Distributed Implementation and Verification of Hybridizable Discontinuous Galerkin Methods for Nonhydrostatic Ocean Processes

Corbin Foucart, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA, USA
pierrel@mit.edu

***Abstract*—Nonhydrostatic, multiscale processes are an important part of our understanding of ocean dynamics. However, resolving these dynamics with traditional computational techniques can often be prohibitively expensive. We apply the hybridizable discontinuous Galerkin (HDG) finite element methodology to perform computationally efficient, high-order, nonhydrostatic ocean modeling by solving the Navier-Stokes equations with the Boussinesq approximation. In this work, we introduce a distributed implementation of our HDG projection method algorithm. We provide numerical experiments to verify our methodology using the method of manufactured solutions and provide preliminary benchmarking for our distributed implementation that highlight the advantages of the HDG methodology in the context of distributed computing. Lastly, we present simulations in which we capture nonhydrostatic internal waves that form as a result of tidal interactions with ocean topography. First, we consider the case of tidally-driven oscillatory flow over an abrupt, shallow seamount, and next, the case of strongly-stratified, oscillatory flow over a tall seamount. We analyze and compare our simulations to other results in literature.**

***Index Terms*—ocean modeling, ocean dynamics, distributed computing, finite element, nonhydrostatic**

I. Introduction

Accurate numerical simulation and modeling of ocean physics is becoming increasingly important in scientific applications spanning many scales and disciplines, from path planning to global climate science. In particular, there is a need to capture nonhydrostatic, nonlinear ocean dynamics. Two such nonhydrostatic phenomena are observed when tidally-driven flow encounters complex, highly variable, and often, very steep bottom topography: internal wave rays and nonlinear internal solitary waves [1]. Such processes have been observed close to undersea ridges in the Luzon Strait and South China Sea [2]. Furthermore, models capable of capturing these interactions can be extended to simulate other situations arising in the ocean when flows encounter abrupt topography, such as that occurring at the continental shelfbreaks, or to study biological processes in highly productive regions such as Stellwagen Bank in Massachusetts [3]–[5]. As such, there is a demand for high-order methodologies that can accurately model such flows with large gradients.

In recent years, Discontinuous Galerkin (DG) finite element methods have become increasingly popular for computational fluid dynamics applications; DG schemes are capable of achieving high-order accuracy on multi-resolution unstructured meshes, resolving sharp gradients in advection-dominated flows, and are well-suited to modern computer architectures. Despite these advantages, the spatially duplicated degrees of freedom requisite to these schemes can become prohibitively expensive. The development of Hybridizable Discontinuous Galerkin (HDG) methods was motivated by a desire to improve the computational efficiency of DG schemes while admitting solutions in discontinuous finite element spaces. However, for large-scale computations, hybridization alone is often insufficient to overcome memory and time-to-solution limitations. Our present research motivation is to apply the HDG methodology to ocean problems in a distributed and computationally tractable manner.

Our work focuses on HDG discretizations of projection methods used for numerical solution of the incompressible Navier-Stokes (INS) equations. For the HDG methodology regarding linear convection-diffusion problems, we refer to [6]. For parallelization of HDG methods in the context of the nonlinear equations of compressible flow, we refer to [7]. For matrix-free methodologies in a Continuous Galerkin finite element context, we refer to [8]. For an alterative HDG discretization of the nonlinear, fully-coupled INS equations, see [9].

This paper is organized as follows. In Sec. II, we briefly describe the system of governing equations we wish to solve, and outline the scheme described in [5]. In Sec. III, we extend the algorithms to a distributed, parallel context. Next, in Sec. IV, we apply the results to a variety of numerical experiments, providing a preliminary investigation into the scalability of our distributed algorithms, and presenting simulations capturing nonhydrostatic ocean processes and dynamics. Finally, some concluding remarks are made and possible future research directions are discussed in Sec. V.

978-1-5386-4814-8/18/$31.00 ©2018 IEEE

II. Methodology

A. Governing equations

Consider the non-dimensionalized unsteady incompressible Navier-Stokes equations on a simply connected domain Ω within a finite time interval $[0, T]$:

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v} + \nabla p &= -\nabla \cdot (\mathbf{v} \otimes \mathbf{v}) + \mathbf{f} \quad \text{in } \Omega \times [0,T], \\ \nabla \cdot \mathbf{v} &= 0 \quad \text{in } \Omega \times [0,T], \\ \mathbf{v}|_{\partial\Omega} &= \mathbf{g}_D \quad \text{in } \partial\Omega \times [0,T], \\ \mathbf{v}|_{t=0} &= \mathbf{v}_0 \quad \text{in } \Omega, \end{aligned} \tag{1}$$

where $\mathbf{v} = [u, v, w]$ is the velocity, $p = \frac{1}{\rho_0} P$, P is the pressure, $\mathbf{f} = \mathbf{g}\rho$ when there is only density forcing, ρ_0 is the mean density, ρ is the density perturbation, $\mathbf{g}_D$ is some Dirichlet velocity boundary condition, and $\mathbf{v}_0$ is the velocity initial condition. For the Boussinesq equations with density forcing, the tracer equation for the density is:

$$\begin{aligned} \frac{\partial \rho}{\partial t} - \nabla \frac{1}{\mathrm{ReSc}} \cdot \nabla \rho &= -\nabla \cdot \mathbf{v}\rho \quad \text{in } \Omega \times [0,T], \\ \rho|_{\partial\Omega} &= g_{D_\rho} \quad \text{in } \partial\Omega \times [0,T], \\ \rho|_{t=0} &= \rho_i \quad \text{in } \Omega, \end{aligned} \tag{2}$$

where $\mathrm{Sc} = \frac{\nu}{\kappa}$ is the Schmidt number, g_{D_ρ}, ρ_i are Dirichlet boundary conditions and initial condition for ρ, respectively. We treat the non-linear term explicitly, and group it with the forcing term, yielding the Stokes-like system we wish to solve,

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v} + \nabla p &= \mathbf{F}_{\partial t} \quad \text{in } \Omega \times [0,T], \\ \nabla \cdot \mathbf{v} &= 0 \quad \text{in } \Omega \times [0,T], \\ \mathbf{v}|_{\partial\Omega} &= \mathbf{g}_D \quad \text{in } \partial\Omega \times [0,T], \\ \mathbf{v}|_{t=0} &= \mathbf{v}_0 \quad \text{in } \Omega, \end{aligned} \tag{3}$$

where $\mathbf{F}_{\partial t} = -\nabla \cdot \mathbf{v}\mathbf{v} + \mathbf{f}$.

B. Projection methods

The time-discretization of these equations (3) using the rotational incremental pressure correction scheme [10], [11] is discussed here. The time-split equations first determine the predictor velocity $\bar{\mathbf{v}}^{k+1}$ using p^k.

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \bar{\mathbf{v}}^{k+1} + \nabla p^k = \mathbf{F}^{k,\,k+1}, \tag{4}$$

$$\bar{\mathbf{v}}|_{\partial\Omega}^{k+1} = \mathbf{g}_D, \tag{5}$$

$$\mathbf{v}|_{t=0} = \mathbf{v}_0, \tag{6}$$

$$p|_{t=0} = p_0, \tag{7}$$

where a is some constant associated with the time-integration method, and $\mathbf{F}^{k,\,k+1}$ contains the explicitly calculated terms and the right-hand-side forcing. Next, a Poisson equation is solved for the pressure corrector δp^{k+1}:

$$-\nabla^2 \delta p^{k+1} = -\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \tag{8}$$

$$\left.\frac{\partial \delta p^{k+1}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega} = 0. \tag{9}$$

Finally, the velocities and pressure are corrected:

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \nabla \delta p^{k+1}, \tag{10}$$

$$p^{k+1} = p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}. \tag{11}$$

C. Equivalent first-order problem

Since HDG is a mixed finite element method, we introduce the new variables $\mathbf{Q} = \frac{1}{\mathrm{Re}} \nabla \mathbf{v}$ and $\mathbf{q}_{\delta p} = \nabla \delta p$. We obtain an equivalent first-order problem:

$$\mathrm{Re}\bar{\mathbf{Q}}^{k+1} - \nabla \bar{\mathbf{v}}^{k+1} = 0, \tag{12}$$

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla \cdot \bar{\mathbf{Q}}^{k+1} + \nabla p^k = \mathbf{F}^{k,k+1}, \tag{13}$$

(8) as

$$\mathbf{q}_{\delta p}^{k+1} - \nabla \delta p^{k+1} = 0, \tag{14}$$

$$-\nabla \cdot \mathbf{q}_{\delta p}^{k+1} = -\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \tag{15}$$

and (10) as

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}^{k+1}, \tag{16}$$

$$\tag{17}$$

while (11) remains the same

$$p^{k+1} = p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}.$$

These are the time-discretized split equations, and the starting point of the HDG weak formulation.

D. Notation

In order to state the weak form of the problem, we will first introduce some requisite notation [5]. We let $\mathcal{T}_h = \cup K_i$ be a finite collection of non-overlapping elements, K_i, that discretizes the entire computational domain Ω (Fig. 1). Also, let $\partial\mathcal{T}_h = \{\partial K : K \in \mathcal{T}_h\}$ be the set of interfaces of all elements, where ∂K is the boundary of element K. For two elements sharing an edge K^+ and K^-, we define $e = \partial K^+ \cap \partial K^-$ as the edge between elements K^+ and K^-. The edges can be classified as ε° and ε^∂, the set of interior and boundary edges, respectively, with $\varepsilon = \varepsilon^\circ \cup \varepsilon^\partial$. K^+ and K^- have outward pointing normals $\hat{\mathbf{n}}^+$ and $\hat{\mathbf{n}}^-$, respectively. The quantities $[\mathbf{a}^\pm, c^\pm]$ denote the traces of $[\mathbf{a}, c]$ on the edge e from the interior of $K^\pm$. The "mean" value $\{\{\bullet\}\}$ and "jumps" $[\![\bullet]\!]$ on the interior interfaces $e \in \varepsilon^\circ$ for scalar and vector quantities are then defined as

$$\{\{\mathbf{a}\}\} = (\mathbf{a}^+ + \mathbf{a}^-)/2, \qquad \{\{c\}\} = (c^+ + c^-)/2,$$
$$[\![\mathbf{a} \cdot \hat{\mathbf{n}}]\!] = \mathbf{a}^+ \cdot \hat{\mathbf{n}}^+ + \mathbf{a}^- \cdot \hat{\mathbf{n}}^-, \quad [\![c\hat{\mathbf{n}}]\!] = c^+ \hat{\mathbf{n}}^+ + c^- \hat{\mathbf{n}}^-.$$

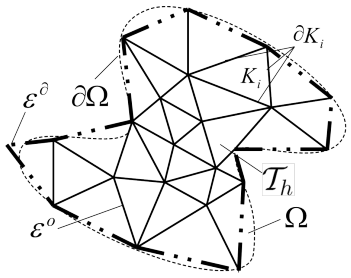

Fig. 1. Notation for domain discretization, [5].

On the set of boundary interfaces $e \in \varepsilon^\partial$, (with outward facing normal $\hat{\mathbf{n}}$ on $\partial\Omega$), we define these mean and jump quantities the usual way:

$$\{\{\mathbf{a}\}\} = \mathbf{a}, \qquad \{\{c\}\} = c,$$
$$[\![\mathbf{a}\cdot\hat{\mathbf{n}}]\!] = \mathbf{a}\cdot\hat{\mathbf{n}}, \qquad [\![c\hat{\mathbf{n}}]\!] = c\hat{\mathbf{n}}.$$

Let $\mathcal{P}^{\mathrm{p}}(D)$ denote the set of polynomials of maximum degree p existing on a domain D. We introduce the discontinuous finite element spaces:

$$\left\{\theta \in L^2(\Omega) : \theta\,|_K \in \mathcal{P}^{\mathrm{p}}(K), \forall K \in \mathcal{T}_h\right\}$$
$$\left\{\boldsymbol{\theta} \in (L^2(\Omega))^d : \boldsymbol{\theta}\,|_K \in (\mathcal{P}^{\mathrm{p}}(K))^d, \forall K \in \mathcal{T}_h\right\}$$
$$\left\{\boldsymbol{\Theta} \in (L^2(\Omega))^{d\times d} : \boldsymbol{\Theta}\,|_K \in (\mathcal{P}^{\mathrm{p}}(K))^{d\times d}, \forall K \in \mathcal{T}_h\right\},$$

In addition, we introduce the traced finite element spaces existing on the unique interfaces ε

$$\left\{\theta_\varepsilon \in L^2(\Omega) : \theta_\varepsilon\,|_e \in \mathcal{P}^{\mathrm{p}}(e), \forall e \in \varepsilon\right\},$$
$$\left\{\boldsymbol{\theta}_\varepsilon \in (L^2(\Omega))^d : \boldsymbol{\theta}_\varepsilon\,|_e \in (\mathcal{P}^{\mathrm{p}}(e))^d, \forall e \in \varepsilon\right\},$$
$$\left\{\boldsymbol{\Theta}_\varepsilon \in (L^2(\Omega))^{d\times d} : \boldsymbol{\Theta}_\varepsilon\,|_e \in (\mathcal{P}^{\mathrm{p}}(e))^{d\times d}, \forall e \in \varepsilon\right\}.$$

We also set $\{\theta_\varepsilon = \mathsf{P}\mathbf{g}_D \text{ on } \partial\Omega\}$, where P is the L^2 projection of the boundary condition $\mathbf{g}_D$ into the same space as θ_ε. Note that θ_ε is continuous on the interface, e, shared by K^+ and K^-, but discontinuous at the borders between different interfaces.

Lastly, we define the inner products over continuous domains $D \in \mathbb{R}^d$ and $\partial D \in \mathbb{R}^{d-1}$ as

$$(\mathbf{a},\mathbf{b})_D = \int_D \mathbf{a}\cdot\mathbf{b}\,\mathrm{d}D \qquad (c,d)_D = \int_D cd\,\mathrm{d}D \tag{18}$$
$$\langle\mathbf{a},\mathbf{b}\rangle_{\partial D} = \int_{\partial D} \mathbf{a}\cdot\mathbf{b}\,\mathrm{d}\partial D \quad \langle c,d\rangle_{\partial D} = \int_{\partial D} cd\,\mathrm{d}\partial D \tag{19}$$

We define the additional inner product on the discontinuous edge space:

$$\langle\mathbf{a},\boldsymbol{\theta}_\varepsilon\rangle_\varepsilon = \sum_{e\in\varepsilon}\langle\mathbf{a},\boldsymbol{\theta}_\varepsilon\rangle_e \qquad \langle c,\theta_\varepsilon\rangle_\varepsilon = \sum_{e\in\varepsilon}\langle c,\theta_\varepsilon\rangle_e \tag{20}$$

for vector or scalar functions $\mathbf{a}$, c defined on ε.

E. HDG weak formulation

We provide an outline of the weak form of the spatial discretization and refer the interested reader to [5], in which the derivations and choices of stability parameters are discussed in detail.

The weak form of the velocity predictor equations (12), (13) are listed in Fig. 2. Similarly, the corresponding pressure corrector equations (14), (15) are given in Fig. 3. Finally, the velocity and pressure updates (16), (18) are given in Fig. 4. We choose $\tau = 1$ and $\tau_p = \frac{1}{a\tau\Delta t}$.

element-local equations:

$$\left(\mathrm{Re}\bar{\mathbf{Q}}^{k+1},\boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1},\boldsymbol{\Theta}\right)_K + \left\langle\bar{\mathbf{v}}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} = \left\langle\bar{\boldsymbol{\lambda}}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K}$$
$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1},\boldsymbol{\theta}\right)_K + \left\langle\tau\bar{\mathbf{v}}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\tau\bar{\boldsymbol{\lambda}}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} - \left(\nabla p^k,\boldsymbol{\theta}\right)_K + \left(\mathbf{F}^{k,k+1},\boldsymbol{\theta}\right)_K$$

global flux conservation equations:

$$\left\langle\left[\!\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}} - \tau\left(\bar{\mathbf{v}}^{k+1} - \bar{\boldsymbol{\lambda}}^{k+1}\right)\right]\!\right],\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon = \left\langle[\![p^k\hat{\mathbf{n}}]\!],\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon + \left\langle\mathbf{g}_N,\boldsymbol{\theta}_\varepsilon\right\rangle_{\varepsilon_N^\partial}$$
$$\bar{\boldsymbol{\lambda}}|_{\varepsilon_D^\partial}^{k+1} = \mathsf{P}\mathbf{g}_D$$

flux definitions:

$$\hat{\bar{\mathbf{v}}}^{k+1} = \begin{cases}\bar{\boldsymbol{\lambda}}^{k+1}, & \text{on } \varepsilon^o\cup\varepsilon_N^\partial,\\ \mathsf{P}\mathbf{g}_D, & \text{on } \varepsilon^\partial\end{cases}, \qquad \hat{\bar{\mathbf{Q}}}^{k+1} - \hat{p}^k\mathbf{I} = \bar{\mathbf{Q}}^{k+1} - p^k\mathbf{I} - \tau\left(\bar{\mathbf{v}}^{k+1} - \hat{\bar{\mathbf{v}}}^{k+1}\right)\hat{\mathbf{n}}$$

Fig. 2. Velocity predictor $\bar{\mathbf{v}}^{k+1}$, weak form and flux definitions

element-local equations:

$$\left(\mathbf{q}_{\delta p}^{k+1},\boldsymbol{\theta}\right)_K - \left(\nabla\delta p^{k+1},\boldsymbol{\theta}\right)_K + \left\langle\delta p^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\lambda_{\delta p}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K}$$
$$-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1},\theta\right)_K + \left\langle\tau_p\delta p^{k+1},\theta\right\rangle_{\partial K} = \left\langle\tau_p\lambda_{\delta p}^{k+1},\theta\right\rangle_{\partial K} - \left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t},\theta\right)_K - \left\langle\frac{(\hat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K}$$

global flux conservation equations:

$$\left\langle\left[\!\left[\mathbf{q}_{\delta p}{}^{k+1}\cdot\hat{\mathbf{n}} - \tau_p\left(\delta p^{k+1} - \lambda_{\delta p}^{k+1}\right)\right]\!\right],\theta_\varepsilon\right\rangle_\varepsilon = \left\langle g_{N_p},\theta_\varepsilon\right\rangle_\varepsilon$$
$$\lambda_{\delta p}|_{\varepsilon_D^\partial}^{k+1} = \mathsf{P}g_{D_p}$$

flux definitions:

$$\hat{\delta p}^{k+1} = \begin{cases}\lambda_{\delta p}^{k+1}, & \text{on } \varepsilon^o\cup\varepsilon_N^\partial\\ \mathsf{P}g_{D_p}, & \text{on } \varepsilon^\partial\end{cases}, \qquad \widehat{\mathbf{q}_{\delta p}}^{k+1} = \mathbf{q}_{\delta p}{}^{k+1} - \tau_p\left(\delta p^{k+1} - \hat{\delta p}^{k+1}\right)\hat{\mathbf{n}}$$

Fig. 3. Pressure corrector δp^{k+1}, weak form and flux definitions

element-local corrections:

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}$$
$$\left(p^{k+1},\theta\right)_K = \left(p^k + \delta p^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla\cdot\bar{\mathbf{v}}^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left\langle(\hat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}.$$

edge-space correction:

$$\boldsymbol{\lambda}^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1} - a\Delta t\widehat{\mathbf{q}_{\delta p}}^{k+1}$$

Fig. 4. Velocity and pressure corrections

We remark that each of the global flux conservation equations ensures that the normal component of the flux is single-valued on the element edge space.

F. Sequential implementation

The weak form outlined in Sec. II-E and in [5] contains two spatial HDG solves, namely, the velocity predictor $\bar{\mathbf{v}}$ and the pressure corrector δp. Although the element-local equations differ, the solution procedure is the same for each, and can be described abstractly in terms of an unknown quantity u, its gradient $\boldsymbol{q}$, and the global flux variable λ. Algorithm 1 details the sequential, matrix-based solution procedure.

We remark that the globally coupled unknowns have support on the element interfaces only. The computation of the elemental contributions can be computed independently of the other elements. Similarly, the reconstruction of the solution on each element requires only the global solution λ applied

Algorithm 1 HDG: Sequential, matrix-based procedure

1: **for** $K \in \mathcal{T}_h$
2: A^K, $b^K \leftarrow$ Elemental contributions from $\boldsymbol{q}^K, u^K, \lambda^K$
3: **end for**
4: A, $\boldsymbol{b} \leftarrow$ assemble A^K, b^K for all $K \in \mathcal{T}_h$
5: $\boldsymbol{\lambda} \leftarrow$ solve $A\boldsymbol{\lambda} = \boldsymbol{b}$
6: **for** $K \in \mathcal{T}_h$
7: $\boldsymbol{q}^K$, $u^K \leftarrow$ Reconstruct elemental solution with λ^K
8: **end for**

as boundary conditions and hence can also be computed independently of the other elements.

III. Efficiency considerations

A. Matrix-free iterative solvers

Matrix-free iterative solvers arise when the explicit assembly of the linear system A (line 4 of Alg. 1) becomes impractical– either due to memory limitations, or in the context of a parallel implementation where A is distributed among several processors without shared memory. In either case, a matrix-free solver does not assemble the complete coefficient matrix, but in instead applies the action of A on the solution vector $\boldsymbol{\lambda}_i$ in the context of an iterative method such as a conjugate gradient (CG) or Generalized Minimal RESidual (GMRES) method.

In the case of distributed problems, the elemental contribution arrays A^K can be explicitly formed and stored. Alternatively, to avoid storage of elemental matrix data, the contribution matrices can be computed "on the fly" and applied to the solution vector at each iteration of the linear solve. Since the HDG methodology requires either the solution of a dense linear system or an explicit matrix inversion in order to compute the elemental contributions [6] (instead of Jacobian computations, as discussed in [8]), in this work, we consider only the former.

B. Distributed and parallel implementation

A typical distributed computing cluster often consists of multiple compute nodes with non-shared memory and network communication between nodes. Therefore our methodology uses a multiple-instruction, multiple data (MIMD) technique to achieve parallelism. Namely, the computational domain is partitioned between N processors $P_0, \ldots, P_{N-1}$ before simulation runtime, at which point the elements in the computational domain $K \in \mathcal{T}_h$ as well as the edges $\partial K \in \varepsilon$ are assigned to processors, and mapping data containing the global and process-local numberings are saved. At runtime, each processor shares no memory with the other processors; for data to be shared, it is explicitly passed from one processor to another using the Message Passing Interface (MPI).

In order to solve the problem arising from the HDG discretization in a distributed manner, we aim to partition both the interior and edge degrees of freedom in a load-balanced manner. We use METIS [12] to perform the decomposition of the mesh elements into partitions based on the number of available processors. However, the globally-coupled unknowns are those on the edges between elements. It is therefore crucial that the edges be assigned to processors in a load-balanced manner. We do so using a greedy algorithm detailed in Alg. 2. Although the algorithm must loop through the "MPI border" edges (edges for which $K^{\pm}$ are owned by distinct processes) twice, this step need only be done once upon domain partitioning, and has the advantage of compensating for edge imbalances that can occur as a result of the fact that METIS partitions only the *elements* $K \in \mathcal{T}_h$. Indeed, for the use cases in this work, we see excellent load balancing in terms of edge computations (see Fig. 6).

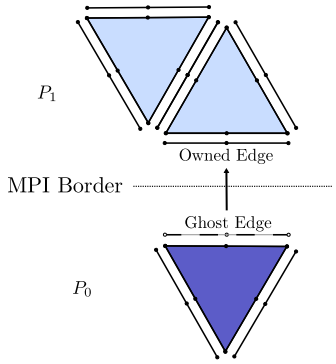

Fig. 5. Message passing between processors. The edge between process 0 and process 1 is owned by process 1. Process 0 passes data from the corresponding ghost edge to the owning process.

Algorithm 2 Interior edge partitioning

1: **for** $\partial K \in \{\varepsilon^\circ\}$
2: **if** K^+, K^- on same process i **then**
3: process $i \leftarrow \partial K$
4: **end if**
5: **end for**
6: **for** $\partial K \in \{\varepsilon^\circ\}$ unassigned
7: K^+ on process i, K^- on process j
8: process with fewer edges $\leftarrow \partial K$
9: **end for**

Although each MPI border edge is assigned to an owning process, the non-owning process contains a ghost edge corresponding to the same edge, as depicted in Fig. 5. Contributions from the non-owning processor are passed via MPI from the ghost edge to the owned edge, and such that the elemental contributions to the global linear system are computed and stored on the owning processor.

We conclude the description of our distributed algorithm with some brief remarks pursuant to Algorithm 3, which provides an implementation of a distributed HDG solve. Line 3, in which the elemental contributions are computed, requires only element data from the mesh local to process n, due to the discontinuous nature of the polynomial spaces in which the solution is sought, and hence requires no communication

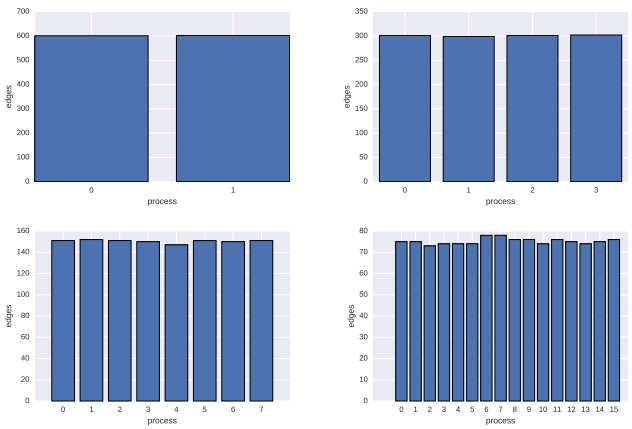

Fig. 6. Edge distribution for unstructured mesh with 1200 edges for 2, 4, 8, 16, processors [top to bottom, left to right].

Algorithm 3 HDG: Distributed, matrix-free procedure

1: **local processor** n
2: **for** $K \in \mathcal{T}_h^n$
3: A^K, $b^K \leftarrow$ Elemental contributions from $\boldsymbol{q}^K, u^K, \lambda^K$
4: **send/recv** border data A^K_{mpi}, b^K_{mpi} to owning processor
5: **end for**
6: A, $\boldsymbol{b} \leftarrow A^K$, b^K $\forall K \in \mathcal{T}_h^n$, forming distributed $A, \boldsymbol{b}$
7: $\boldsymbol{\lambda} \leftarrow$ distributed solve $A\boldsymbol{\lambda} = \boldsymbol{b}$
8: **for** $K \in \mathcal{T}_h^n$
9: $\boldsymbol{q}^K$, $u^K \leftarrow$ Reconstruct elemental solution with λ^K
10: **end for**

between processors. On the other hand, forming the distributed matrix-free operator A does require passing elemental contributions from ghost edges to owned edges, necessitating the message passing in line 4. Furthermore, although the advection term is computed explicitly, the weak formulation in [ref discretization] contains an edge integration term which may require remote data to compute the flux terms $\hat{\phi}\hat{\mathbf{v}}$ (see Eq. (21)), where we denote a generic unknown with ϕ. Therefore, the right hand side contributions b^K_{mpi} also span processes.

$$\boldsymbol{F}_K^{k,k+1} = -\underbrace{\left(\nabla\cdot(\phi\mathbf{v}),\,\theta\right)_K}_{\text{interior}} + \underbrace{\left\langle(\hat{\phi}\hat{\mathbf{v}}-\phi\mathbf{v})\cdot\hat{\mathbf{n}},\,\theta\right\rangle_{\partial K}}_{\text{edge (MPI)}} \quad (21)$$

The distributed solve referenced in line 7 refers to an iterative solution procedure such as CG or GMRES. Each iteration of the iterative method involves distributed matrix-vector products, and since the operator A is distributed between processors, each iteration of the solution λ_i must update the current solution between processors as well. Each processor holds part of the complete solution vector as well as data corresponding to ghost edges as depicted in Fig. 7.

IV. Numerical Experiments

A. Verification with manufactured solution

We consider a steady 3D advection-diffusion problem on $\Omega = (-2,\,2)\times(-2,\,2)\times(-2,\,2)$,

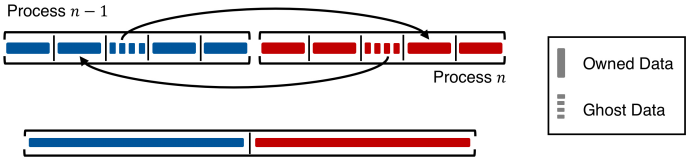

Fig. 7. Schematic of distributed solution vector $\boldsymbol{\lambda}$

$$\begin{aligned} -\nabla\cdot(\kappa\nabla\phi) + \nabla\cdot(\mathbf{v}\phi) &= 0 \quad \text{on} \quad \Omega, \\ \phi|_{\partial\Omega_D} &= g_D, \\ \phi|_{\partial\Omega_N} &= g_N \end{aligned} \quad (22)$$

with velocity field $\mathbf{v} = [u, v, w]$ such that

$$\begin{aligned} u &= 2c\sin\left(\frac{\pi}{2}(x+1)\right)\left(\cos\left(\frac{\pi}{2}(y+1)\right) - \cos(\pi z)\right) \\ v &= 2c\sin\left(\frac{\pi}{2}(y+1)\right)\left(\cos(\pi z) - \cos\left(\frac{\pi}{2}(y+1)\right)\right) \\ w &= c\sin(\pi z)\left(\cos\left(\frac{\pi}{2}(x+1)\right) - \cos\left(\frac{\pi}{2}(y+1)\right)\right) \end{aligned} \quad (23)$$

and with $c = 0.1$, The source term f and mixed Dirichlet and Neumann boundary conditions g_D, g_N are chosen such that we have the exact solution as follows:

$$\phi = \sin\left(\frac{\pi}{2}(x+1)\right)\cos\left(\frac{\pi}{2}(y+1)\right)\cos\left(\frac{\pi}{2}(z+2)\right) \quad (24)$$

We present the error and order of convergence in the L^2-norm in Figure 8 and Table I, respectively. Both ϕ and $\mathbf{q}$ (table not shown) converge at the optimal rate [13] of $p+1$.

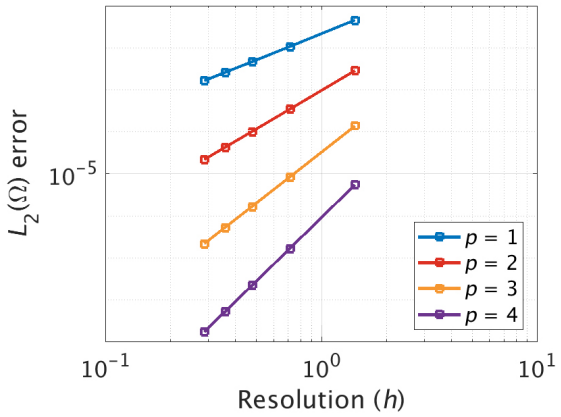

Fig. 8. Convergence of ϕ, measured in the L^2 norm

B. Preliminary scalability investigation

We verify our numerical software with a manufactured solution, and perform a preliminary investigation into scalability of the algorithm with a swirl test case similar to that described in [14]. We consider a non-dimensional, time-dependent, 2D

TABLE I
HISTORY OF CONVERGENCE ORDER IN THE L^2 NORM FOR THE ADVECTION-DIFFUSION TEST CASE

Mesh	**Degree** (p)			
N	1	2	3	4
4	-	-	-	-
8	2.098	3.049	4.027	5.038
12	2.065	3.034	4.014	5.015
16	2.042	3.021	4.008	5.005
32	2.029	3.014	4.005	4.972

advection-diffusion problem on $\Omega = (-1,\,1) \times (-1,\,1)$,

$$\begin{aligned} \frac{\partial \phi}{\partial t} - \nabla \cdot (\kappa \nabla \phi) &= -\nabla \cdot (\mathbf{v}\phi) \quad \text{in} \quad \Omega \times [0,T], \\ \phi|_{\partial\Omega} &= g_D \quad \text{in} \quad \partial\Omega \times [0,T], \\ \phi|_{t=0} &= \phi_0 \quad \text{in} \quad \Omega, \end{aligned} \tag{25}$$

where diffusion constant κ is small, on the order of 10^{-4}, and with the alternating swirl velocity field

$$\begin{aligned} u &= \sin\left(\frac{\pi}{5}t\right)\left[\frac{1}{2}\sin(2\pi y)\sin^2(\pi x)\right] \\ v &= -\sin\left(\frac{\pi}{5}t\right)\left[\frac{1}{2}\sin(2\pi x)\sin^2(\pi y)\right] \end{aligned} \tag{26}$$

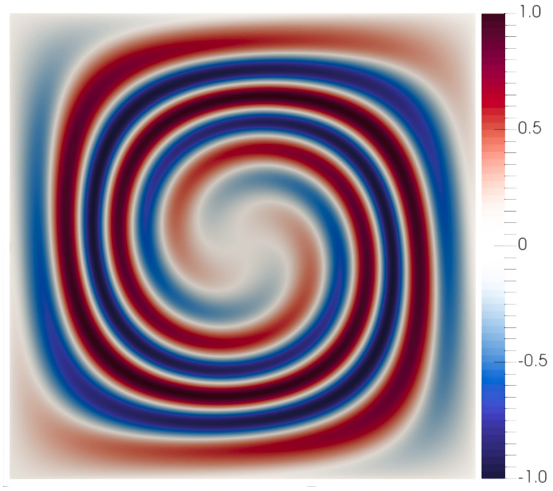

Fig. 9. Swirl flow test case

Since each step of the projection method is a Poisson-like system, and message passing is required for the explicit computation of the advection term (fluxes), this problem is a good prototype for a scalability investigation.

The results show near-perfect strong scaling in the computation of the element local contributions and reconstruction, as well as the speedups in conjugate gradient iteration of the global linear solve. for the cases of 2, 4, 8, and 24 processors, compared to a single processor. We can see that for the case of 24 processors, the message passing overhead provides substantially worse performance for small problems, but achieves the best speedup as problem size grows.

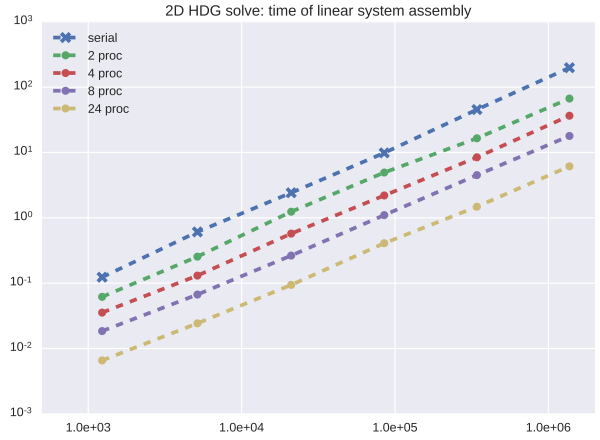

Fig. 10. Average wallclock time of system assembly (s) vs. problem size (edge space degrees of freedom) for different numbers of processors.

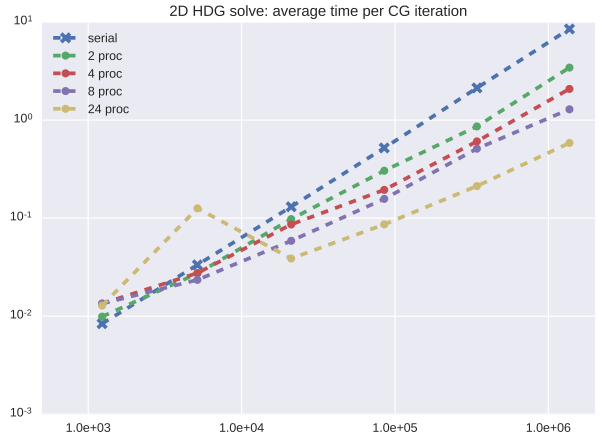

Fig. 11. Average wallclock time per conjugate gradient iteration (s) vs. problem size (edge space degrees of freedom) for different numbers of processors.

C. Internal wave formation

Our next numerical experiment is a benchmark test case used in Vitousek and Fringer [1] and is set-up here to demonstrate the ability of our high-order HDG code to simulate non-hydrostatic physics. In this example, tidally-driven, oscillatory flow with variable density encounters a single isolated, shallow, but abrupt seamount; the surrounding sea bed is flat, and the height of the seamount is only 2% of the total depth.

The domain considered here is $[-L, L] \times [-H, 0]$, where $L = 1500$ m and $H = 1000 - 20\exp(-x^2/1800)$ m; that is, the seamount is represented as a Gaussian sill. As in Vitousek and Fringer [1], the viscosity $\nu = 10^{-6}$ m^2 s^{-1}, the mean

density $\rho_0 = 1000\ \mathrm{kg\,m^{-3}}$, and the Brunt–Väisälä frequency $N = 0.007\ \mathrm{s^{-1}}$. The Coriolis force is neglected here. The flow starts from rest, with a linear stratification given by $\frac{\partial \rho'}{\partial z} = -0.005\ \mathrm{kg\,m^{-4}}$. A no-slip boundary condition is imposed on the velocity along the sea bed, and a free-slip condition is imposed at the surface. On the left and right (open) boundaries, oscillatory flow is forced: $u_D = 0.01 \sin(\omega t)\ \mathrm{m\,s^{-1}}$, where $\omega = 0.0056\ \mathrm{s^{-1}}$. Homogeneous Neumann BCs are imposed on the density perturbation everywhere. The domain is discretized into 788 elements in the x-direction and 414 in the z-direction, and second-degree polynomials are used to represent all variables (that is, the scheme is third-order accurate in space). A first-order IMEX-RK time stepping scheme is used, with 500 time steps per tidal cycle.

Fig. 12 shows the vertical velocity after 17.75 tidal cycles. Non-hydrostatic internal wave rays have fully developed, emanating from the seamount, reflecting off the surface and bottom until being damped at the open boundaries by our numerical sponge layer. The sign of w changes with the tide (not shown). For our choice of tidal and buoyancy frequency, the theoretical non-hydrostatic IW ray angle [1], [15] with respect to a flat bottom is given by

$$\theta_{\mathrm{nhs}} = \tan^{-1}\left(\sqrt{\omega^2/(N^2-\omega^2)}\right) \approx 53.1^\circ,$$

whereas the theoretical hydrostatic IW ray angle is

$$\theta_{\mathrm{hs}} = \tan^{-1}(\omega/N) \approx 38.7^\circ.$$

These angles are shown in fig. 12 by green and red lines for the non-hydrostatic and hydrostatic cases, respectively. Our results are in very good agreement with the theoretical non-hydrostatic ray angle, and are also in agreement with the results presented in previous studies [1].

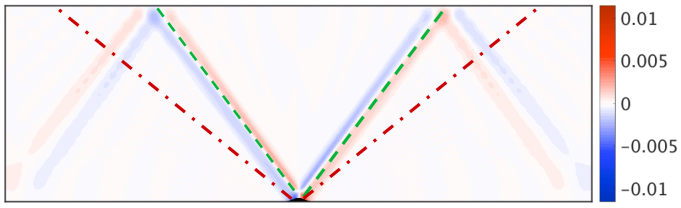

Fig. 12. Vertical velocity ($\mathrm{m\,s^{-1}}$) at $t = 17.75$ tidal cycles, along with theoretical non-hydrostatic (green) and hydrostatic (red) beam angles. The flow is from right to left.

D. Internal solitary waves

In this experiment, (non-hydrostatic) internal solitary wave trains are simulated, which are generated when a strongly-stratified, oscillatory flow encounters and passes over a tall seamount. Such non-hydrostatic processes are of great interest to oceanographers, and have been observed close to undersea ridges in the Luzon Strait and the South China Sea [2]. Unlike the previous example, the height of the seamount here is 87% of the total depth.

Once again, the domain considered is $[-L, L]\times[-H, 0]$, but $L = 2\times10^5$ m and $H = 3000-2600\exp(-x^2/2.88\times10^8)$. To better emulate realistic ocean conditions, anisotropic diffusion is used, with $\nu_x = 100\ \mathrm{m^2\,s^{-1}}$ and $\nu_z = 10^{-4}\ \mathrm{m^2\,s^{-1}}$, while the mean density is taken as $\rho_0 = 1024.75\ \mathrm{kg\,m^{-3}}$. The flow again starts from rest, but is nonlinearly stratified according to the profile given in Vitousek and Fringer [1]. The boundary conditions are the same as in sec. IV-C, except that $u_D = 0.134 \sin(\omega t)\ \mathrm{m\,s^{-1}}$ on the open boundaries, where $\omega = 1.41\times10^{-4}\ \mathrm{s^{-1}}$ (this is the approximate M2 tidal frequency). The domain is discretized into 218 elements in x-direction and 100 in the z-direction; this grid is four times coarser in the x-direction than that used in Vitousek and Fringer [1]. However, second-degree polynomial approximations to our solution are used, effectively doubling our resolution in both the x- and z-directions while retaining low computational cost. A first-order IMEX-RK time stepping scheme is again used, with approximately 8912 time steps per tidal cycle.

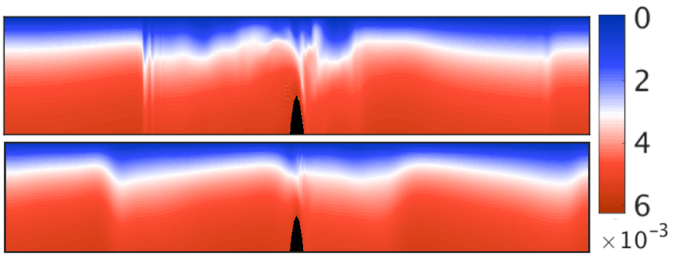

Fig. 13. Normalized density perturbation ρ'/ρ_0 after 1.25 tide cycles. The ISW train is captured (top) when a 3rd order accurate HDG scheme is used, but is not resolved (bottom) when a 2nd order accurate HDG scheme is used. Only the top 20% of the domain is shown.

The resulting normalized density perturbation ρ'/ρ_0 after 1.25 tide cycles is shown in fig. 13. Consider first the high-order HDG case (top of fig. 13). After about one tide cycle, a leftward-propagating nonlinear internal solitary wave forms, developing into a wave train by 1.25 cycles. These wave trains propagate away from the seamount, until they are damped out when they enter our numerical sponge layer imposed near the open boundaries. Additional solitons and wave trains develop every half-cycle, and propagate in alternating directions according to the phase of the tide. This result is in qualitative agreement with the expected non-hydrostatic behavior. Crucially, however, the ISW train is captured only when the higher-order HDG scheme is used. In the low-order HDG case, a leftward-propagating bore forms after 1.25 cycles and does not develop into a wave train; additional bores form every half-cycle thereafter. This behavior is very similar to that predicted by hydrostatic models [1]. In short, the result demonstrates the necessity of using higher-order HDG methods to capture non-hydrostatic dynamics and other complex ocean processes.

V. Conclusion

We provided a preliminary distributed implementation of our HDG projection method algorithm. We discussed strategies for partitioning the element edge space in a load balanced

manner and for handling message passing between different processors with no shared memory. We verified our software with a manufactured solution and obtained the optimal order of convergence without post-processing. We benchmarked our parallel algorithm and demonstrated the advantages of the HDG algorithm due to the steps of the algorithm that are embarassingly parallelizable. Furthermore, we benchmarked the parallel performance of the globally coupled portion of the HDG algorithm, and demonstrated the speedups therein. Finally, we provided simulations of ocean flows encountering steep topography which capture nonhydrostatic internal wave effects.

Multiple extensions of the research presented in this work are possible. Multi-threading on each processor could yield improved time-to-solution [8] and an added layer of parallelism. Element-local computations could be efficiently sped up with use of graphics processors [16]. Alternatively, instead of traditional computing clusters, a cloud-based computing system could better fit scientists' needs for large-scale, massively parallel ocean simulations.

ACKNOWLEDGMENTS

We are grateful to the members of our MSEAS group for useful discussions, as well as to Arkopal Dutt and Jing Lin for advice and feedback. We are grateful to the Office of Naval Research for support under grant N00014-15-1-2626 (FLEAT), and the National Oceanographic Partnership Program (NOPP) for support under grant N00014-15-1-2597 (Seamless Multiscale Forecasting), each to the Massachusetts Institute of Technology.

REFERENCES

[1] S. Vitousek and O. B. Fringer, "A nonhydrostatic, isopycnal-coordinate ocean model for internal waves," *Ocean Modelling*, vol. 83, pp. 118–144, 2014.

[2] M. Buijsman, Y. Kanarska, and J. McWilliams, "On the generation and evolution of nonlinear internal waves in the South China Sea," *Journal of Geophysical Research: Oceans*, vol. 115, no. C2, 2010.

[3] M. P. Ueckermann and P. F. J. Lermusiaux, "High order schemes for 2D unsteady biogeochemical ocean models," *Ocean Dynamics*, vol. 60, no. 6, pp. 1415–1445, Dec. 2010.

[4] M. P. Ueckermann, "High order hybrid discontinuous Galerkin regional ocean modeling," PhD thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, MA, Feb. 2014.

[5] M. P. Ueckermann and P. F. J. Lermusiaux, "Hybridizable discontinuous Galerkin projection methods for Navier–Stokes and Boussinesq equations," *Journal of Computational Physics*, vol. 306, pp. 390–421, 2016.

[6] N. C. Nguyen, J. Peraire, and B. Cockburn, "An implicit high-order hybridizable discontinuous galerkin method for linear convection–diffusion equations," *Journal of Computational Physics*, vol. 228, no. 9, pp. 3232–3254, 2009.

[7] X. Roca, C. Nguyen, and J. Peraire, "Scalable parallelization of the hybridized discontinuous galerkin method for compressible flow," in *21st AIAA Computational Fluid Dynamics Conference*, 2013, p. 2939.

[8] M. Kronbichler and K. Kormann, "A generic interface for parallel cell-based finite element operator application," *Computers & Fluids*, vol. 63, pp. 135–147, 2012.

[9] N. C. Nguyen, J. Peraire, and B. Cockburn, "An implicit high-order hybridizable discontinuous galerkin method for the incompressible navier–stokes equations," *Journal of Computational Physics*, vol. 230, no. 4, pp. 1147–1170, 2011.

[10] L. Timmermans, P. Minev, and F. Van De Vosse, "An approximate projection scheme for incompressible flow using spectral elements," *International journal for numerical methods in fluids*, vol. 22, no. 7, pp. 673–688, 1996.

[11] J. Aoussou, J. Lin, and P. F. J. Lermusiaux, "Iterated pressure-correction projection methods for the unsteady incompressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 373, pp. 940–974, Nov. 2018.

[12] G. Karypis and V. Kumar, "Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0," 1995.

[13] B. Cockburn, J. Gopalakrishnan, and F.-J. Sayas, "A projection-based error analysis of hdg methods," *Mathematics of Computation*, vol. 79, no. 271, pp. 1351–1367, 2010.

[14] D. R. Durran, *Numerical methods for wave equations in geophysical fluid dynamics*. Springer Science & Business Media, 2013, vol. 32.

[15] P. Kundu and L. Cohen, "Fluid mechanics, 638 pp," *Academic, Calif*, 1990.

[16] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, "Nodal discontinuous galerkin methods on graphics processors," *Journal of Computational Physics*, vol. 228, no. 21, pp. 7863–7882, 2009.