

and

$$\hat{u} = \begin{cases} u|_{K_n} & if\ S_n^m > 0, \\ u|_{K_m} & if\ S_m^n < 0. \end{cases} \tag{15}$$

Weakly imposed boundary conditions are used, the fluxes on the domain boundary are

$$\hat{\boldsymbol{q}} = \begin{cases} \boldsymbol{q} - C_D(u - g_D)\boldsymbol{n} & \text{on } \Gamma_D, \\ g_N \boldsymbol{n} & \text{on } \Gamma_N, \end{cases} \tag{16}$$

and

$$\hat{u} = \begin{cases} g_D & \text{on } \Gamma_D, \\ u & \text{on } \Gamma_N. \end{cases} \tag{17}$$

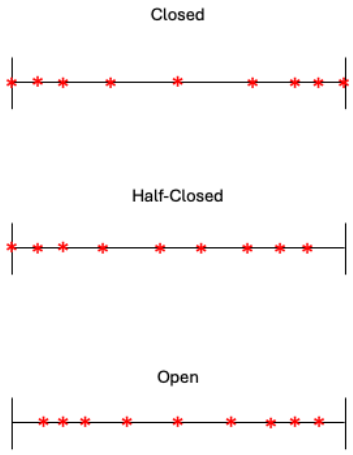


Fig. 2 Visualization of node classifications.

Here, $C_D > 0$ is a user defined stabilization constant. As with the conservation law, we represent both u and $\boldsymbol{q}$ as a linear combination of the basis functions and use the same basis functions for the test functions. Writing the system of equations in operator form (for arbitrary dimension),

$$\begin{bmatrix} M & G^d \\ -D^d & \end{bmatrix} \begin{bmatrix} q^d \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ f \end{bmatrix}. \tag{18}$$

Here, G is the gradient operator related to the divergence operator via the adjoint property $G = -D^T$. The solution u can be solved for without needing to explicitly solve for $\boldsymbol{q}$ by taking the Schur complement of Eq. 18 and by solving $-Lu = f$. Finally, the Laplacian operator L is given by

$$L = \sum_d D^d M^{-1} G^d. \tag{19}$$

IV. Half-Closed Discontinuous Galerkin Method

The Half-Closed Discontinuous Galerkin (HCDG) method provides the advantages of increased flexibility in operator construction and better sparsity patterns while retaining the underlying DG properties. Each advantage will now be discussed.

A. Nodes

The name "half-closed" comes from the choice of nodes. For DG methods, commonly used nodes include Gauss-Lobatto nodes and Gauss-Legendre nodes. Gauss-Lobatto nodes are an example of closed nodes since they have nodes on the element boundaries whereas Gauss-Legendre nodes are an example of open nodes since they do not have any nodes on the element boundaries. The visual distinction between closed, half-closed and open nodes is shown in Fig. 2.

Closed nodes have the restriction that there must be nodes from neighboring elements K_n and K_m on the intra-element boundary. In HCDG, at the intra-element boundary between an element K_n and its neighbor K_m, element K_n must place nodes on the boundary such that the switch function satisfies $S_n^m + S_m^n = 0$. This relaxed constraint ensures that at each intra-element boundary there exist nodes from at least one element, allowing for more efficient operator construction by increasing flexibility for the choice of node placement. An example of node placement in HCDG is shown in Fig. 3 where the node on the boundary is set according to the switch function. In this work, we use Gauss-Radau nodes as our half-closed nodes.

Similarly to the discontinuous Galerkin spectral element method (DGSEM), in the half-closed discontinuous Galerkin method we use the solution nodes as the quadrature nodes. One advantage of using Gauss-Radau nodes

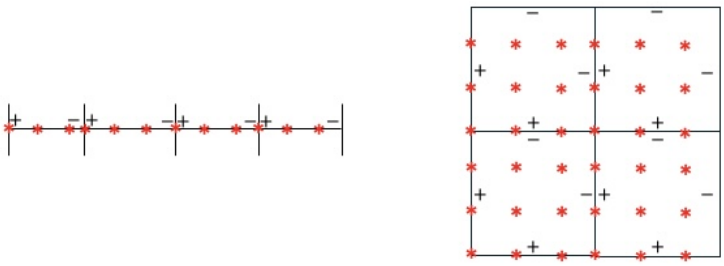


Fig. 3 Half-closed node placement according to switch function.

for quadrature is that we obtain an extra degree of accuracy over the more commonly used Gauss-Lobatto nodes. Gauss-Radau quadrature can exactly integrate polynomials of degree $2n-2$ while Gauss-Lobatto quadrature can only exactly integrate polynomials of degree $2n-3$.

B. Sparsity Patterns

The sparsity pattern of the operators previously described in Section III will now be described briefly. The full proof of the sparsity of these operators can be found in [7], however for conciseness only the final results will be presented here.

1. Mass Matrix

The mass matrix presented in Eq. 6 is block diagonal, regardless of node choice since it solely consists of volume terms. However, for certain choices of nodes the mass matrix can be diagonal if

$$M_{ij} = \sum_n \int_{K_n} \phi_i \phi_j dx = W_i \delta_{ij} \tag{20}$$

where $W_i = \sum_n \int_{K_n} \phi_i^2 dx$. This occurs when the basis functions are pairwise orthogonal. Gauss-Radau nodes are half-closed nodes that are known to possess this property. At the time of writing, there are no known closed nodes that possess this property. Thus, there is an advantage to using Gauss-Radau nodes over closed nodes as it produces a diagonal mass matrix, which is trivial to invert.

2. Divergence Operator

The divergence operator (and the gradient operator) consist of contributions from the volume integral and the boundary integral terms as seen in Eq. 7. For the case of open nodes, an interpolation of all nodal values from an element are required to calculate the boundary fluxes. This implies a communication pattern where all the nodes of neighboring elements K_n and K_m communicate with each other ultimately resulting in a more dense operator. For the case of closed nodes, nodes are placed on each intraelement boundary by each neighboring element. On the boundary, the basis functions for the nodes not on the boundary are zero. This implies a communication pattern where only the boundary nodes from neighboring elements K_n and K_m communicate with each other. This results in a more sparse divergence operator when compared to using open nodes. In the case of half closed nodes, at each intraelement boundary there exist nodes from one of the neighboring elements. This results in a mix of the procedures from both open and closed nodes. An interpolation of all nodal values is required for the element that does not have nodes on the boundary, while the element that does have nodes on the boundary relies only on the boundary nodal values. This implies a communication pattern involving less nodes when compared to using open nodes, albeit involving more nodes then when using closed nodes. Thus, using half-closed nodes has the advantage over open nodes of producing a sparse divergence operator. A visual example from [7] of the different types of sparsities is presented in Fig. 4.

3. Laplacian Operator

The LDG Laplacian in Eq. 19 is composed using the divergence operator, the inverse of the mass matrix and the gradient operator ($G = -D^T$). Through considering the sparsity patterns of these operators, it was shown in [7] that the

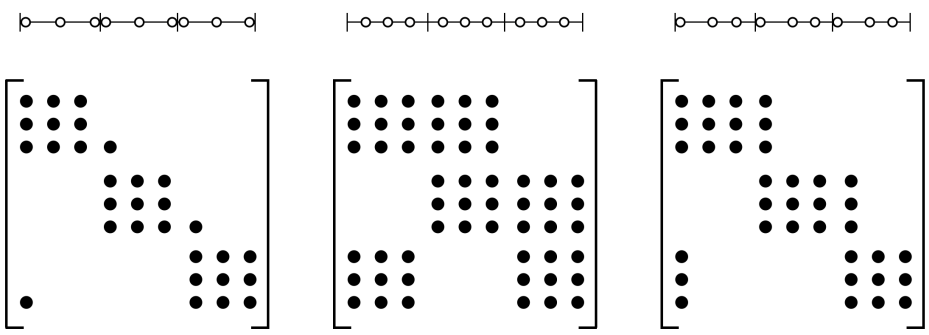


Fig. 4 Sparsity pattern comparison of the one dimensional divergence operator with three $p = 2$ elements, upwinding flux, and periodic boundary conditions [7]. Left to right shows: Closed nodes, open nodes, and half-closed nodes.

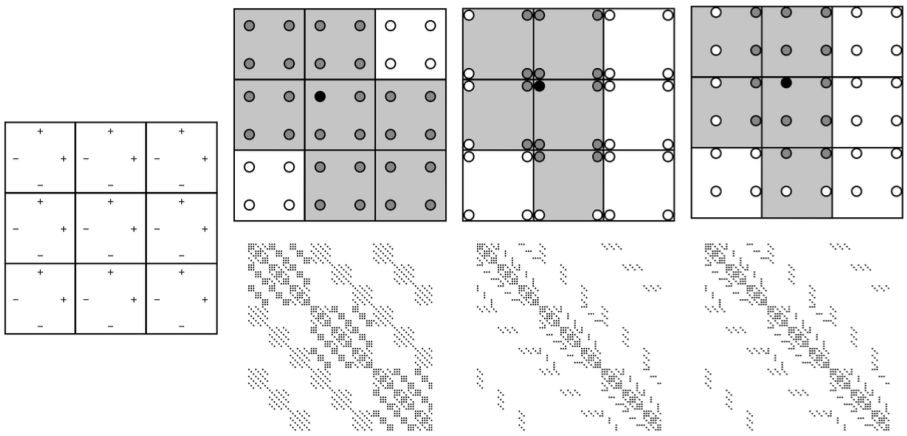


Fig. 5 Comparison of sparsity patterns of the LDG Laplacian operator. Left to right: Mesh $p = 1$, open nodes, closed nodes, half-closed nodes. Half-closed nodes produce an identical sparsity pattern to closed nodes, while open nodes produce a more dense Laplacian operator [7].

same sparsity pattern is obtained for the Laplacian when using half-closed or closed nodes, whilst a denser operator is obtained when using open nodes. This is the case despite the fact that the divergence operator using half-closed nodes is slightly more dense than the one using closed nodes. A visual example of the Laplacian operator sparsity patterns from [7] is presented in Fig. 5.

V. Numerical Results

In this work, we consider the time-dependent non-dimensional incompressible Navier-Stokes equations,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla P + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \boldsymbol{f},$$
$$\nabla \cdot \boldsymbol{u} = 0,$$

where $\boldsymbol{u}$ and P are the non-dimensionalized velocity and pressure, $\boldsymbol{f}$ is the body force and Re is the Reynolds number. We utilize the fractional step method by Perot to solve the governing equations [9] [10]. The convective terms are integrated in time using the second order explicit Adams-Bashforth method while the diffusion terms are integrated with the implicit Crank-Nicolson method. The steps are as follows (from [9] and [10]). First, we compute the intermediate velocity field

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -((\boldsymbol{u} \cdot \nabla)\boldsymbol{u})^{n+\frac{1}{2}} + \frac{1}{2\mathrm{Re}} \nabla^2 (\boldsymbol{u}^* + \boldsymbol{u}^n) + \boldsymbol{f}. \tag{21}$$

Next we solve the elliptic problem and perform the projection step

$$\nabla^2 \phi^{n+1} = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^* \tag{22}$$

$$\boldsymbol{u^{n+1}} = \boldsymbol{u}^* - \Delta t \nabla \phi^{n+1}, \tag{23}$$

and finally we use the following expression for the pressure update, where $P^{n+\frac{1}{2}}$ is the time centered pressure

$$(M + \frac{\Delta t}{2\mathrm{Re}} \nabla^2) P^{n+\frac{1}{2}} = \phi^{n+1}. \tag{24}$$

A. Taylor-Green Vortex

We first perform convergence tests using the Taylor-Green vortex problem to verify the high-order accuracy of our method. The problem is defined on the domain $\Omega = [0, 2\pi]^2$ with periodic boundary conditions. The exact solution is given by

$$u = \sin(x)\cos(y)e^{-2t/\mathrm{Re}}, \tag{25}$$

$$v = -\cos(x)\sin(y)e^{-2t/\mathrm{Re}}, \tag{26}$$

$$P = \frac{1}{4}(\cos(2x) + \cos(2y))e^{-4t/\mathrm{Re}}. \tag{27}$$

The Reynolds number is set to $\mathrm{Re} = 20$ and the final time is $T = 1$. The time step is set to $\Delta t = 1 \times 10^{-5}$ to ensure the spatial errors dominate over the temporal errors. First, we used a polynomial order of $p = 2$ and refined the element size starting from the coarsest size of $h = 1.2566$. We also performed the convergence test using closed nodes (Gauss-Lobatto) for comparison. The results are shown in Fig. 6. We obtain $p + 1$ convergence in both velocity and pressure using HCDG. HCDG appears to be more accurate by a constant in velocity and shows a higher order of accuracy in pressure. It has been previously shown that solutions of higher accuracy are obtained at Gauss-Radau nodes for LDG discretisations in 1D [11], although we are not aware of other results establishing this more generally for other equations and in higher dimensions. This is something we plan to explore further in a future publication.

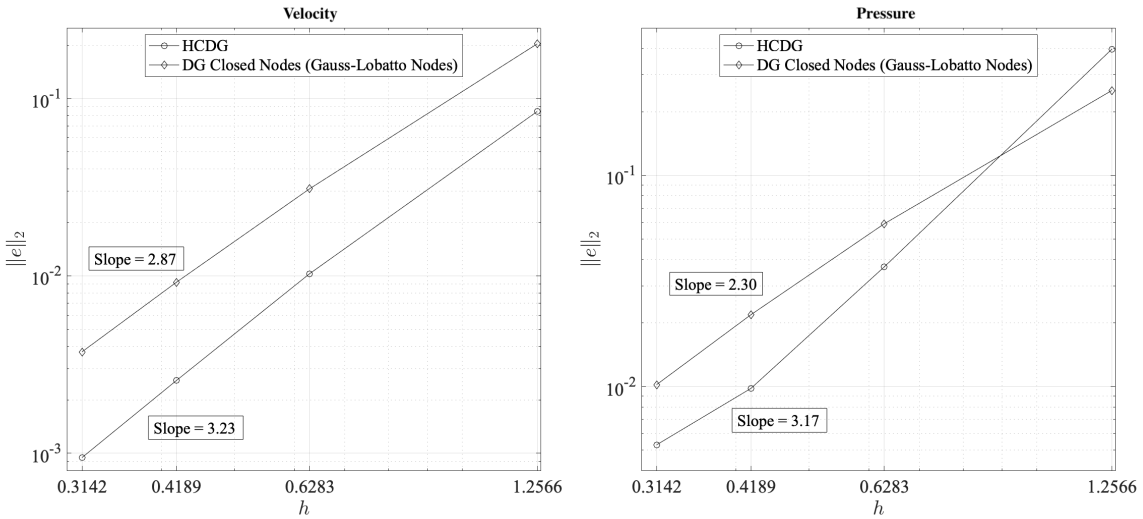


Fig. 6 L^2 Error with h refinement for u-velocity (left) and pressure (right) with Polynomial Order $p = 2$

We also performed a p-convergence test where we discretize the domain with a constant number of elements of $5 \times 5 = 25$ elements for a range of polynomial orders. The results shown in Fig. 7 demonstrate the expected exponential convergence for both velocity and pressure with respect to polynomial degree.

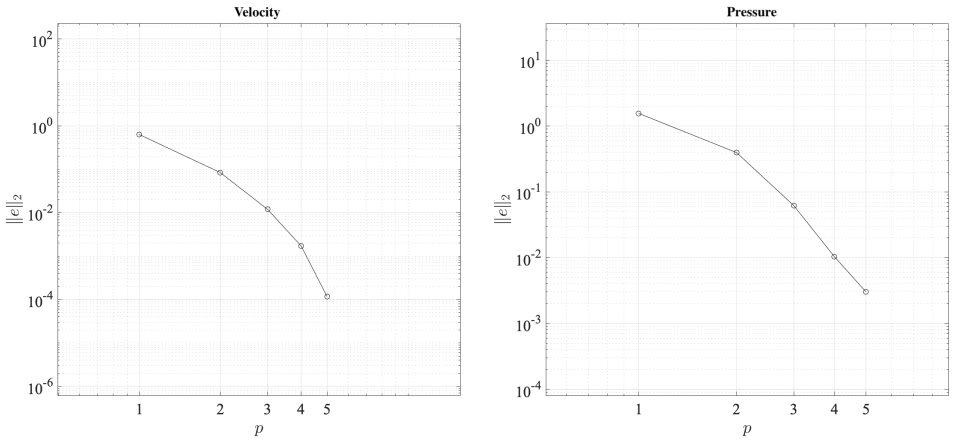


Fig. 7 L^2 Error with p refinement for u-velocity (left) and pressure (right) with element Size $h = 1.2566$

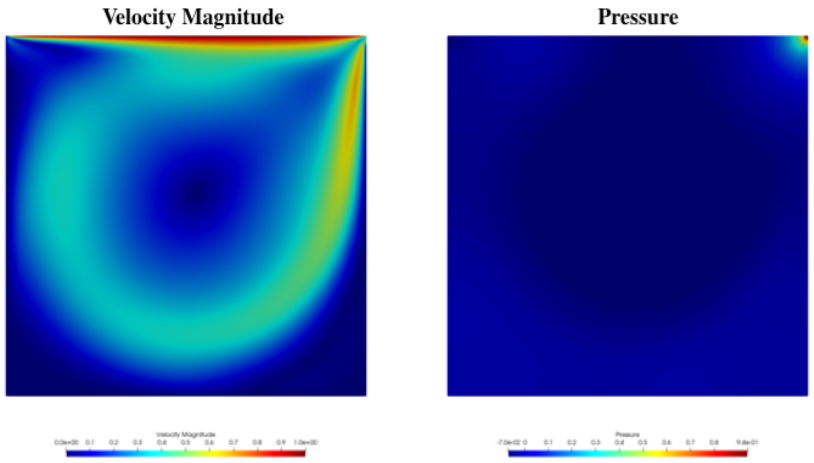


Fig. 8 Velocity magnitude (left) and pressure (right) for the lid driven cavity problem with Re = 1000. Domain is discretized with 32×32 elements with a polynomial order of $p = 3$.

B. Lid-driven Cavity Flow

We also test our implementation on the lid-driven cavity flow problem. The domain is a box $\Omega = [0, 1]^2$ discretized with 32×32 elements in each direction and a polynomial order of $p = 3$. The top wall is moving ($u = 1$ and $v = 0$) and the walls elsewhere are non-moving ($u = v = 0$). The initial condition is $u = v = 0$. The Reynolds number is Re = 1000 and the simulation is run until a final time of $T = 30$ with a time step of $\Delta t = 8 \times 10^{-4}$.

The velocity, pressure, streamlines and vorticity are presented in Figs. 8 and 9. The results were also compared with the reference data from [12]. The comparison of the velocity components along the geometric center lines is shown in Fig. 10. The numerical results show good agreement with the reference data.

C. Flow over NACA0012 Airfoil

The final test case we explore is the flow over a NACA0012 airfoil with an angle of attack $\alpha = 30°$ at a Reynolds number of Re = 1000. The grid is shown in Fig. 11. The left, top and bottom boundaries are inlets with $u = 1$ and $v = 0$. Outlet boundary conditions are imposed on the right boundary and wall boundary conditions are imposed on the airfoil. The initial condition is set to a uniform velocity of $u = 1$ and $v = 0$. We run the simulation with a polynomial order of $p = 3$ with a polynomial order of the geometry of $q = 3$ for a total time of $T = 3$ with a time step of $\Delta t = 1 \times 10^{-4}$.

Figures 12 to 17 show the flow over the airfoil at different times during the simulation. These results are meant

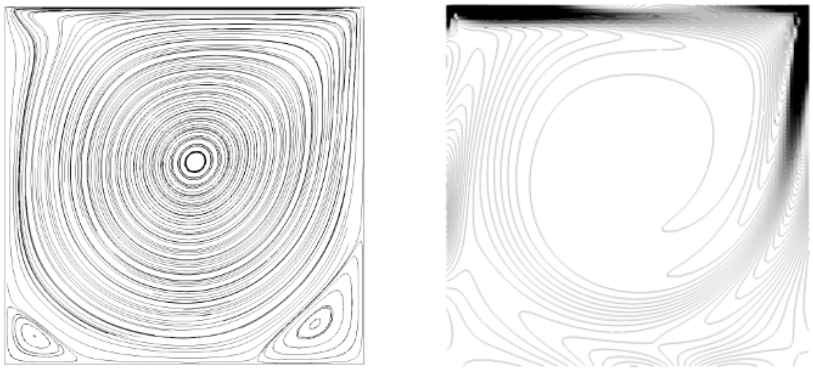


Fig. 9 Streamlines (left) and vorticity contours (right) for the lid driven cavity problem. Parameters as in Figure 8.

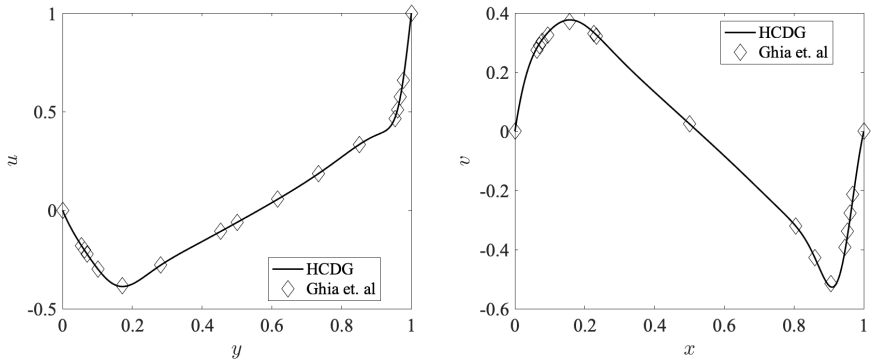


Fig. 10 Comparison of velocity components along geometric centers to reference data [12] at $\mathrm{Re} = 1000$**. Left: along vertical centerline, right: along horizontal centerline.**

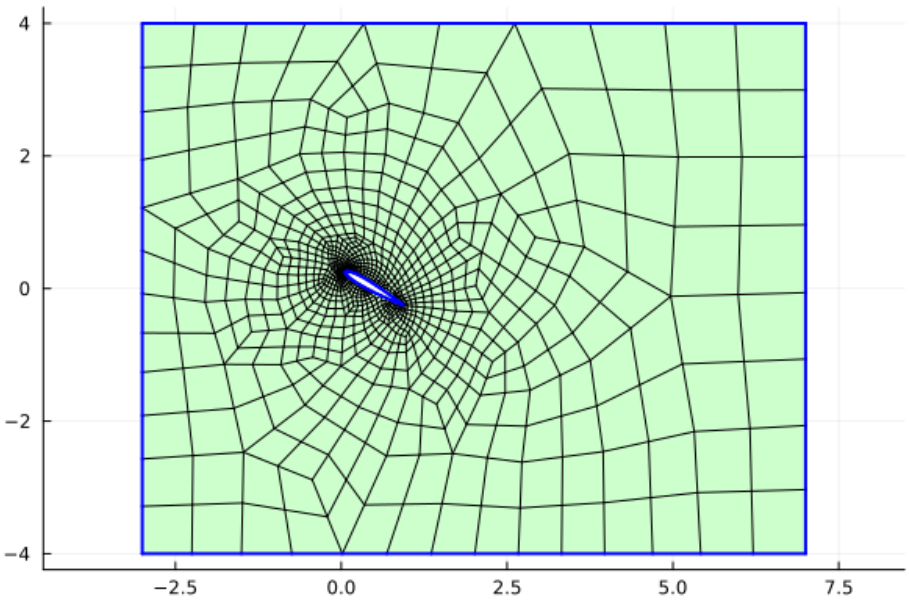


Fig. 11 Grid for the flow over NACA 0012 problem at AoA $\alpha = 30°$. Domain discretized with 1008 elements with polynomial order $p = 3$. Simulation is run until a final time of $T = 3$ with a timestep of $\Delta t = 1 \times 10^{-4}$.

to demonstrate that the method is capable of simulating external aerodynamic flows at moderate Reynolds numbers without additional stabilization.

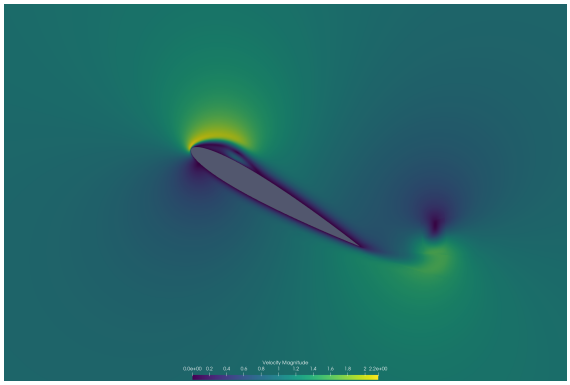


Fig. 12 Velocity magnitude at $T = 0.5$. Parameters as in Figure 11.

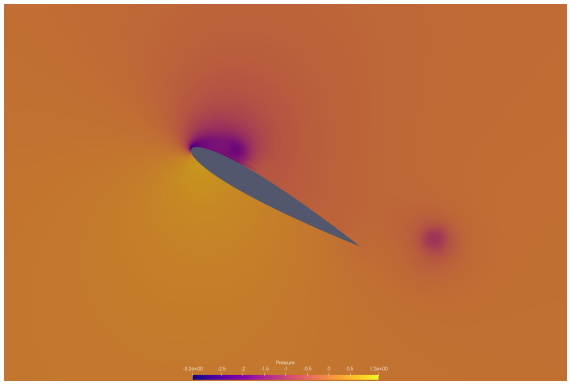


Fig. 13 Pressure at $T = 0.5$. Parameters as in Figure 11.

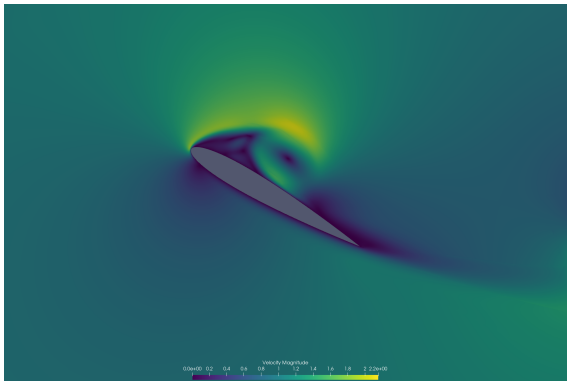


Fig. 14 Velocity magnitude at $T = 1.5$. Parameters as in Figure 11.

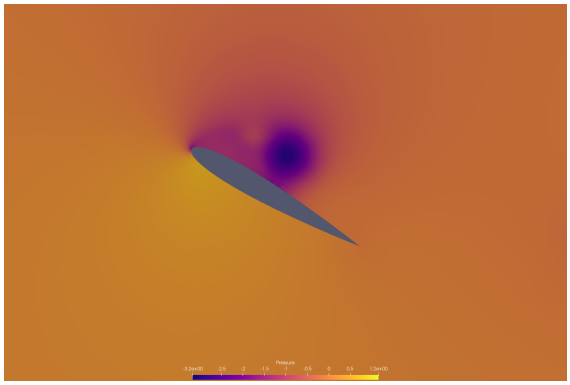


Fig. 15 Pressure at $T = 1.5$. Parameters as in Figure 11.

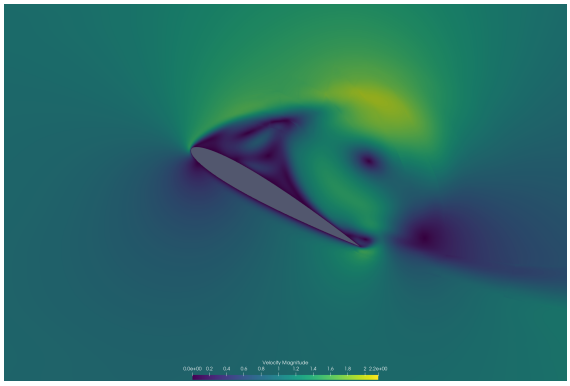


Fig. 16 Velocity magnitude at $T = 3.0$. Parameters as in Figure 11.

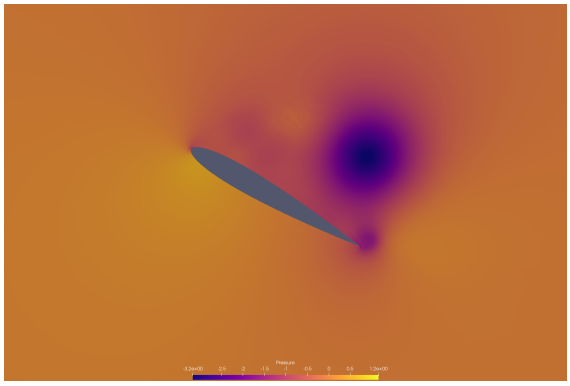


Fig. 17 Pressure at $T = 3.0$. Parameters as in Figure 11.

VI. Conclusion

In this work, we have applied the newly developed half-closed discontinuous Galerkin method to the simulation of incompressible flows. The use of half-closed nodes obtains a combination of the advantages of open and closed nodes such as efficient operator construction and good sparsity patterns. We performed a variety of tests using a fractional step method to demonstrate its high-order convergence and feasibility for aerodynamic simulations. In the future, we wish to take advantage of the sparsity and communication patterns of HCDG to construct efficient linear solvers. Furthermore, we will run simulations at higher Reynolds numbers.

Acknowledgments

The authors would like to acknowledge the Lawrence Berkeley National Laboratory and the Natural Sciences and Engineering Research Council of Canada (NSERC). The work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and in part by the National Science Foundation under Grant DMS-2309596.

References

[1] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P., Van Leer, B., and Visbal, M., "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, p. 811–845. https://doi.org/10.1002/fld.3767, URL https://onlinelibrary.wiley.com/doi/10.1002/fld.3767.

[2] Hesthaven, J. S., and Warburton, T., *Nodal Discontinuous Galerkin Methods*, Texts in Applied Mathematics, Vol. 54, Springer,

New York, 2008. https://doi.org/10.1007/978-0-387-72067-8, URL https://doi.org/10.1007/978-0-387-72067-8, Algorithms, Analysis, and Applications.

[3] Cockburn, B., and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *J. Sci. Comput.*, Vol. 16, No. 3, 2001, pp. 173–261. https://doi.org/10.1023/A:1012873910884, URL https://doi.org/10.1023/A:1012873910884.

[4] Cockburn, B., Gopalakrishnan, J., and Lazarov, R., "Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems," *SIAM Journal on Numerical Analysis*, Vol. 47, No. 2, 2009, p. 1319–1365. https://doi.org/10.1137/070706616, URL http://epubs.siam.org/doi/10.1137/070706616.

[5] Kopriva, D. A., and Kolias, J. H., "A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows," *Journal of Computational Physics*, Vol. 125, No. 1, 1996, p. 244–261. https://doi.org/10.1006/jcph.1996.0091, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999196900911.

[6] Huynh, H. T., "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods," *18th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Miami, Florida, 2007. https://doi.org/10.2514/6.2007-4079, URL https://arc.aiaa.org/doi/10.2514/6.2007-4079.

[7] Pan, Y., and Persson, P.-O., "Half-closed discontinuous Galerkin discretisations," , Nov. 2024. https://doi.org/10.48550/arXiv.2405.12383, URL http://arxiv.org/abs/2405.12383, arXiv:2405.12383 [cs, math].

[8] Cockburn, B., and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, p. 2440–2463. https://doi.org/10.1137/S0036142997316712, URL http://epubs.siam.org/doi/10.1137/S0036142997316712.

[9] Perot, J., "An Analysis of the Fractional Step Method," *Journal of Computational Physics*, Vol. 108, No. 1, 1993, p. 51–58. https://doi.org/10.1006/jcph.1993.1162, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999183711629.

[10] Brown, D. L., Cortez, R., and Minion, M. L., "Accurate Projection Methods for the Incompressible Navier–Stokes Equations," *Journal of Computational Physics*, Vol. 168, No. 2, 2001, p. 464–499. https://doi.org/10.1006/jcph.2001.6715, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999101967154.

[11] Liu, X., Zhang, D., Meng, X., and Wu, B., "Superconvergence of local discontinuous Galerkin methods with generalized alternating fluxes for 1D linear convection-diffusion equations," *Science China Mathematics*, Vol. 64, No. 6, 2021, p. 1305–1320. https://doi.org/10.1007/s11425-019-1627-7, URL https://link.springer.com/10.1007/s11425-019-1627-7.

[12] Ghia, U., Ghia, K., and Shin, C., "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method," *Journal of Computational Physics*, Vol. 48, No. 3, 1982, p. 387–411. https://doi.org/10.1016/0021-9991(82)90058-4.

Chapter 5

Eliminated p-Multigrid for Half-Closed Discontinuous Galerkin Methods

The previous chapter focused on discretization. Now we focus on a new linear solver to improve efficiency. We test this new linear solver using the half-closed discontinuous Galerkin discretization on both structured and unstructured grids. The following is a journal paper in preparation that will be submitted in the near future.

Contributions

The author wrote this paper in collaboration with the other co-authors. The author implemented the eliminated p-multigrid method and performed all of the simulations.

Nomenclature

A	Matrix
b	Linear system right hand side
C_D	Boundary stabilization parameter
D	Divergence
d	Dimension
e	Error vector
$F(u)$	Flux
$\hat{F}(u)$	Numerical flux
$\boldsymbol{f}$	Forcing function
G	Gradient
g_D, g_N	Dirichlet and Neumann boundary value
h	Element size
K_n	Element
L	Laplacian
M	Mass matrix
N	Number of elements
P	Pressure
p	Polynomial order
Re	Reynolds number
r	Residual vector
S_n^m	Switch function
t	Time
u	Solution
V_h	Discontinuous function space
v	Test function
x_d	Spatial coordinate
α	Angle of attack
$\partial\Omega$	Boundary
ρ	Relative residual reduction factor
ϕ	Basis function
Ω	Domain

Eliminated p-Multigrid for Half-Closed Discontinuous Galerkin Methods

Amit Nayak[a,c], Yulong Pan[b,c], Per-Olof Persson[b,c]

[a] *University of Ottawa, Department of Mechanical Engineering, Ottawa, Ontario, Canada*
[b] *University of California, Berkeley, Department of Mathematics, Berkeley, California, USA*
[c] *Lawrence Berkeley National Laboratory, Mathematics Group, Berkeley, California, USA*

Abstract

We present an "eliminated" linear solver designed for local discontinuous Galerkin (LDG) discretizations. We apply static condensation by eliminating nodes based on the LDG switch function, and solve using p-multigrid operating on the eliminated system. A cost analysis was conducted, showing significant savings in key operations in the p-multigrid procedure, such as smoothing and restriction / extension. Numerical tests were performed using the half-closed discontinuous Galerkin (HCDG) discretization on two-dimensional quadrilateral elements. Significant reductions in degrees of freedom were shown for high polynomial orders. Furthermore, we observed a reduction in the number of p-multigrid V-cycles required for convergence compared to traditional p-multigrid for both elliptic problems and incompressible flow. The combination of operating on a reduced size linear system and the reduction in number of V-cycles required leads to a significant improvement in performance.

Keywords: Discontinuous Galerkin, Linear Solvers, p-Multigrid, Static Condensation, Incompressible Flow

1. Introduction

High-order discontinuous Galerkin (DG) methods have risen in popularity due to numerous benefits including arbitrarily high order of spatial accuracy, parallizability due to its local nature and the capacity to support complex geometries through unstructured meshes [1, 2]. However, despite these benefits, the computational cost is still an ongoing challenge to make these methods competitively feasible for industrial applications. In this work, we attempt to address this high cost by developing a more efficient "eliminated" linear solver for DG methods.

Multigrid is an extremely prominent linear solver technique which has been applied to elliptic problems [3]. The multigrid method operates using a hierarchy of grids such that low frequency errors after smoothing on finer grids can be efficiently dealt with by solving for a correction on coarser grids. It has been used extensively for computational fluid dynamics (CFD) in schemes such as the finite volume method [4, 5, 6]. The multigrid method has also been applied to high-order DG methods. In the context of high-order DG methods, there is the h-multigrid where the coarse grids are generated by the coarsening of elements [7, 8], the p-multigrid where the coarsening is done by reducing the polynomial order [9, 10], and the hp-multigrid which combines both approaches [11, 12, 13].

Static condensation is a technique in which degrees of freedom are eliminated such that a reduced linear system can instead be solved [14]. Although static condensation has been used extensively in the finite element method, it has also been applied to DG methods [15]. Static condensation is attractive for use in implicit high-order methods, as the resulting Jacobian matrices

can often be very large.

In our work, we introduce an "eliminated" p-multigrid by applying static condensation through eliminating nodes based on the local discontinuous Galerkin (LDG) [16] switch function, this is combined with using p-multigrid on the eliminated system as a linear solver. The idea is to help mitigate the cost of the high-order DG discretization by operating on a reduced size linear system. A similar approach combining p-multigrid and static condensation was introduced in [17], however this work was based on spectral element method discretizations. To the best of the authors' knowledge, there has been no work for DG discretizations that utilizes p-multigrid operating on a statically condensed system.

We apply our method to the half-closed discontinuous Galerkin method (HCDG) [18]. In this method, "half-closed" nodes are placed on exactly half the boundaries of each element, based on the LDG switch function. The benefits of HCDG include efficient construction of DG operators and optimal sparsity patterns. While our method can be used for standard DG methods using LDG, in this work we focus on two-dimensional quadrilateral elements using the HCDG discretization.

The rest of this work is as follows. In Section 2 we present the HCDG method for both first and second-order operators. In Section 3 we describe our elimination procedure and corresponding costs. Next, in Section 4, we outline the traditional p-multigrid and our eliminated p-multigrid method. We then provide a direct comparison of the costs associated with each method. In Section 5 we first show the benefits obtained by elimination. Next, we apply our method to Poisson problems on both structured and un-

structured grids and compare with the traditional p-multigrid. Finally, we use our eliminated p-multigrid for an incompressible flow case. We conclude and discuss future directions in Section 6.

2. Discontinuous Galerkin Discretization

In this work we use the HCDG method. The HCDG method directly follows the standard DG discretization, however the difference is in the choice of nodes and node placement. We use Gauss-Radau nodes as our solution nodes and our quadrature nodes. Gauss-Radau nodes have the benefit of an extra-degree of quadrature accuracy over the commonly used Gauss-Lobatto nodes. The general DG discretization procedure will now be outlined, while further details on HCDG can be found in [18].

2.1. First Order Equations

The domain Ω is discretized into a set of elements $T_h = \{K_n : \bigcup K_n = \Omega\}$. We then introduce a discontinuous function space:

$$V_h = \{v_h|_K \in V_h(K_n)\}, \tag{1}$$

where for our quadrilateral elements V_h is chosen as the outer-product polynomials of a given degree p. We apply the discretization to the conservation law:

$$\frac{\partial u}{\partial t} + \nabla \cdot \boldsymbol{F}(u) = 0, \tag{2}$$

where u is the conserved quantity and $\boldsymbol{F}(u)$ is the flux function. We multiply Eq. (2) by a test function $v \in V_h$ and apply integration by parts,

$$\frac{\partial}{\partial t}\int_{K_n} uv dx = \int_{K_n} \nabla v \cdot \boldsymbol{F}(u) dx - \int_{\partial K_n} v\hat{\boldsymbol{F}}(u) \cdot \boldsymbol{n} ds, \quad \forall v \in V_h. \tag{3}$$

Since functions are discontinuous along $\partial\Omega$, we have introduced numerical fluxes $\hat{\boldsymbol{F}}$ at the discontinuities. Here, we use the upwind flux as our numerical flux. The solution u is represented as a linear combination of the basis functions $\phi_j \in V_h$,

$$u = \sum_j \phi_j u_j. \tag{4}$$

Furthermore, the test function is chosen from the same family of basis functions $\phi_i \in V_h$, producing the following expression:

$$\sum_n \int_{K_n} \phi_i \phi_j dx \cdot \frac{\partial}{\partial t} u_j = \sum_d \sum_n \int_{K_n} \frac{\partial \phi_i}{\partial x_d} \phi_j dx \cdot F_d(u_j) - \int_{\partial K_n} \phi_i \hat{F}_d(u^{\text{int}}, u^{\text{ext}}) \cdot n_d ds, \tag{5}$$

where d denotes the spatial dimension and u^{int} and u^{ext} are the solution in the interior and exterior of the element, separated by the element boundary with unit normal $\boldsymbol{n}$. The left hand side of Eq. (5) produces the mass matrix

$$M_{ij} = \sum_n \int_{K_n} \phi_i \phi_j dx, \tag{6}$$

and the right hand side produces the discrete divergence operator

$$D^d F_d = \sum_n \int_{K_n} \frac{\partial \phi_i}{\partial x_d} \phi_j F_d(u_j) dx - \int_{\partial K_n} \phi_i \hat{F}_d(u^{\text{int}}, u^{\text{ext}}) \cdot n_d ds. \tag{7}$$

We ultimately obtain the discontinuous Galerkin spatial discretization

$$M\dot{u} = \sum_d D^d \boldsymbol{F}_d. \tag{8}$$

2.2. Second Order Equations

Poisson's equation is used as our model problem for second-order partial differential equations

$$-\nabla^2 u = f. \tag{9}$$

We introduce $\boldsymbol{q}$ as the gradient of u, split into a system of first order partial differential equations, and utilize the local discontinuous Galerkin method [16]

$$\boldsymbol{q} = \nabla u, \tag{10}$$

$$-\nabla \cdot \boldsymbol{q} = f. \tag{11}$$

Next, we multiply Eq. (10) and Eq. (11) by test functions $v \in V_h$ and $\boldsymbol{\tau} \in V_h^d$ respectively, and integrate by parts

$$\int_{K_n} \boldsymbol{q} \cdot \nabla v dx - \int_{\partial K_n} v \hat{\boldsymbol{q}} \cdot \boldsymbol{n} ds = \int_{K_n} f v dx \quad \forall v \in V_h, \tag{12}$$

$$\int_{K_n} (\boldsymbol{q} + u\nabla) \cdot \boldsymbol{\tau} dx - \int_{\partial K_n} \hat{u} \boldsymbol{\tau} \cdot \boldsymbol{n} ds = 0 \quad \forall \boldsymbol{\tau} \in V_h^d. \tag{13}$$

We use a so-called "switch function" to specify the numerical fluxes. The switch function either has a value of -1 or $+1$. The switch function is defined on each boundary of each element with the constraint that on any boundary separating elements K_n and K_m, the summation of the switch functions from neighboring elements is $S_n^m + S_m^n = 0$. Based on the switch function, we utilize the following upwind-downwind flux in Eq. (14) and Eq. (15)

$$\hat{\boldsymbol{q}} = \begin{cases} \boldsymbol{q}|_{K_n} & \text{if } S_n^m < 0, \\ \boldsymbol{q}|_{K_m} & \text{if } S_m^n > 0, \end{cases} \tag{14}$$

and

$$\hat{u} = \begin{cases} u|_{K_n} & if\ S_n^m > 0, \\ u|_{K_m} & if\ S_m^n < 0. \end{cases} \tag{15}$$

Boundary conditions are imposed weakly. The following fluxes are used to imposed either Dirichlet or Neumann boundary conditions.

$$\hat{\boldsymbol{q}} = \begin{cases} \boldsymbol{q} - C_D(u - g_D)\boldsymbol{n} & \text{on } \Gamma_D, \\ g_N \boldsymbol{n} & \text{on } \Gamma_N, \end{cases} \tag{16}$$

and

$$\hat{u} = \begin{cases} g_D & \text{on } \Gamma_D, \\ u & \text{on } \Gamma_N, \end{cases} \tag{17}$$

where g_D and g_N are the values of the Dirichlet and Neumann boundary conditions respectively, and $C_D > 0$ is a stabilization parameter for Dirichlet boundary conditions. In this work we use $C_D = \frac{1000}{h}$, where h is the boundary element size, however lower values can be used. The system of equations in operator form (regardless of spatial dimension) is written as:

$$\begin{bmatrix} M & G^d \\ -D^d & \end{bmatrix} \begin{bmatrix} q^d \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ f \end{bmatrix}. \tag{18}$$

The divergence and gradient operator are related by the adjoint property $D = -G^T$. To solve for u, we take the Schur complement of Eq. (18) to obtain the Laplacian operator

$$L = \sum_d D^d M^{-1} G^d. \tag{19}$$

3. Static Condensation

3.1. Method

The discretized system can be partitioned into independent and dependent nodes [14]

$$\begin{pmatrix} A_{II} & A_{ID} \\ A_{DI} & A_{DD} \end{pmatrix} \begin{pmatrix} u_I \\ u_D \end{pmatrix} = \begin{pmatrix} b_I \\ b_D \end{pmatrix}. \tag{20}$$

Then, the Schur complement of the system can be calculated:

$$\tilde{A} = A_{II} - A_{ID} A_{DD}^{-1} A_{DI}, \tag{21}$$

$$\tilde{b} = b_I - A_{ID} A_{DD}^{-1} b_D. \tag{22}$$

The solution at the independent nodes can then be found by solving the following linear system:

$$\tilde{A} u_I = \tilde{b}. \tag{23}$$

After obtaining the solution at the independent nodes, the solution at the dependent nodes can be directly solved for using the expression

$$u_D = A_{DD}^{-1}(b_D - A_{DI} u_I). \tag{24}$$

Specifically in our work, the independent nodes are the boundary nodes where the LDG switch function is positive ($S_n^m > 0$). The dependent nodes are all the other nodes in the element. Examples of the elimination on both half-closed and closed nodes elements for both simplex and quadrilateral elements are shown in Figure 1.

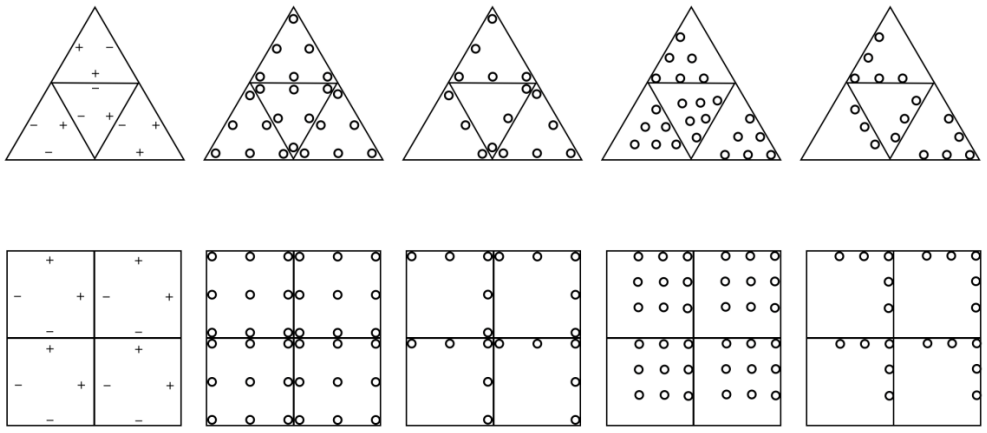


Figure 1: Static condensation for p=2 simplex (top) and quadrilateral (bottom) elements. Left to right: LDG switch functions, closed nodes, closed nodes after static condensation, half-closed nodes, half-closed nodes after static condensation. Reprinted from [18].

3.2. Costs

The cost of eliminating and reconstructing the full solution will now be discussed. The full problem is discretized with N elements with polynomial order p. The size of the full linear system A_p is $N(p+1)^2 \times N(p+1)^2$. We will often consider our operator as an $N \times N$ block matrix where the blocks are dense of size $(p+1)^2 \times (p+1)^2$. For a two-dimensional quadrilateral element after applying static condensation, $(p+1)^2$ nodes is reduced to $(2p+1)$ independent nodes and p^2 dependent nodes. The size of the components from Eq. (20) are shown in Table 1.

The cost to form the eliminated linear operator $\tilde{A}$ using Eq. (21) will now be outlined. The matrix A_{DD}^{-1} is calculated once and stored. However, A_{DD}^{-1} is block diagonal due to the choice of nodes to eliminate, so the inverse can be precomputed efficiently element-wise. The cost of calculating the

Table 1: Sizes of operator partitions from static condensation for 2D quadrilateral elements.

Partition	Size
A_{II}	$N(2p+1) \times N(2p+1)$
A_{ID}	$N(2p+1) \times Np^2$
A_{DI}	$Np^2 \times N(2p+1)$
A_{DD}	$Np^2 \times Np^2$
u_I and b_I	$N(2p+1) \times 1$
u_D and b_D	$Np^2 \times 1$

inverse of the block diagonals is $O(Np^6)$. Next, $\tilde{A}$ is computed by a series of matrix multiplications and subtractions. For the second term, A_{DI} and A_{ID} are block sparse, however, we will assume the blocks are dense to obtain a conservative estimate for the cost. Based on the sizes in Table 1, the cost for each matrix multiplication is $O(Np^5)$. The cost for the matrix subtraction is $O(Np^2)$. Therefore, the total cost for forming the condensed system after precomputations is $O(Np^5)$.

Next the cost to form the eliminated right hand side will be outlined. Since the matrix multiplication $A_{ID}A_{DD}^{-1}$ is already required to calculate $\tilde{A}$, it will be neglected for the cost of forming $\tilde{b}$ since $A_{ID}A_{DD}^{-1}$ is stored and since $\tilde{b}$ will be calculated multiple times during the V-Cycle. The cost for $\tilde{b}$ includes the matrix-vector multiplication of $A_{ID}A_{DD}^{-1}$ and b_D: $O(Np^3)$, and the vector subtraction $O(Np)$. Thus the cost to form $\tilde{b}$ is $O(Np^3)$.

The cost to calculate the solution for the dependent degrees of freedom u_D is as follows. The matrix multiplication $A_{DD}^{-1}A_{DI}$ is precomputed as

$A_{DD}^{-1}A_{DI}$ does not change. The cost to calculate u_D comes from the matrix multiplication and subtraction inside the parenthesis in Eq. (24): $O(Np^3)$ and the matrix multiplication of A_{DD}^{-1} afterwards: $O(Np^4)$. Thus the total cost of computing u_D is $O(Np^4)$. A summary of all the costs associated with the static condensation are presented in the Table 2.

Table 2: Precomputation and construction costs for forming statically condensed system.

Operation	**Big-O**
Construct $\tilde{A}$	$O(Np^5)$
Construct $\tilde{b}$	$O(Np^3)$
Construct u_d	$O(Np^4)$
Precompute A_{DD}^{-1}	$O(Np^6)$
Precompute $A_{DD}^{-1}A_{DI}$	$O(Np^5)$
Precompute $A_{ID}A_{DD}^{-1}$	$O(Np^5)$

4. p-Multigrid

4.1. p-Multigrid Method

In the p-multigrid method, instead of coarsening by element size coarsening is done by reducing polynomial order. In our work we use V-cycles, dividing the polynomial order by 2 and take the floor to obtain the coarse grid levels as seen in Fig. 2.

In our work, we use a direct sparse method as our bottom level solver for $p = 1$. There are other possible bottom solvers that can be used, such as geometric/algebraic multigrid or preconditioned iterative solvers. However, since the focus of this paper is on the p-multigrid we have not investigated

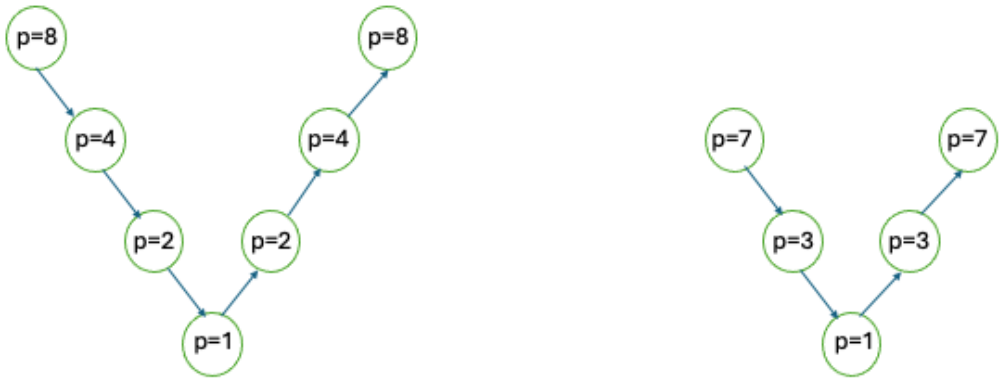


Figure 2: Example V-cycles, left: top level $p = 8$, right: top level $p = 7$. Subsequent lower levels are determined by dividing by two and taking the floor.

these choices. Our p-multigrid pseudocode is outlined in Algorithm 1. Next, the smoothers and restriction/prolongation operators will be discussed.

Algorithm 1 p-Multigrid V-Cycle: VG(u_p^{old}, b_p)

Input: u_p^{old}, b_p

Output: u_p^{new}

if $p \neq p_{\min}$ **then**

Apply block smoothing ν_1 times on $A_p u_p = b_p$

$r_{p_{\text{coarse}}} \leftarrow I_p^{p_{\text{coarse}}}(b_p - A_p u_p)$

$e_{p_{\text{coarse}}} \leftarrow \text{VG}(0, r_{p_{\text{coarse}}})$

$u_p \leftarrow u_p + I_{p_{\text{coarse}}}^p e_{p_{\text{coarse}}}$

Apply block smoothing ν_2 times on $A_p u_p = b_p$

return u_p

else

Solve $A_p u_p = b_p$ using bottom solver.

return u_p

end if

4.1.1. Restriction and Prolongation Operators

The restriction and prolongation operators are used to transfer the residual and errors between polynomial levels. The prolongation operator from level p_{coarse} to level p acts on a vector:

$$I_{p_{\text{coarse}}}^p u_{p_{\text{coarse}}} = u_p. \tag{25}$$

The prolongation operator used in our work is element-wise polynomial interpolation. The restriction operator is defined as the transpose of the prolongation operator:

$$I_p^{p_{\text{coarse}}} = I_{p_{\text{coarse}}}^{p^T}. \tag{26}$$

The restriction and prolongation operations are performed element-wise, therefore the corresponding matrices are block sparse. Thus, the cost of restriction or prolongation from one polynomial level to another is $O(Np^2p_{coarse}^2)$.

4.1.2. Smoothers

We utilize block based smoothers such as block damped Jacobi or block Gauss-Seidel due to the discontinuous Galerkin discretization:

$$u^{(i+1)} = u^{(i)} + \omega S^{-1} r^{(i)}. \tag{27}$$

For block damped Jacobi, S is the block diagonal D. For block Gauss-Seidel, S is the lower triangular portion of the operator and the block diagonal $(L + D)$. The damping factor is ω and the residual is $r^{(i)} = b - Au^{(i)}$. The cost of applying block damped Jacobi will be now analyzed, however the analysis also applies to block Gauss-Seidel as the cost is on the same order. First, the inverse of the block diagonal $S = D^{-1}$ will be pre-computed and stored as it is constant, the cost in Big-O notation is $O(Np^6)$. Now the cost of an individual smoother iteration after S has been pre-computed will be discussed. The cost to calculate the residual includes the matrix-vector multiplication $O(Np^4)$ and vector subtraction $O(Np^2)$, ultimately resulting in a cost of $O(Np^4)$. The cost to update to $u^{(i+1)}$ involves another matrix-vector multiplication and vector addition. Thus, the cost for an iteration of block damped Jacobi is: $O(Np^4)$. If block Gauss-Seidel was used as the smoother, the cost would be on the same order of $O(Np^4)$.

4.2. Eliminated p-Multigrid

The core idea behind the eliminated p-multigrid is to operate (smoothing/bottom solving) on the statically condensed or reduced degree of freedom

linear system. The eliminated p-multigrid pseudocode is outlined in Algorithm 2. First, some specific aspects of the eliminated p-multigrid will be presented. Smoothing, restriction, prolongation, and bottom solving is done on the statically condensed system. The full solution is constructed only at the very end of the V-cycle. At the top level $\tilde{b}$ is pre-calculated since b is constant. However, at the lower polynomial levels, $\tilde{b}$ must be calculated before smoothing since the right hand side is the residual from the upper level, which changes every V-cycle. Each aspect of the eliminated p-multigrid will now be analyzed. A visual comparison of the operating grids for the traditional p-multigrid and eliminated p-multigrid is presented in Figure 3.

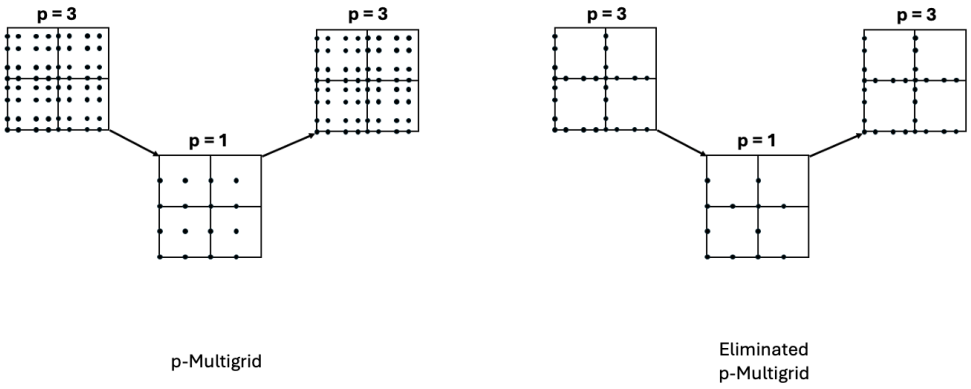


Figure 3: Visual comparison of traditional p-multigrid and eliminated p-multigrid. Left: traditional p-multigrid with top level $p = 3$ and bottom level $p = 1$. Right: eliminated p-multigrid with top level $p = 3$ and bottom level $p = 1$.

Algorithm 2 Eliminated p-Multigrid V-Cycle: $\mathrm{VG}_{\mathrm{elim}}(u_p^{\mathrm{old}}, b_p)$

Input: u_p^{old}, b_p

Output: u_p^{new}

if $p = p_{\max}$ **then**

Apply block smoother ν_1 times on $\tilde{A}_p u_{p_I} = \tilde{b}_p$

$r_{p_{\mathrm{coarse}}} \leftarrow I_{p_I}^{p_{\mathrm{coarse}}}(\tilde{b}_p - \tilde{A}_p u_{p_I})$

$e_{p_{coarse_I}} \leftarrow \mathrm{VG}_{\mathrm{elim}}(0, r_{p_{\mathrm{coarse}}})$

$u_{p_I} \leftarrow u_{p_I} + P_{p_{coarse_I}}^{p_I} e_{p_{coarse_I}}$

Apply block smoother ν_2 times on $\tilde{A}_p u_{p_I} = \tilde{b}_p$

$u_{p_D} \leftarrow A_{p_{DD}}^{-1} b_{p_D} - A_{p_{DD}}^{-1} A_{p_{DI}} u_{p_I}$ ▷ Only construct full solution at end of V-Cycle

return u_p

else if $p \neq p_{\min}$ **then**

$\tilde{b_p} \leftarrow b_{p_I} - A_{p_{ID}} A_{p_{DD}}^{-1} b_{p_D}$

Apply block smoother ν_1 times on $\tilde{A}_p u_{p_I} = \tilde{b}_p$

$r_{p_{\mathrm{coarse}}} \leftarrow I_{p_I}^{p_{\mathrm{coarse}}}(\tilde{b}_p - \tilde{A}_p u_{p_I})$

$e_{p_{coarse_I}} \leftarrow \mathrm{VG}_{\mathrm{elim}}(0, r_{p_{\mathrm{coarse}}})$

$u_{p_I} \leftarrow u_{p_I} + P_{p_{coarse_I}}^{p_I} e_{p_{coarse_I}}$

Apply block smoother ν_2 times on $\tilde{A}_p u_{p_I} = \tilde{b}_p$

return u_{p_I}

else

$\tilde{b_p} \leftarrow b_{p_I} - A_{p_{ID}} A_{p_{DD}}^{-1} b_{p_D}$

Solve $\tilde{A}_p u_{p_I} = \tilde{b}_p$ using bottom solver.

return u_{p_I}

end if

4.2.1. Eliminated Restriction and Prolongation Operators

The eliminated prolongation operator is essentially a subset of the prolongation for the traditional p-multigrid

$$I^{p_{\mathrm{I}}}_{p_{\mathrm{coarse_I}}} = I^{p}_{p_{\mathrm{coarse}}}[\mathrm{Independent}, \mathrm{Independent}_{\mathrm{coarse}}], \tag{28}$$

which transfers the condensed (independent nodes) coarse grid error to the fine grid as a condensed correction. As with the traditional p-multigrid, the operator is block sparse. Now, the cost scales with the size of the condensed block of $(2p+1) \times (2p_{\mathrm{coarse}}+1)$. Thus, the cost of applying the eliminated prolongation operator is $O(Np_{coarse}p)$. The eliminated restriction operator is also a subset of the restriction operator from the traditional p-multigrid.

$$I^{p_{\mathrm{coarse}}}_{p_{\mathrm{I}}} = I^{p_{\mathrm{coarse}}}_{p}[:, \mathrm{Independent}]. \tag{29}$$

However, it transfers the condensed residual from the fine grid to the full residual on the coarse grid, since the full residual is required on the coarse grid to calculate $\tilde{b}$. The cost of the restriction operation is: $O(Np^2_{coarse}p)$.

4.2.2. Eliminated Smoothers

The smoothing costs follows an identical pattern to smoothing on the full system; however, the costs now scale with the size of the condensed blocks and system. Similarly to the cost analysis for the traditional p-multigrid, we take block damped Jacobi from Eq. (27) as an example. The cost of pre-computing $S = D^{-1}$ is $O(Np^3)$, since the block size is $(2p+1) \times (2p+1)$. The cost to calculate the residual vector is now: $O(Np^2)$. This is also the cost to update to u^{i+1} as it also involves a matrix-vector multiplication and vector addition. Thus, the total cost for an iteration of block damped Jacobi

on the eliminated system is $O(Np^2)$. As with the non-eliminated case, the cost for block Gauss-Seidel are on the same order when smoothing on the eliminated system.

4.3. Comparison of Costs

The estimated costs of the traditional p-multigrid and the eliminated p-multigrid are compared in Table 3. We observe valuable efficiency gains by operating on the eliminated system. Smoothing on the eliminated system is more efficient by a factor of p^2 and precomputing the eliminated smoothing operator is more efficient by a factor of p^3. We also observe gains for restriction and prolongation in the eliminated p-multigrid. Restriction is more efficient by a factor of p and prolongation is more efficient by a factor of pp_{coarse}. For the eliminated p-multigrid there is still the precomputing $\tilde{b}$ at the start of each polynomial level. However, this is still a factor of p cheaper than a smoothing iteration on the full system. Furthermore, constructing u_D is only done at the end of the V-cycle, and is roughly equivalent to smoothing on the full system. Overall, these savings are promising as multiple pre-smoothing, post-smoothing, restriction, and prolongation are prominent components of a V-cycle.

5. Numerical Results

5.1. Static Condensation

To quantify the effect of eliminating on the linear system, we apply static condensation to an unstructured grid seen in Fig. 4. We obtain the ratio of degrees of freedom and the ratio of non-zero entries from $p = 1$ to $p = 7$.

Table 3: Comparison of operation costs for traditional p-multigrid and eliminated p-multigrid 2D quadrilaterals elements.

Operation	**p-Multigrid**	**Eliminated p-Multigrid**
Smoothing	$O(Np^4)$	$O(Np^2)$
Precompute Smoothing Operator	$O(Np^6)$	$O(Np^3)$
Restriction	$O(Np^2p_{\text{coarse}}^2)$	$O(Npp_{\text{coarse}}^2)$
Prolongation	$O(Np^2p_{\text{coarse}}^2)$	$O(Npp_{\text{coarse}})$
Precompute $\tilde{b}$	N/A	$O(Np^3)$
Construct u_D at end of V-Cycle	N/A	$O(Np^4)$

The results are shown in Fig. 5. We observe a drastic reduction in both the degrees of freedom and non-zero entries as the polynomial order grows. This is promising for the eliminated p-multigrid as smoothing on the higher levels of the V-cycle become much more economical. For lower polynomial orders there is still a significant reduction, in $p = 2$ approximately a factor of two reduction in degrees of freedom and non-zero entries.

5.2. Poisson's Equation - Structured Grid

We compare our eliminated p-multigrid method with traditional p-multigrid using Poisson's equation on a structured grid. The domain is defined $\Omega \in [-1, 1]^2$ and is discretized with 21 elements in each direction resulting in a total number of elements $N = 441$. We impose Dirichlet boundary conditions and the solution is chosen as

$$u(x, y) = e^{\sin(\pi x)\sin(\pi y)}. \tag{30}$$

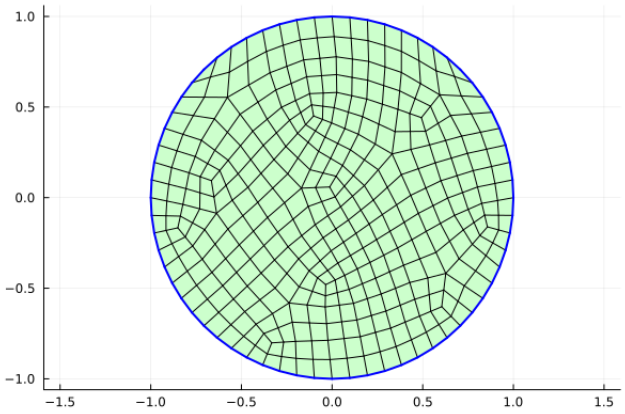


Figure 4: Circular domain with unit radius, unstructured grid with $N = 356$ quadrilateral elements.

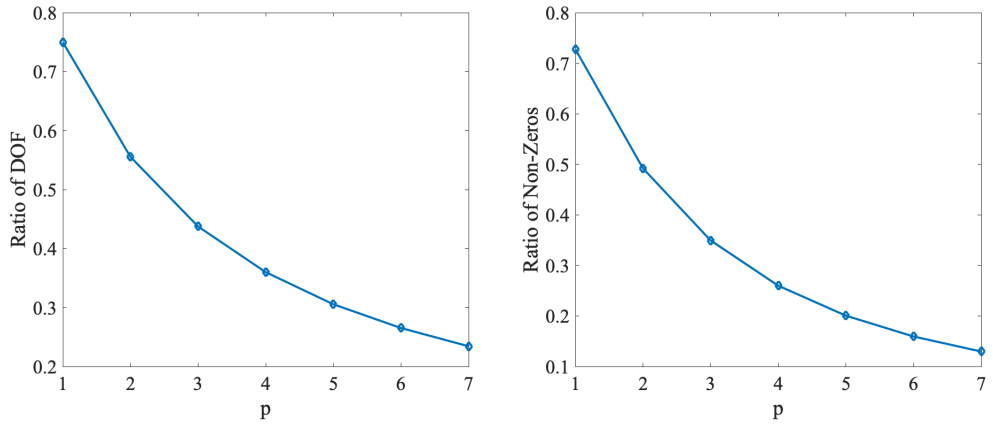


Figure 5: Ratio of degrees of freedom in eliminated system to degrees of freedom in non-eliminated system (left). Ratio of non-zero entries in eliminated system to non-zero entries in non-eliminated system (right).

We iterate with the multigrid method until the relative residual $\frac{\|b-Au\|}{\|b\|}$ is no greater than 10^{-8}. Different polynomial orders were tested, $p \in [2, 3, 4, 5]$. The relative residual with respect to the V-cycles for both the eliminated

and traditional p-multigrid for $p = 5$ is shown in Fig. 6. The eliminated p-multigrid terminates in approximately half the number of V-cycles. The total number of V-cycles required to reach the set tolerance for each polynomial order tested is shown in Fig. 7. The eliminated p-multigrid overall requires less V-cycles to converge, approaching almost half the number of V-cycles required for higher polynomial order.

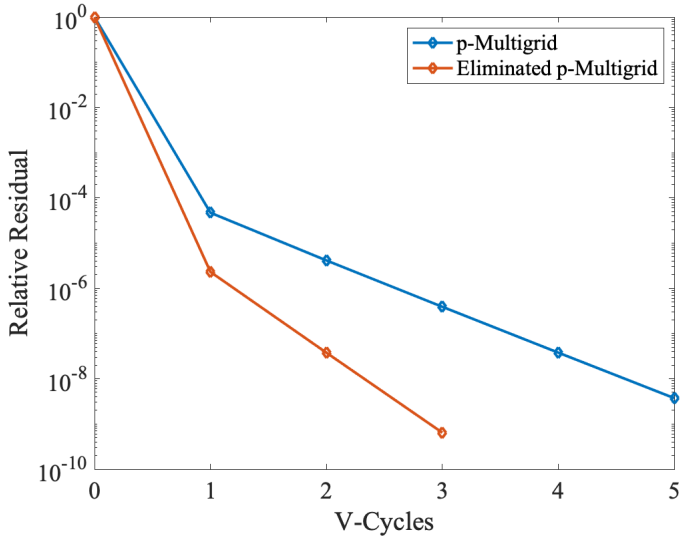


Figure 6: Relative residual vs V-cycles for $p = 5$ and $N = 441$ elements (structured grid case) for the eliminated p-multigrid and traditional p-multigrid.

Next, we calculate and compare the reduction factor of the relative residual

$$\rho = \frac{\|b - Au\|^{i+1}}{\|b - Au\|^{i}}, \tag{31}$$

where i is the V-cycle iteration. We calculate the reduction factor using the relative residual from the final two V-cycles, avoiding the initial rapid reduction from the first V-cycle. The reduction factors for the eliminated

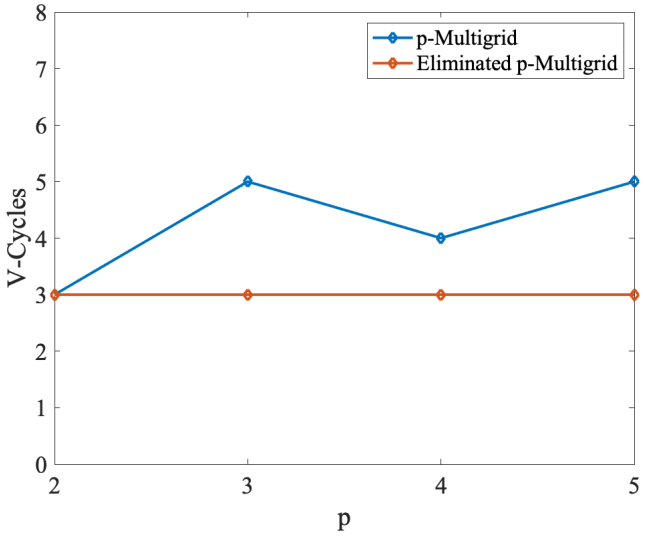


Figure 7: Number of V-cycles until convergence for $p \in [2, 3, 4, 5]$ and $N = 441$ elements (structured grid case) for the eliminated p-multigrid and traditional p-multigrid.

and traditional p-multigrid for each polynomial order are shown in Table 4. The eliminated p-multigrid is shown to have a significantly better reduction factor, at a minimum twice as effective and at a maximum an order of magnitude more effective.

Table 4: Residual reduction factor per polynomial order p (structured grid case) for the eliminated p-multigrid and traditional p-multigrid.

Polynomial Order	ρ	$\rho_{\text{Eliminated}}$
2	0.026	0.010
3	0.111	0.026
4	0.036	0.010
5	0.099	0.017

5.3. Unstructured Grid

Next, we perform the same test case for an unstructured grid (Fig. 8) with a very similar number of elements $N = 468$. We test polynomial orders ranging from $p \in [2, 3, 4, 5]$. The relative residual with respect to the V-cycles for both p-multigrid methods with $p = 5$ is shown in Fig. 9. The eliminated p-multigrid converges in exactly half the number of V-cycles required for the traditional p-multigrid. Fig. 10 shows that the eliminated p-multigrid consistently outperforms the traditional multigrid. Especially for $p > 3$, requiring consistently approximately half the number of V-cycles to converge.

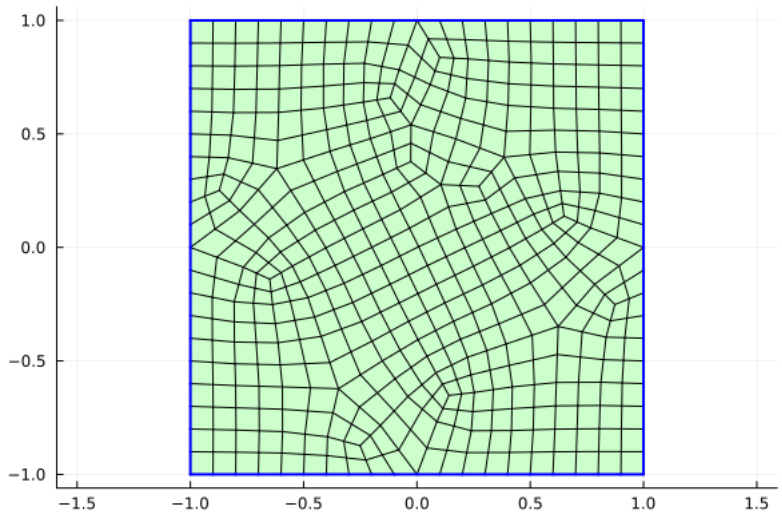


Figure 8: Unstructured mesh of square domain with $N = 468$ quadrilateral elements.

The reduction factor for the relative residual ρ was calculated for each case and is shown in Table 5. The eliminated p-multigrid shows a significantly better reduction factor, with an order of magnitude improvement over traditional p-multigrid for $p > 3$.

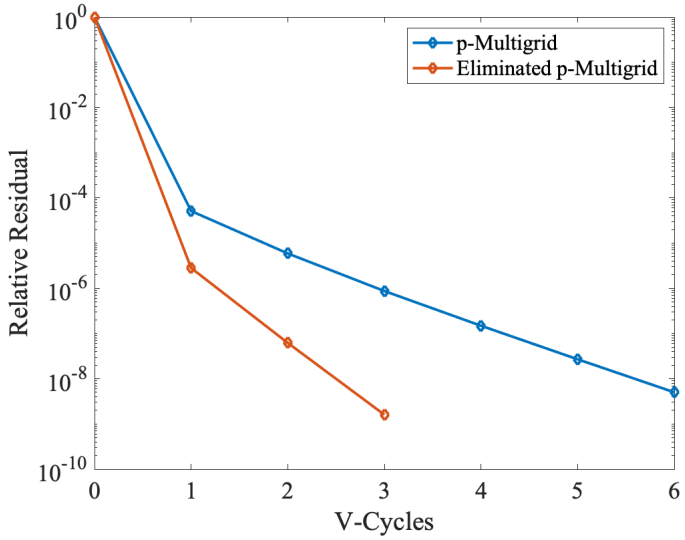


Figure 9: Relative residual vs V-cycles for $p = 5$ and $N = 468$ elements (unstructured grid case) for the eliminated p-multigrid and traditional p-multigrid.

Table 5: Residual reduction factor per polynomial order p (Unstructured Grid) for the eliminated p-multigrid and traditional p-multigrid.

Polynomial Order	ρ	$\rho_{\text{Eliminated}}$
2	0.050	0.013
3	0.158	0.029
4	0.100	0.017
5	0.186	0.026

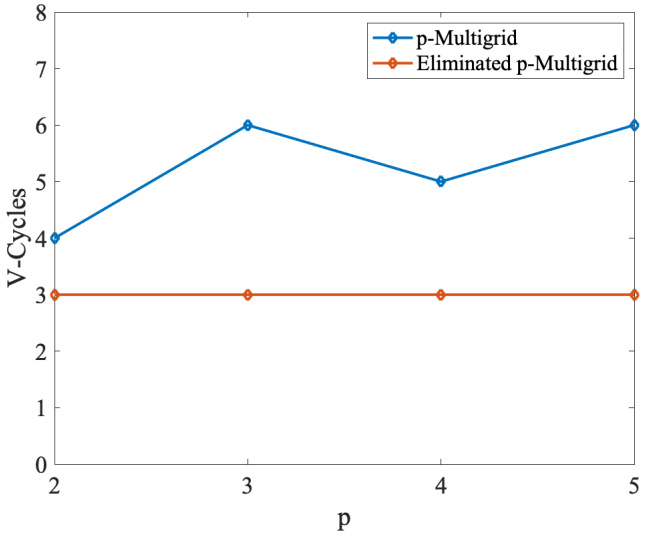


Figure 10: Number of V-cycles until convergence for $p \in [2, 3, 4, 5]$ and $N = 468$ elements (unstructured grid case) for the eliminated p-multigrid and traditional p-multigrid.

5.4. Incompressible Navier-Stokes

For the final numerical experiment, we apply our eliminated p-multigrid to the unsteady non-dimensional incompressible Navier-Stokes equations,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla P + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \boldsymbol{f},$$
$$\nabla \cdot \boldsymbol{u} = 0,$$

where $\boldsymbol{u}$ and P are the non-dimensionalized velocity vector and pressure, $\boldsymbol{f}$ is the body force vector and Re is the Reynolds number. To solve the governing equations, we use the fractional step method in [19, 20]. The procedure is as follows. The diffusive terms are treated with the Crank-Nicolson method and the convective terms are treated with the Adams-Bashforth 2 method.

We first obtain the intermediate velocity field by solving Eq. (32) for $\boldsymbol{u}^*$.

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -((\boldsymbol{u} \cdot \nabla)\boldsymbol{u})^{n+\frac{1}{2}} + \frac{1}{2\mathrm{Re}}\nabla^2(\boldsymbol{u}^* + \boldsymbol{u}^n) + \boldsymbol{f}. \tag{32}$$

Next we solve the elliptic problem and perform the projection step

$$\nabla^2\phi^{n+1} = \frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^* \tag{33}$$

$$\boldsymbol{u}^{\boldsymbol{n+1}} = \boldsymbol{u}^* - \Delta t \nabla \phi^{n+1}. \tag{34}$$

To update the pressure, we use the following expression where $P^{n+\frac{1}{2}}$ is the time centered pressure

$$(M + \frac{\Delta t}{2\mathrm{Re}}\nabla^2)P^{n+\frac{1}{2}} = \phi^{n+1}. \tag{35}$$

We apply our eliminated p-multigrid for a single timestep to solve for both the intermediate velocity in Eq. (32) and the elliptic problem in Eq. (33). We also compare to the traditional p-Multigrid.

The test case we apply our method to is the flow over a NACA 0012 airfoil at an angle of attack of $\alpha = 30^\circ$ with a Reynolds number of $\mathrm{Re} = 1000$, which was previously run in [21]. We use an unstructured grid with $N = 1008$ elements (Fig. 11) and a polynomial order of $p = 4$. The left, bottom and top boundaries have inlet boundary conditions, the right boundary has outlet boundary conditions and the airfoil has wall boundary conditions imposed. We use a timestep of $\Delta t = 2 \times 10^{-5}$. For the sake of a visual description of the test case, the flow field at $t = 3$ is shown in Fig. 12 and Fig. 13.

Both p-multigrid methods solved for the intermediate velocity using one V-cycle. The comparison of the relative residual is shown in Fig. 14. We observe that the eliminated p-multigrid converges in 20% fewer V-cycles compared to the traditional p-multigrid. While the gain is not as drastic as the

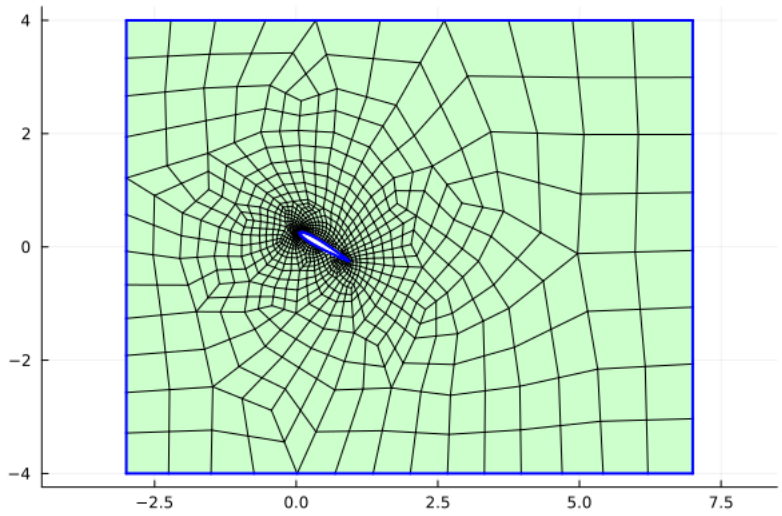


Figure 11: Unstructured mesh for flow over a NACA 0012 airfoil at Re = 1000 and $\alpha = 30°$. Domain discretized with $N = 1008$ quadrilateral elements and polynomial order of $p = 4$.

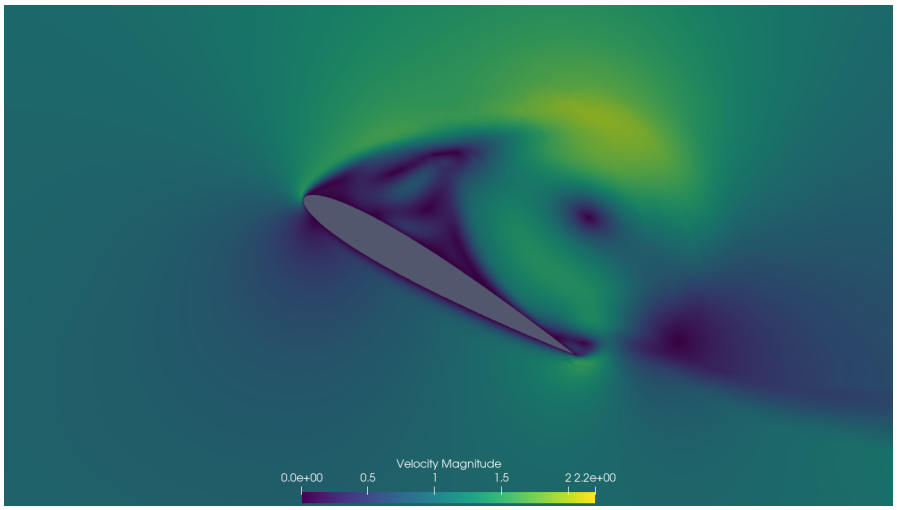


Figure 12: Velocity magnitude at $t = 3$ for flow over a NACA 0012 airfoil at Re = 1000 and $\alpha = 30°$.

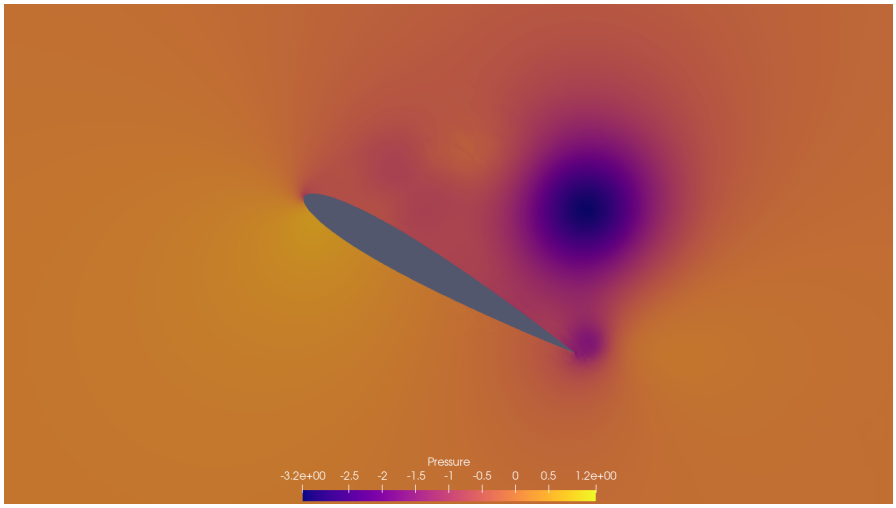


Figure 13: Pressure at $t = 3$ for flow over a NACA 0012 airfoil at $\mathrm{Re} = 1000$ and $\alpha = 30°$.

previous cases, there is also the benefit that in addition to fewer V-cycles, each V-cycle is operating on a smaller linear system.

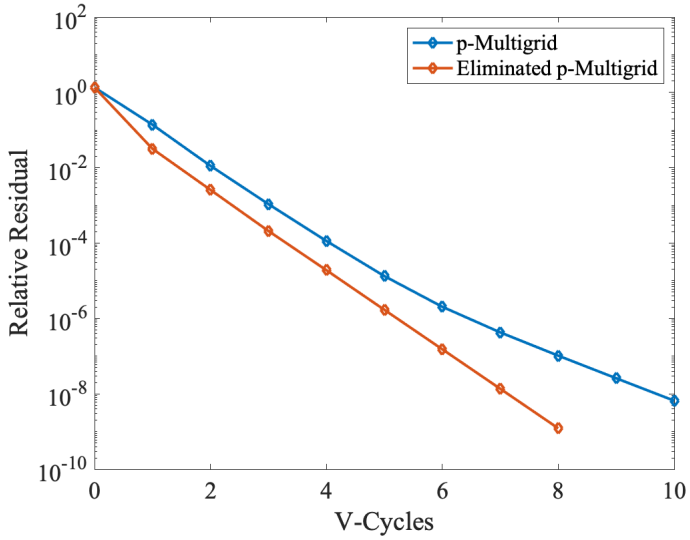


Figure 14: Relative Residual vs V-Cycles for the flow over a NACA 0012 airfoil elliptic problem Eq. (33), unstructured grid with $p = 4$ and $N = 1008$ elements.

We also compute the reduction factors and compare in Table 6. We observe that the eliminated p-multigrid has a notably smaller reduction factor compared to the traditional p-multigrid.

Table 6: Comparison of Reduction Factor for Eliminated vs Non-Eliminated p-Multigrid for solving Pressure Poisson Equation

Method	ρ
Eliminated	0.0894
Non-Eliminated	0.2540

6. Conclusion

In this work, we developed a new linear solver approach specifically for LDG viscous discretizations where p-multigrid is used on an eliminated system via static condensation. We eliminate degrees of freedom based on the LDG switch function. This approach allows for operation on drastically reduced linear systems while maintaining optimal sparsity. We applied this method to two-dimensional HCDG discretizations for Poisson's equation and incompressible flow cases with unstructured quadrilateral grids. We showed that as the eliminated procedure is applied to higher polynomial order discretizations, larger reductions are obtain in terms of degrees of freedom and number of non-zero entries. This is favorable for the use of very high polynomial orders $p > 5$. Even for moderate polynomial orders eg: $p = 2$, a significant reduction is still obtained, approximately a factor of two. We compared our eliminated p-multigrid to a traditional p-multigrid for structured and unstructured grids. We observed essentially a factor of two reduc-

tion in the number of V-cycles required for convergence. The reduction in V-cycles coupled with operating on a reduced linear system is very promising for mitigating the cost of high order methods. We also successfully applied our method to an incompressible flow problem and showed a reduction in the number of V-cycles required compared to traditional p-multigrid.

For future work, we will test our method on DG discretizations with closed nodes such as Gauss-Lobatto nodes as they are very commonly used in DG implementations. Furthermore, we will explore using our method on compact discontinuous Galerkin discretizations since it extends from LDG [22]. Other future work includes using our method on simplex elements and three-dimensional elements with both closed and half-closed nodes. We have used our method as a linear solver, however, we will investigate the properties when used as a preconditioner for Krylov methods. Finally, since we obtain large reductions in the linear system, we will apply our method on fully implicit DG discretizations.

References

[1] J. S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods, volume 54 of *Texts in Applied Mathematics*, Springer, New York, 2008. URL: https://doi.org/10.1007/978-0-387-72067-8. doi:10.1007/978-0-387-72067-8, Algorithms, Analysis, and Applications.

[2] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (2001) 173–261. URL: https://doi.org/10.1023/A:1012873910884. doi:10.1023/A:1012873910884.

[3] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Mathematics of Computation 31 (1977) 333–390. URL: https://www.ams.org/mcom/1977-31-138/S0025-5718-1977-0431719-X/. doi:10.1090/S0025-5718-1977-0431719-X.

[4] A. Jameson, D. Mavriplis, Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh, AIAA Journal 24 (1986) 611–618. URL: https://arc.aiaa.org/doi/10.2514/3.9315. doi:10.2514/3.9315.

[5] D. J. Mavriplis, A. Jameson, Multigrid solution of the Navier-Stokes equations on triangular meshes, AIAA Journal 28 (1990) 1415–1425. URL: https://arc.aiaa.org/doi/10.2514/3.25233. doi:10.2514/3.25233.

[6] M. Hortmann, M. Perić, G. Scheuerer, Finite volume multigrid prediction of laminar natural convection: Bench-mark solutions, International Journal for Numerical Methods in Fluids 11 (1990) 189–207. URL: https://onlinelibrary.wiley.com/doi/10.1002/fld.1650110206. doi:10.1002/fld.1650110206.

[7] Y. Pan, P.-O. Persson, Agglomeration-based geometric multigrid solvers for compact discontinuous Galerkin discretizations on unstructured meshes, Journal of Computational Physics 449 (2022) 110775. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999121006707. doi:10.1016/j.jcp.2021.110775.

[8] C. Klaij, M. Van Raalte, H. Van Der Ven, J. Van Der Vegt,

h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, Journal of Computational Physics 227 (2007) 1024–1045. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999107003762. doi:10.1016/j.jcp.2007.08.034.

[9] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, Journal of Computational Physics 207 (2005) 92–113. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999105000185. doi:10.1016/j.jcp.2005.01.005.

[10] H. Luo, J. D. Baum, R. Löhner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, Journal of Computational Physics 211 (2006) 767–783. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999105003128. doi:10.1016/j.jcp.2005.06.019.

[11] C. R. Nastase, D. J. Mavriplis, High-order discontinuous Galerkin methods using an hp-multigrid approach, Journal of Computational Physics 213 (2006) 330–357. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999105003839. doi:10.1016/j.jcp.2005.08.022.

[12] K. Shahbazi, D. J. Mavriplis, N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, Journal

of Computational Physics 228 (2009) 7917–7940. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999109003878. doi:10.1016/j.jcp.2009.07.013.

[13] D. Fortunato, C. H. Rycroft, R. Saye, Efficient Operator-Coarsening Multigrid Schemes for Local Discontinuous Galerkin Methods, SIAM Journal on Scientific Computing 41 (2019) A3913–A3937. URL: https://epubs.siam.org/doi/10.1137/18M1206357. doi:10.1137/18M1206357.

[14] R. J. Guyan, Reduction of stiffness and mass matrices, AIAA Journal 3 (1965) 380–380. URL: https://arc.aiaa.org/doi/10.2514/3.2874. doi:10.2514/3.2874.

[15] A. M. Rueda-Ramírez, E. Ferrer, D. A. Kopriva, G. Rubio, E. Valero, A statically condensed discontinuous Galerkin spectral element method on Gauss-Lobatto nodes for the compressible Navier-Stokes equations, Journal of Computational Physics 426 (2021) 109953. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999120307270. doi:10.1016/j.jcp.2020.109953.

[16] B. Cockburn, C.-W. Shu, The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems, SIAM Journal on Numerical Analysis 35 (1998) 2440–2463. URL: http://epubs.siam.org/doi/10.1137/S0036142997316712. doi:10.1137/S0036142997316712.

[17] L. Haupt, J. Stiller, W. Nagel, A fast spectral element solver combining static condensation and multigrid techniques, Jour-

nal of Computational Physics 255 (2013) 384–395. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999113005251. doi:10.1016/j.jcp.2013.07.035.

[18] Y. Pan, P.-O. Persson, Half-closed discontinuous Galerkin discretisations, Journal of Computational Physics 524 (2025) 113731. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999125000142. doi:10.1016/j.jcp.2025.113731.

[19] J. Perot, An Analysis of the Fractional Step Method, Journal of Computational Physics 108 (1993) 51–58. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999183711629. doi:10.1006/jcph.1993.1162.

[20] D. L. Brown, R. Cortez, M. L. Minion, Accurate Projection Methods for the Incompressible Navier–Stokes Equations, Journal of Computational Physics 168 (2001) 464–499. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999101967154. doi:10.1006/jcph.2001.6715.

[21] A. Nayak, Y. Pan, P.-O. Persson, Half-Closed Discontinuous Galerkin Methods For Incompressible Flow Problems, in: AIAA SCITECH 2025 Forum, American Institute of Aeronautics and Astronautics, Orlando, FL, 2025. URL: https://arc.aiaa.org/doi/10.2514/6.2025-1297. doi:10.2514/6.2025-1297.

[22] J. Peraire, P.-O. Persson, The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems, SIAM Journal on Scientific Computing

30 (2008) 1806–1824. URL: http://epubs.siam.org/doi/10.1137/070685518. doi:10.1137/070685518.

Chapter 6

Conclusions

In this thesis, several different approaches were taken to push the discontinuous Galerkin spectral element method (DGSEM) to model complex geometries in more efficient ways than currently available. Research was conducted on both underlying discretizations and linear solvers. The first two works model acoustic wave propagation problems while the final two works model incompressible fluid flow. Each work produced its own conclusions, which will now be discussed.

In the first chapter, a transfinite mapping was used to extend an existing dynamic load balanced hp-adaptive DGSEM acoustic wave equation solver which could only support Cartesian grids. Both strong and weak scaling tests were performed to observe the effect of the curvilinear geometry interface on the existing code. The weak scaling tests show that the code scales very similarly to the Cartesian grid code, however there is a slight computational overhead incurred by the transfinite mapping. Unfortunately, the strong scaling tests show the code does not scale very well compared to the Cartesian grid code. After observing the results, we decided to explore the immersed boundary method to model complex geometries as an alternative since only Cartesian grids are required.

In the second paper, the immersed boundary method (IBM) was implemented in the hp-adaptive DGSEM acoustic wave equation code to model complex geometries. The advantage of using the IBM is that it drastically simplifies the meshing process for mod-

eling complex geometries, as only simple Cartesian grids are required. We utilized the volume penalty method to model walls as porous obstacles. While it is possible to bypass a complicated meshing process through transfinite mapping or unstructured grids, the IBM does incur a loss of accuracy and produces oscillatory behavior. This issue was addressed by utilizing hp-adaptivity and using a low porosity parameter. Accuracy was improved and oscillations were dampened by the use of hp-adaptivity. Furthermore, it was possible to model complex geometries with small length scales, which would prove to be very difficult using transfinite mapping. Overall, the IBM method showed promising results. The implications of this work is that it lowers the barrier to entry for engineers to use high order DGSEM methods. Furthermore, given our hp-adaptive approach, an engineer could use this method to determine a preliminary starting grid to know what portions of the domain require more refinement. The engineer could then use this grid either with the IBM or use it as a prototype for traditional meshing techniques.

In the third paper, unstructured grids were used to model complex geometries for incompressible flow, however the focus was on the underlying discretization scheme to improve on efficiency. The HCDG method was used to model the incompressible Navier-Stokes equations. The HCDG method provides the advantages of good sparsity patterns and increased efficiency in DG operator construction. Using half-closed nodes produces better sparsity patterns compared to using open nodes for first and second order operators and identical sparsity patterns to closed nodes for second order operators. Using Gauss-Radau nodes also provides an extra degree of quadrature accuracy compared to using the commonly used closed nodes Gauss-Lobatto nodes. An increase in solution accuracy was also observed when using HCDG. The results are promising, but more work will be conducted to further investigate the properties of using these nodes.

In the final paper, a new linear solver is developed and is applied to the HCDG method since it uses the local discontinuous Galerkin (LDG) [27] viscous discretization.

This linear solver uses p-multigrid operating on a system eliminated by static condensation [28]. The static condensation procedure is specified by the LDG discretization. Drastic reductions in degrees of freedom are obtained using the elimination procedure, making this method well suited for very high-order discretizations. Remarkably, a reduction in V-cycles up to a factor of two required for convergence for elliptic problems was observed as well. This method was also applied to the incompressible Navier-Stokes equations where a 20% reduction in V-cycles was also observed. The compounding effect of operating on a reduced degree of freedom system and a reduction of V-cycles is very encouraging for addressing the issue of computational cost for high-order DGSEM.

All of the research conducted in this thesis works towards making DGSEM more practical for industrial use, either by aiding the modeling of more complex geometries or by improving the underlying cost. Ideally, with further research and development, high-order DGSEM will become popular for industrial CFD. This would improve the modeling of turbulent flows which would have impacts in the aerospace, energy, marine, and weather prediction sectors. Additionally, improvements to high-order DGSEM can be beneficial to academia, as they can provide a more accurate framework for studying the fundamental turbulent flow physics of high-speed flows. Furthermore, by improving the underlying numerics for high-order DGSEM, other high-order methods can also see improvement by applying and extending the improved approaches accordingly. This would hopefully lead to better scientific computing beyond just CFD, potentially impacting fields such as computational chemistry and computational finance since these fields also numerically solve PDEs. It should also be mentioned, higher order simulations will also produce more accurate data for use in data driven methods such as reduced-order modeling (ROMs) or machine learning (ML). If the data is not of sufficient quality, the data driven methods will not be effective in real world scenarios. Thus, it is crucial to ensure the production of high quality data as society increasingly utilizes data driven technologies.

Despite the progress made and the promising potential, there are still improvements

that can be made through future research. The future directions of all the works will now be discussed.

Future Works

Each research work in this thesis has possible future directions to improve the methods or to simulate more realistic scenarios. For the transfinite mapping of the Cartesian hp-adaptive DGSEM acoustic code, further research would be to improve scaling performance by investigating alternative software implementations for the transfinite mapping, which are better suited to parallel computing. Since this code is designed to solve hyperbolic conservation laws, this code can be extended to model the two-dimensional compressible Euler equations.

For the IBM research, one direction is to perform scaling tests using the IBM implementation and compare to the transfinite mapped code to obtain a formal measure of the performance gains. As mentioned previously in the context of this code, another future direction is to model the two-dimensional compressible Euler equations. However, in this case, the penalization procedure would have to be heavily modified as in this work the penalization procedure was designed specifically for acoustic problems. Another direction is to model a PDE with viscosity. Since the current research was done without viscosity, adding viscosity should further aid in reducing oscillations. Furthermore, extending the code to three-dimensional problems allows for more realistic simulations.

For the HCDG research, further investigation will be conducted on the properties of using Gauss-Radau nodes. For example, we will investigate the stability of using Gauss-Radau nodes for LES simulations. The superconvergence properties of Gauss-Radau nodes will also be investigated. Simulating the three-dimensional compressible Navier-Stokes equations is of interest since DG formulations are well suited for modeling compressible flows and it allows for more realistic aeronautics simulations. Implementing hp-adaptivity is desired, given the local nature of HCDG and the observed benefits

from the other works in this thesis.

For the eliminated p-multigrid research, further investigation will include applying the method on closed nodes and different element types (three-dimensional quadrilaterals, two- and three-dimensional simplexes) since these other node types and element types are also commonly used in DG. Implementing the method in a production code and performing wall-clock time tests will provide a more realistic measure of the gains from the eliminated p-multigrid. To ensure the eliminated p-multigrid is widely applicable, this method will be tested on fully implicit DG formulations. Additionally, testing the use of this method as a preconditioner for Krylov methods such as GMRES can also be explored.

Bibliography

[1] A. Bondeson, T. Rylander, and P. Ingelstrom, *Computational Electromagnetics*. Texts in Applied Mathematics, New York, N.Y: Springer, 2005.

[2] A. Curnier, *Computational Methods in Solid Mechanics*, vol. 29 of *Solid Mechanics and Its Applications*. Dordrecht: Springer Netherlands, 1994.

[3] H. Lomax, T. H. Pulliam, and D. W. Zingg, *Fundamentals of Computational Fluid Dynamics*. Scientific Computation, Berlin Heidelberg: Springer, corr. 2. print ed., 2003.

[4] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, "OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, vol. 59, p. 1075–1104, Apr. 2019.

[5] J. P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis, "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," *NASA CR-2014-218178, Langley Research Center*, Mar. 2014.

[6] F. F. Buscariolo, J. Hoessler, D. Moxey, A. Jassim, K. Gouder, J. Basley, Y. Murai, G. R. Assi, and S. J. Sherwin, "Spectral/hp element simulation of flow past a Formula One front wing: Validation against experiments," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 221, p. 104832, Feb. 2022.

[7] H. Pitsch and H. Steiner, "Large-eddy simulation of a turbulent piloted methane/air diffusion flame (Sandia flame D)," *Physics of Fluids*, vol. 12, p. 2541–2554, Oct. 2000.

[8] A. Mansi and D. Aydin, "The impact of trailing edge flap on the aerodynamic performance of small-scale horizontal axis wind turbine," *Energy Conversion and Management*, vol. 256, p. 115396, Mar. 2022.

[9] Y.-C. Pan, H.-X. Zhang, and Q.-D. Zhou, "Numerical Prediction of Submarine Hydrodynamic Coefficients using CFD Simulation," *Journal of Hydrodynamics*, vol. 24, p. 840–847, Dec. 2012.

[10] M. Vadsola, *High-Order Spectral Element Method Simulation of Flow Past a 30P30N Three-Element High Lift Wing*. Master's thesis, University of Ottawa, Ottawa, Canada, Sept. 2020.

[11] Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P. Persson, B. Van Leer, and M. Visbal, "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, vol. 72, p. 811–845, July 2013.

[12] A. T. Patera, "A spectral element method for fluid dynamics: Laminar flow in a channel expansion," *Journal of Computational Physics*, vol. 54, p. 468–488, June 1984.

[13] S. K. Lele, "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics*, vol. 103, p. 16–42, Nov. 1992.

[14] Z. Wang, "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids. Basic Formulation," *Journal of Computational Physics*, vol. 178, p. 210–251, May 2002.

[15] O. V. Vasilyev, S. Paolucci, and M. Sen, "A Multilevel Wavelet Collocation Method for Solving Partial Differential Equations in a Finite Domain," *Journal of Computational Physics*, vol. 120, p. 33–47, Aug. 1995.

[16] B. Cockburn and C.-W. Shu, "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *J. Sci. Comput.*, vol. 16, no. 3, pp. 173–261, 2001.

[17] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods*, vol. 54 of *Texts in Applied Mathematics*. Springer, New York, 2008. Algorithms, Analysis, and Applications.

[18] N. Chaderjian, "Improved Resolution of Rotorcraft Flows Using Adaptive Mesh Refinement - (NASA@SC15) url: https://www.nas.nasa.gov/sc15/demos/demo5.html."

[19] D. A. Kopriva and J. H. Kolias, "A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows," *Journal of Computational Physics*, vol. 125, p. 244–261, Apr. 1996.

[20] S. He, *Dynamic Load Balancing for a hp-adaptive Discontinuous Galerkin Wave Equation Solver via Spacing-Filling Curve and Advanced Data Structure*. Master's thesis, University of Ottawa, Ottawa, Canada, 2021.

[21] C. S. Peskin, "Flow patterns around heart valves: A numerical method," *Journal of Computational Physics*, vol. 10, p. 252–271, Oct. 1972.

[22] Y. Pan and P.-O. Persson, "Half-closed discontinuous Galerkin discretisations," *Journal of Computational Physics*, vol. 524, p. 113731, Mar. 2025.

[23] D. A. Kopriva, *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Scientific Computation, Dordrecht: Springer Netherlands, 2009.

[24] C. Mavriplis, "Adaptive mesh strategies for the spectral element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 116, p. 77–86, Jan. 1994.

[25] Y. Maday, C. Mavriplis, and A. Patera, "Nonconforming mortar element methods: Application to spectral discretizations," in *T.F. Chan, R. Glowinski, J. Periaux, O.B. Widlund (Eds.)*, Domain Decomposition Methods, p. 392–418, SIAM Philadelphia, 1989.

[26] G. Tousignant, *A Graphics Processing Unit Based Discontinuous Galerkin Wave Equation Solver with hp-Adaptivity and Load Balancing*. Master's thesis, University of Ottawa, Ottawa, Canada, 2022.

[27] B. Cockburn and C.-W. Shu, "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM Journal on Numerical Analysis*, vol. 35, p. 2440–2463, Dec. 1998.

[28] R. J. Guyan, "Reduction of stiffness and mass matrices," *AIAA Journal*, vol. 3, p. 380–380, Feb. 1965.