An Anisotropic hp-Adaptation Framework for Ultraweak Discontinuous Petrov–Galerkin Formulations

Ankit Chakraborty[a], Stefan Henneking[a], Leszek Demkowicz[a]

[a] *Oden Institute, The University of Texas at Austin*

Abstract

In this article, we present a three-dimensional anisotropic hp-mesh refinement strategy for ultraweak discontinuous Petrov–Galerkin (DPG) formulations with optimal test functions. The refinement strategy utilizes the built-in residual-based error estimator accompanying the DPG discretization. The refinement strategy is a two-step process: (a) use the built-in error estimator to mark and isotropically hp-refine elements of the (coarse) mesh to generate a finer mesh; (b) use the reference solution on the finer mesh to compute optimal h- and p-refinements of the selected elements in the coarse mesh. The process is repeated with coarse and fine mesh being generated in every adaptation cycle, until a prescribed error tolerance is achieved. We demonstrate the performance of the proposed refinement strategy using several numerical examples on hexahedral meshes.
Keywords: DPG, anisotropy, hp-adaptivity

Acknowledgments: We thank Jacob Badger for fruitful discussions. Ankit Chakraborty, Stefan Henneking and Leszek Demkowicz were supported with NSF award 2103524. Ankit Chakraborty is partially supported with the Peter O'Donnell Jr. Postdoctoral Fellowship. All numerical experiments in this article were performed on *Frontera's* Intel Cascade Lake (CLX) nodes located at the Texas Advanced Computing Center [1] using DMS22025 allocation.

1. Introduction

Automatic hp-mesh refinement algorithms are powerful tools that aid finite element discretizations in computing solutions of partial differential equations (PDEs) in an efficient and accurate manner. They achieve this efficiency and accuracy by constructing meshes with optimally distributed element size h and polynomial order of approximation p [2, 3]. Finite element meshes with optimal element size and polynomial distribution are critical for resolving solution features such as boundary layers in convection-dominated diffusion problems

*Corresponding author
Email address: ankit.chakraborty@austin.utexas.edu (Ankit Chakraborty)

or point and edge singularities in problems with re-entrant corners. In such problems, optimal *hp*-meshes are indispensable for achieving exponential convergence [4, 5, 6, 7, 8]. Designing algorithms capable of generating a sequence of optimal *hp*-meshes that deliver optimal convergence rates in a problem-agnostic manner has been a significant challenge in finite element research over the past few decades [6, 9, 10, 11]. Typically, automatic mesh refinement strategies are driven by computable error estimates. These error estimates are computed using the approximate solution delivered by the discretization scheme. Therefore, the accuracy and stability of the underlying numerical discretization are paramount for the effectiveness of the mesh refinement strategy.

The DPG methodology with optimal test functions, first introduced by Demkowicz and Gopalakrishnan in [12, 13, 14], has emerged as a critical technology in terms of robustness and stability over the past decade. Given a stable variational formulation of an underlying PDE and a trial approximation space, the DPG method computes a test space so that the resulting discretization is *inf–sup* stable. The methodology delivers an orthogonal projection in the so-called energy norm. Another significant advantage of the DPG methodology is the presence of a built-in residual-based error estimator, also known as the energy error estimate. This makes the DPG method an ideal candidate for automatic mesh optimization algorithms.

In this article, we focus on the ultraweak (UW) discontinuous Petrov–Galerkin (DPG) finite element formulation with optimal test functions and propose a problem-agnostic anisotropic *hp*-mesh refinement strategy. It is critical to mention that, for the UW DPG method, the energy norm is *equivalent* to the L^2-error [15]. Consequently, the method delivers essentially the L^2-projection of the unknown solution.

The proposed refinement strategy consists of the following steps:

- **Step 1**: Solve the problem on the current *coarse mesh*.
- **Step 2**: Utilize the computed DPG residual to mark coarse mesh elements for refinements.
- **Step 3**: Isotropically *hp*-refine the marked elements to generate a *fine mesh*.
- **Step 4**: Solve the problem on the fine mesh to obtain the fine mesh solution u.
- **Step 5**: Use the fine mesh solution u as a *reference solution* to determine optimal (anisotropic) *hp*-refinements of the selected coarse grid elements.
- **Step 6**: Restore the coarse mesh and execute the optimal *hp*-refinements.

We essentially use the *hp*-algorithm from [2, 3]. Optimizing the mesh in the L^2-space greatly simplifies the original procedure. There is no need for mesh optimization on edges and faces; the *Projection-Based Interpolation* reduces to the L^2-projection performed on elements only. The optimal refinements of a coarse element K are determined by maximizing the rate (e_{hp}) with which the projection error decreases,

$$e_{hp} := \frac{\|u - P_{\text{coarse}}u\|^2 - \|u - P_{\text{opt}}u\|^2}{N_{\text{opt}} - N_{\text{coarse}}}\,.$$

Here, P_{coarse} denotes the L^2-projection onto the coarse mesh, P_{opt} is the projection onto the optimal mesh to be determined, N_{opt} and N_{coarse} denote the number of degrees-of-freedom (dof) of the optimal and coarse grid elements, respectively. As the L^2-projection onto discontinuous polynomial spaces is a purely local operation, the mesh optimization can be trivially performed in parallel.

The article is organized as follows. Section 2 briefly introduces the ultraweak DPG finite element discretization with optimal test functions. Section 3 provides the details of the mesh optimization algorithm. In Section 4, numerical results demonstrate the efficacy of the proposed refinement strategy. Finally, we conclude with a short discussion in Section 5.

2. DPG Methodology

The core idea behind the (ideal) DPG method is to automatically generate a stable discretization for a given well-posed variational formulation and an approximate trial space. The method achieves stability by computing an optimal discrete test space [13] corresponding to the approximate trial space in such a way that the supremum over the continuous test space in the discrete *inf–sup* [16] is automatically attained over the discrete test space. The optimal test space is obtained by inverting the Riesz map corresponding to the test inner product over a *discontinuous or broken*[1] test space. Unfortunately, inverting the Riesz operator exactly is impossible due to the infinite-dimensional nature of the continuous test space. Thus, in practical realizations of DPG methods, we approximate the inverse of the Riesz operator by inverting the Gram matrix induced by the test norm on a larger, but finite-dimensional *enriched* discontinuous test space.[2] The use of broken test spaces enables element-wise inversion of the Gram matrix, but it also introduces trace variables defined on the mesh skeleton [17].

We consider a model Poisson problem. Let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz domain with its boundary, denoted as Γ, is split into two disjoint parts: Γ_u and Γ_σ. The first-order formulation of the Poisson problem is given by:

$$\begin{cases} \boldsymbol{\sigma} - \nabla u = \mathbf{0} & \text{in} \quad \Omega, \\ -\nabla \cdot \boldsymbol{\sigma} = f & \text{in} \quad \Omega, \\ u = u_0 & \text{on} \quad \Gamma_u, \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} = \sigma_0 & \text{on} \quad \Gamma_\sigma, \end{cases}$$

[1] Hence the "D" in the DPG method.

[2] We then refer to it as the *practical* DPG method.

where $f \in L^2(\Omega)$ represents the source term and $\boldsymbol{n}$ denotes the outward normal. Before presenting the ultraweak variational formulation, we briefly introduce the energy spaces used in this article. We define the standard energy spaces as:

$$\begin{aligned} L^2(\Omega) &= \{u : \Omega \to \mathbb{R} : \|u\|_2 < \infty\}\,, \\ H^1(\Omega) &= \left\{v : \Omega \to \mathbb{R} : v \in L^2(\Omega), \nabla v \in \left(L^2(\Omega)\right)^3\right\}, \\ \boldsymbol{H}(\mathrm{div}, \Omega) &= \left\{\boldsymbol{w} : \Omega \to \mathbb{R}^3 : \boldsymbol{w} \in (L^2(\Omega))^3, \nabla \cdot \boldsymbol{w} \in L^2(\Omega)\right\}. \end{aligned} \tag{2.1}$$

In the DPG method, discontinuous energy spaces are used for the test functions. Thus, we must define broken equivalents of $H^1(\Omega)$ and $\boldsymbol{H}(\mathrm{div}, \Omega)$ spaces for the finite element mesh (Ω_h):

$$\begin{aligned} H^1(\Omega_h) &:= \left\{v : \Omega \to \mathbb{R} : v\big|_K \in H^1(K) \quad \forall K \in \Omega_h\right\}, \\ \boldsymbol{H}(\mathrm{div}, \Omega_h) &:= \left\{\boldsymbol{w} : \Omega \to \mathbb{R}^3 : \boldsymbol{w}\big|_K \in \boldsymbol{H}(\mathrm{div}, K) \quad \forall K \in \Omega_h\right\}, \end{aligned} \tag{2.2}$$

where $K \in \Omega_h$ represents an element of the finite element mesh. Use of the broken test spaces [17] leads to the introduction of additional trace unknowns on the mesh skeleton. The traces spaces are defined as:

$$\begin{aligned} H^{1/2}(\Gamma_h) &:= \left\{\hat{u} : \exists\, u \in H^1(\Omega) \,\text{such that}\, \hat{u} = \gamma^K(u\big|_K) \,\text{on}\, \partial K \quad \forall K \in \Omega_h\right\}, \\ H^{-1/2}(\Gamma_h) &:= \left\{\hat{\sigma}_n : \exists\, \boldsymbol{\sigma} \in \boldsymbol{H}(\mathrm{div}, \Omega) \,\text{such that}\, \hat{\sigma}_n = \gamma_n^K(\boldsymbol{\sigma}\big|_K) \,\text{on}\, \partial K \quad \forall K \in \Omega_h\right\}, \end{aligned} \tag{2.3}$$

where γ^K and γ_n^K represent continuous and normal trace operators, respectively [18].

Ultraweak formulation. Let $(U, \hat{U})$ be the approximation trial space, V the test space, and $V^{'}$ the dual space of V. Then, the ultraweak DPG formulation of the Poisson problem can be stated as: Given $l \in V^{'}$, find $\mathfrak{u} \in U$ and $\hat{\mathfrak{u}} \in \hat{U}$ satisfying:

$$b(\mathfrak{u}, \mathfrak{v}) + \hat{b}(\mathfrak{u}, \mathfrak{v}) = l(\mathfrak{v}) \quad \forall\, \mathfrak{v} \in V, \tag{2.4}$$

where

$$\begin{aligned} \mathfrak{u} &= (u, \boldsymbol{\sigma}) \in L^2(\Omega) \times (L^2(\Omega))^3, \\ \hat{\mathfrak{u}} &= (\hat{u}, \hat{\sigma}_n) \in H^{1/2}(\Gamma_h) \times H^{-1/2}(\Gamma_h) : \hat{u} = u_0 \,\text{on}\, \Gamma_u, \hat{\sigma}_n = \sigma_0 \,\text{on}\, \Gamma_\sigma, \\ \mathfrak{v} &= (v, \boldsymbol{\tau}) \in H^1(\Omega_h) \times \boldsymbol{H}(\mathrm{div}, \Omega_h), \\ b(\mathfrak{u}, \mathfrak{v}) &= (\boldsymbol{\sigma}, \nabla v)_{\Omega_h} + (\boldsymbol{\sigma}, \boldsymbol{\tau})_{\Omega_h} + (u, \nabla \cdot \boldsymbol{\tau})_{\Omega_h}, \\ \hat{b}(\hat{\mathfrak{u}}, \mathfrak{v}) &= -\langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n}\rangle_{\Gamma_h} - \langle \hat{\sigma}_n, v\rangle_{\Gamma_h}, \\ l(\mathfrak{v}) &= (f, v)_{\Omega_h} + \langle \sigma_0, v\rangle_{\Gamma_\sigma} + \langle u_0, \boldsymbol{\tau} \cdot \boldsymbol{n}\rangle_{\Gamma_u}. \end{aligned} \tag{2.5}$$

In 2.5, $\langle \cdot, \cdot \rangle_{\Gamma_h}$ represents duality pairings defined over mesh skeleton Γ_h,

$$\begin{aligned} \langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n}\rangle_{\Gamma_h} &:= \sum_{K \in \Omega_h} \langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n}_K\rangle_{\partial K}\,, \\ \langle \hat{\sigma}_h, v\rangle_{\Gamma_h} &:= \sum_{K \in \Omega_h} \langle \hat{\sigma}_h, v\rangle_{\partial K}\,, \end{aligned}$$

and

$$(\cdot,\cdot)_{\Omega_h} = \sum_{K\,\in\,\Omega_h} (\cdot,\cdot)_{L^2(K)}\,.$$

For the boundary integrals to be well defined, we assume $u_0 \in H^{1/2}(\Gamma_u)$ and $\sigma_0 \in H^{-1/2}(\Gamma_\sigma)$. The broken test space is equipped with the adjoint graph norm [19, 20]:

$$\|\boldsymbol{v}\|_V^2 := \|A_h^\star \boldsymbol{v}\|^2 + \alpha\|\boldsymbol{v}\|^2 \tag{2.6}$$

where $\alpha > 0$ is a scaling constant, and $A_h^\star \boldsymbol{v} = (\nabla\cdot\boldsymbol{\tau}, \nabla v + \boldsymbol{\tau})_{\Omega_h}$ is the (formal) adjoint operator of $A_h \boldsymbol{u} = (\boldsymbol{\sigma} - \nabla u, -\nabla\cdot\boldsymbol{\sigma})_{\Omega_h}$ computed element-wise. In this work, all numerical experiments use $\alpha = 1$. Next, we briefly discuss the built-in error estimator. Let $V_h(K) \subset V(K)$ be the *enriched* finite-dimensional test space approximating the element test space $V(K)$, $(U_h, \hat{U}_h) \subset (U, \hat{U})$ the finite-dimensional approximate trial space. The basis functions for $V_h(K)$, U_h and $\hat{U}_h$ are denoted by φ_i, ψ_i and $\hat{\psi}_i$ respectively. From 2.5, we construct the following element stiffness matrices for an element $K \in \Omega_h$,

$$\begin{aligned}
\mathrm{G}_{K,lj} &= (\varphi_l, \varphi_j)_V\,,\\
\mathrm{B}_{K,ij} &= b_K(\varphi_i, \psi_j)\,,\\
\hat{\mathrm{B}}_{k,ij} &= \hat{b}_K(\varphi_i, \hat{\psi}_j)\,,\\
\mathrm{l}_{K,i} &= l_K(\varphi_i),
\end{aligned}$$

where $\mathrm{G}_{K,lj}$ represents the element Gram matrix corresponding to the test inner product, $\mathrm{B}_{K,ij}$ represents the element stiffness matrix corresponding to the L^2 variables, $\hat{\mathrm{B}}_{k,ij}$ represents the element stiffness matrix corresponding to the trace variables, and $\mathrm{l}_{K,i}$ is the element load vector. As usual, $b_K(\cdot,\cdot)$, $\hat{b}_K(\cdot,\cdot)$ and $l_K(\cdot)$ denote element K contributions to bilinear forms $b(u,v), \hat{b}(\hat{u},v)$, and linear form $l(v)$, respectively. An in-depth exposition of the algebraic structure of the linear system induced by DPG formulation for a diffusion problem can be found in [13, 21].

The built-in energy error estimate for a mesh element K in the finite element mesh (Ω_h) is given by:

$$\|(u,\hat{u}) - (u_h,\hat{u}_h)\|_{E,K}^2 := \|{R_V}^{-1}\left(l_K(\cdot) - b_K(u_h,,\cdot) - \hat{b}_K(\hat{u}_h,\cdot)\right)\|_{V(K)}^2$$

where

$$R_V\ : V(K) \to (V(K))'$$

is the Riesz operator corresponding to the test inner product. With the element test space $V(K)$ approximated by a finite-dimensional enriched subspace $V_h(K)$, the element error indicators are computed as:

$$\eta_K := \|\mathrm{G}^{-1}(l_K - \mathrm{B}_K u_h - \hat{\mathrm{B}}_K \hat{u}_h)\|_{V(K)}^2\,. \tag{2.7}$$

3. Determining Optimal *hp* Refinements

The *hp*-algorithm described in this section is *exactly* the algorithm from [2, 3], but specialized to the L^2-energy space. The corresponding algorithms for the H^1, $H(\text{curl})$, and $H(\text{div})$ energy spaces, all based on minimizing the *Projection-Based (PB) interpolation error*, are significantly more intricate and consist of several steps reflecting the nature of the particular energy space. For instance, the algorithms for H^1 and $H(\text{curl})$ spaces consist of three stages involving mesh optimization on (interiors of) edges, faces and, finally, elements. The optimal mesh determined in each step serves as a starting point for the optimization in the subsequent step.

In the case of the L^2-energy space, there are no global conformity requirements; the PB interpolation reduces to just the L^2-projection, and the mesh optimization takes place over elements only. The implementation of the algorithm is thus much simpler. The second difference between the presented and the original *hp*-algorithm lies in the involved elements. In the original algorithm, the optimization takes place over *all elements*, whereas here it only does for elements marked for refinement by the DPG residual. The number of elements entering the mesh optimization is thus much smaller.[3] The *fine mesh* providing the *reference solution* for the mesh optimization is also much smaller than the globally *hp*-refined mesh used in [2, 3]. Figure 1 illustrates a two-dimensional case of mesh elements being marked by the DPG residual, followed by their isotropic *hp*-refinement[4] to generate the *fine mesh*.

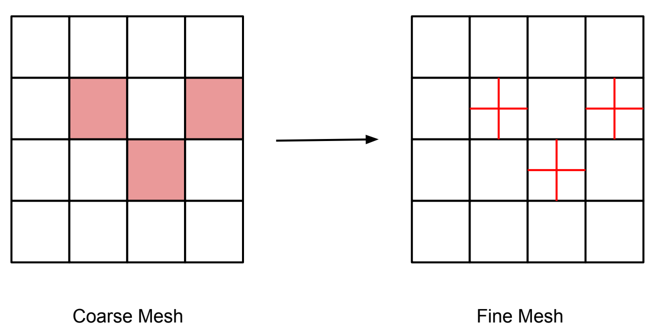


Figure 1: Isotropic *hp*-refinement of the marked elements: the elements marked for refinement are shaded in red on the coarse mesh.

The *hp*-algorithm consists of three steps: the first and third step are purely local (can be done trivially in parallel) while the second step requires a loop over all elements preselected for refinement by the DPG residual.

Step 1: Staging a competition between p and various anisotropic h-refinements, and computing the guaranteed error reduction rate. The comparison between the various candidate refinements is based on the *error reduction*

[3] Dependent upon the parameter in the Dörfler strategy [22].

[4] For a three-dimensional hexahedral element, isotropic *hp*-refinement denotes an isotropic *h*8-refinement followed by an isotropic *p*-refinement of order 1.

rate (e_{hp}) defined as:

$$e_{hp} := \frac{\|u - P_{\text{old}} u\|^2 - \|u - P_{\text{new}} u\|^2}{N_{\text{new}} - N_{\text{old}}}, \tag{3.8}$$

where u represents the *reference solution* obtained with the hp-refined mesh generated using the DPG residual, P_{old} is the L^2-projection onto the original coarse mesh element (space), P_{new} is the L^2-projection onto a *refined element* (space), N_{new} and N_{old} are dimensions of the new and old spaces (number of dof), and $\|\cdot\|$ denotes the L^2-norm over the considered element K.

The optimal element refinement is determined by staging a competition among various candidate refinements. For a hexahedral element discussed in this paper, there are eight possibilities: no h-refinement (i.e. p-refinement only), three possible anisotropic $h2$-refinements, three possible anisotropic $h4$-refinements, and the isotropic $h8$-refinement. Figure 2 illustrates all possible h-refinement candidates. Each of the eight refinements is accompanied with the determination of the optimal distribution of polynomial degrees. This leads to a catastrophically large number of possible cases. With $p = 1, \dots, 10$, there are only 10^3 scenarios for the just p-refined element, but a staggering total of 10^{24} cases for the $h8$-refined element. Clearly, a simple search through all possible cases is unfeasible. The discrete nature of the optimization also eliminates the use of gradient-based algorithms. Instead we rely on the classical p-refinement strategy, see e.g. [23], based on increasing the polynomial order in the subelement with the maximum error. This reduces the discrete search to the so-called *maximum error reduction path* through the vast discrete space of potentially possible refinements.

Maximum error reduction path for a p-refined element. We begin the discussion with the simplest case: p-refinement only. Assuming that the polynomial order can only increase, there are only a total of $2^3 - 1 = 7$ possible scenarios. The direct search is then possible but can be replaced with a slightly faster dynamic search, as illustrated in Figure 3. To choose the optimal p-refinement, we traverse from (p_x, p_y, p_z) to $(p_x + 1, p_y + 1, p_z + 1)$ by increasing the order in directions that maximize e_{hp}. For a hexahedral element, the path of traversal has two stages. The first stage has three branches corresponding to p_x, p_y, and p_z. The second stage has two branches corresponding to the remaining directions, with the final configuration being $(p_x + 1, p_y + 1, p_z + 1)$. In Figure 3, the arrows in red represent the branches corresponding to the highest values of e_{hp} at each stage, and the polynomial order marked in red indicates the polynomial order increased after each stage.

Following the path, we select the p-refinement that delivers the largest error reduction rate. In the case of an affine element, the element Jacobian (jac) is constant, and the L^2-Piola transform (pullback map) reduces to a scaling with the Jacobian:

$$\phi_j(x) = \frac{1}{\text{jac}} \hat{\phi}_j(\xi), \qquad \text{jac} = \left| \frac{\partial x_i}{\partial \xi_j} \right|,$$

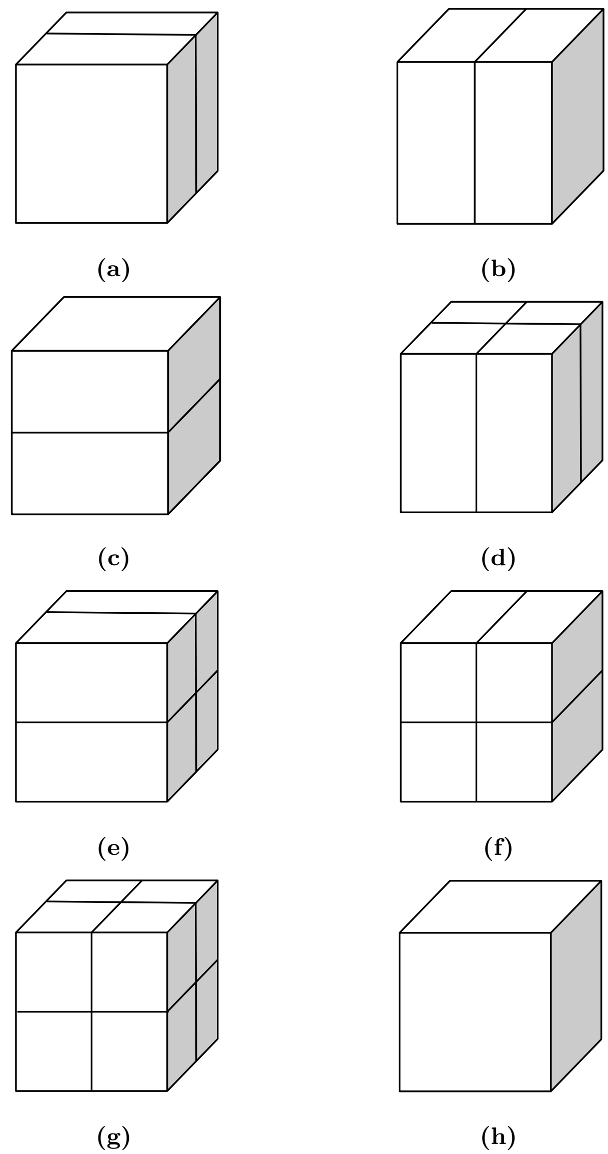


Figure 2: Various possible h-refinements for a hexahedral element: (a),(b) and (c) depict anisotropic $h2$-refinements. Anisotropic $h4$-refinements are illustrated by (d),(e) and (f). Finally, (g) and (h) depict isotropic $h8$-refinement and p-only refinement (no h-refinement), respectively.

where ϕ_j is an element L^2 shape function corresponding to a master element shape function $\hat{\phi}_j$. Consequently, the L^2 mass matrix,

$$M_{ij} := \int_K \phi_i \phi_j \, dx = \frac{1}{\text{jac}} \int_{\hat{K}} \hat{\phi}_i \hat{\phi}_j \, d\xi \,,$$

is diagonal, and the evaluation of the L^2 projection of a function u onto a subspace spanned by functions $\phi_1, \ldots, \phi_N$, reduces to the evaluation of the load vector:

$$P_N u = \sum_{j=1}^{N} u_j \phi_j, \qquad u_j = \frac{\int_K u \phi_j \, dx}{M_{jj}} \,.$$

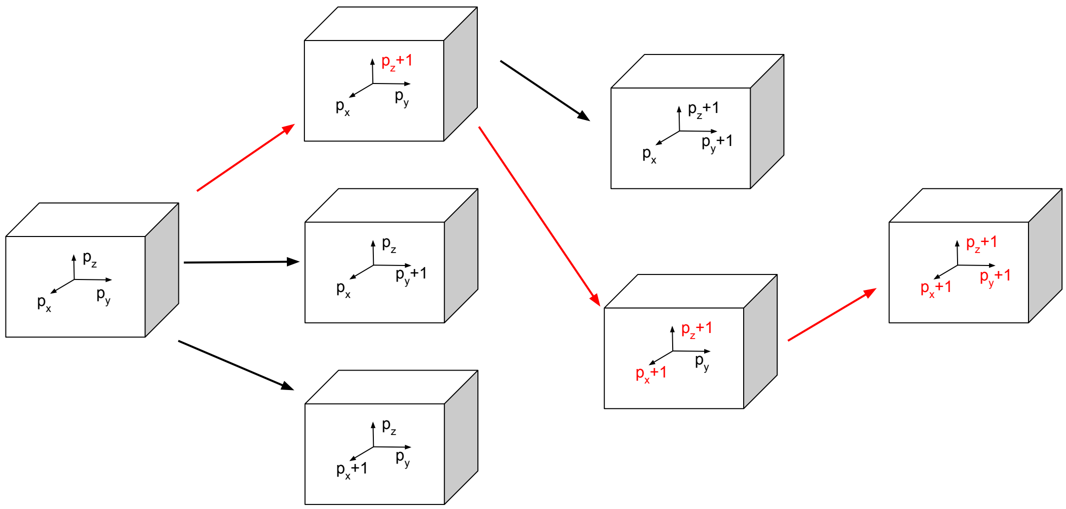


Figure 3: Maximum error reduction path for the p-refined element: traversing from (p_x, p_y, p_z) to $(p_x + 1, p_y + 1, p_z + 1)$ for a hexahedral element.

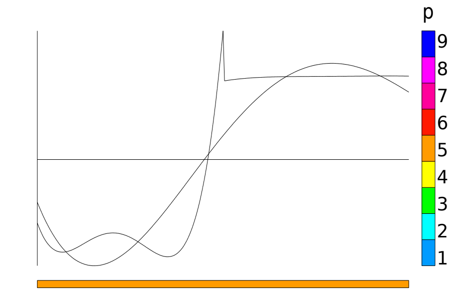
Raising the polynomial order in one direction amounts to adding extra orthogonal shape functions ϕ_{N+l} with $l = 1, ..., n$. Consequently, evaluation of the error reduction rate reduces simply to:

$$
\begin{aligned}
\frac{\|u - P_N u\|^2 - \|u - P_{N+1} u\|^2}{n} &= \frac{\|P_{N+1} u\|^2 - \|P_N u\|^2}{n} \\
&= \frac{\sum_{l=1}^{n} |u_{N+l}|^2 M_{N+l,N+l}}{n} = \frac{1}{n} \sum_{l=1}^{n} \left(\frac{\int_K u \phi_{N+l} \, dx}{M_{N+l,N+l}} \right)^2 M_{N+l,N+l} \\
&= \frac{\sum_{l=1}^{n} M_{N+l,N+l}^{-1} \left(\int_K u \phi_{N+l} \, dx \right)^2}{n} .
\end{aligned}
$$

In the case of a general curvilinear element, the L^2 mass matrix is not diagonal, and we use the telescopic solver based on the Cholesky decomposition described in [3], p.140.

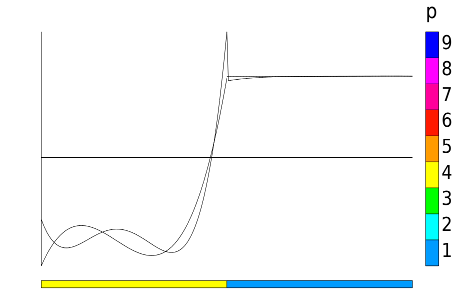
Maximum error reduction path for an h-refined element. Contrary to the pure p-refinement, we always start with a trilinear element where $p_x = p_y = p_z = 1$. The reference solution u is projected onto the subelement mesh and, based on the distribution of the error, subelements are selected for refinement using a *greedy strategy* with a 70% factor. Once the subelements have been identified for p-refinement, the routine described above is employed to determine the optimal p-refinement for each subelement. The path is illustrated in Figure 4 and Figure 5 for the simple case of an h-refined 1D element. The maximum degree ceiling is determined by the polynomial degree of the fine mesh elements. As we proceed along the refinement path, the corresponding error reduction rate is computed for each subelement mesh.

Selection of the optimal refinement is carried out by comparing the best error reduction rates delivered by the eight differently h-refined meshes. We document the best error reduction rate and call it the element *guaranteed error reduction rate*.



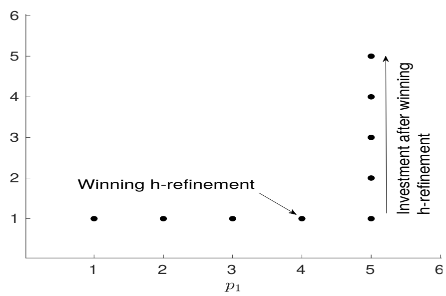
(a) p-refinement:

$$r := \frac{\|u_{h/2,p+1} - u_{h,p}\|^2 - \|u_{h/2,p+1} - u_{h,p+1}\|^2}{1}$$
$$= 0.27609 - 0.25657 = 0.01952$$

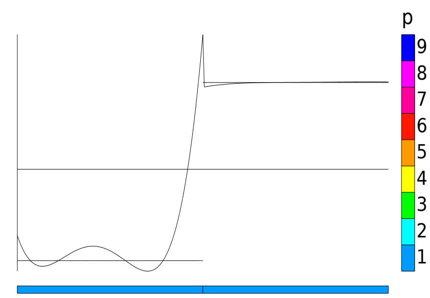


(b) Winning h-refinement:

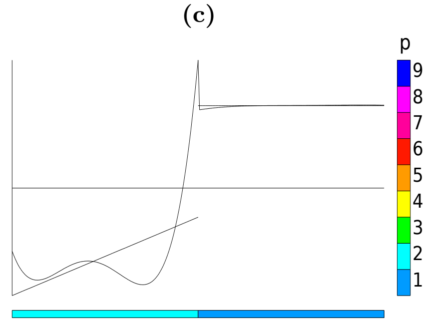
$$r := \frac{\|u_{h/2,p+1} - u_{h,p}\|^2 - \|u_{h/2,p+1} - u_{h,p+1}\|^2}{\Delta N}$$
$$= 0.27609 - 0.04413 = 0.23196$$



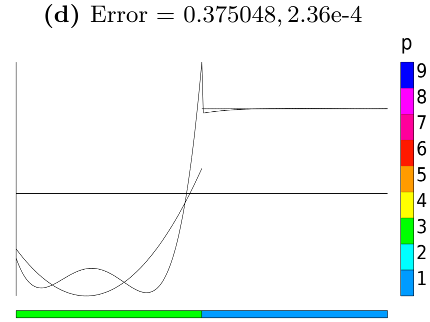
(c)



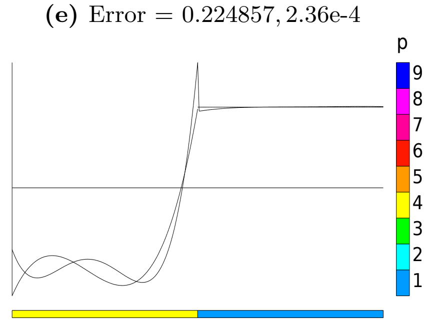
(d) Error $= 0.375048, 2.36\text{e-}4$



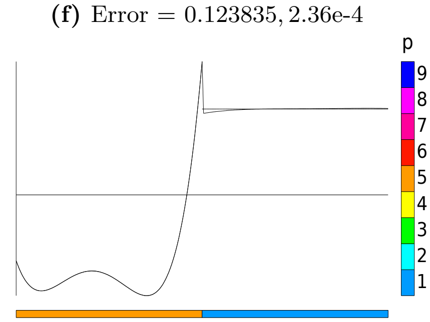
(e) Error $= 0.224857, 2.36\text{e-}4$



(f) Error $= 0.123835, 2.36\text{e-}4$

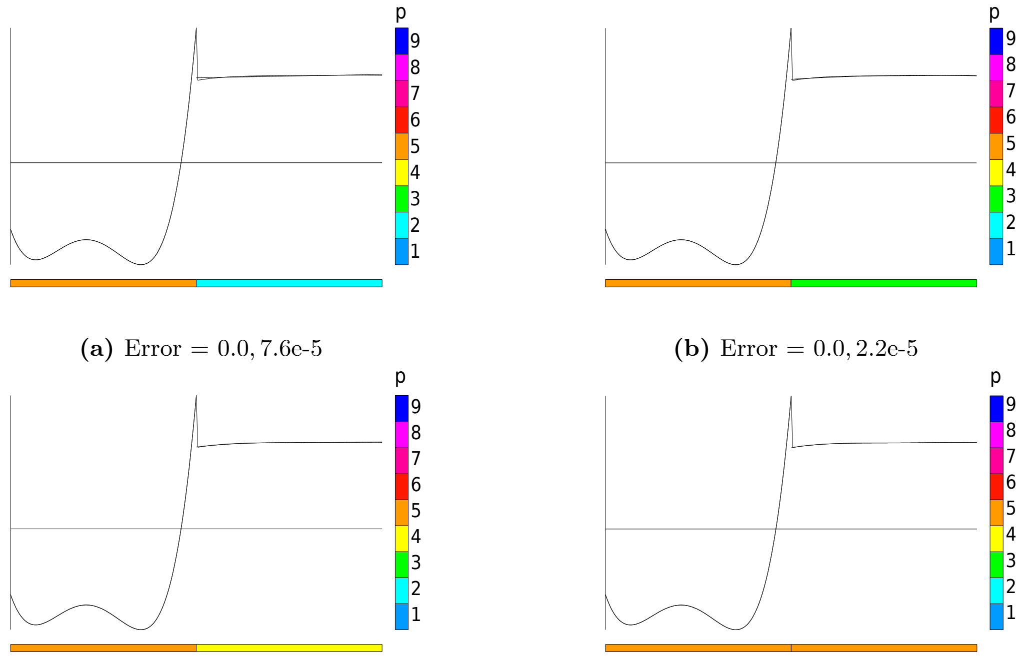


(g) Error $= 0.043898, 2.36\text{e-}4$



(h) Error $= 0.0, 2.36\text{e-}4$

Figure 4: Maximum error reduction path: traversing from $(p_1, p_2) = (1, 1)$ to $(p_1, p_2) = (5, 1)$. Error denotes the projection error in each element.



(a) Error $= 0.0, 7.6\text{e-}5$

(b) Error $= 0.0, 2.2\text{e-}5$

(c) Error $= 0.0, 4.0\text{e-}6$

(d) Error $= 0.0, 0.0$

Figure 5: Maximum error reduction path: traversing from $(p_1, p_2) = (5, 2)$ to $(p_1, p_2) = (5, 5)$. Error denotes the projection error in each element.

Step 2: Determining which elements to refine. We loop over all considered coarse mesh elements to determine the element with the *best guaranteed error reduction rate*. In principle, one could then refine only this one element. However, to accelerate the refinements, the greedy strategy is employed, selecting all elements that deliver a rate greater than or equal to 25% of the best guaranteed error reduction rate. Thus, in general, there may be elements selected for refinement by the DPG residual which, after the comparison, will remain unrefined.

Step 3: Determining the final refinements. We can simply execute the optimal refinements determined in Step 1, and this is indeed the case for the purely p-refined elements. However, in general, we invest more dofs while performing h-refinements. This approach is best understood by drawing a financial analogy with investments. Suppose we have unlimited funds and are trying to find banks to invest in. The banks offer different rates with some bank(s) offering, say, 10% and others less. We decide to invest in all banks that guarantee at least 7% interest for a 1M investment. We invest 1M into each of those banks but we learn that the best bank(s) offers a slightly *higher* rate of 8% if we invest 2M (or more). Well, if we do have unlimited funds, it does make sense to invest more than 1M in such a bank!

This is exactly what happens in the third step of our algorithm for elements selected for an h-refinement. In Step 1, we record the error reduction rates for all subelement meshes following the maximum error reduction path. Starting with the best refinement (that won the competition), we investigate the following p-refinements and select the maximum investment that still delivers the 25% of the best guaranteed error reduction rate. In the 1D case illustrated in Figure 4, case (g) won the competition with the p-refinement but (dependent upon the threshold value used in the greedy strategy), we may select case (h) with an additional dof invested in the first subelement.

Next, we consolidate all the steps (1-3) and present the mesh optimization algorithm. In Algorithm 1, tol denotes the user-provided tolerance value for the DPG residual.

Algorithm 1 Mesh Optimization Algorithm

Start with an initial trial mesh
while $\eta_{\Omega_h} > \text{tol}$ **do**
    Solve the problem on the current mesh.
    Compute the DPG residual for the current mesh: $\eta_{\Omega_h} = \left(\sum_{K\in\Omega_h} \eta_K\right)^{1/2}$.
    Use the element residuals (η_k) to mark elements for refinements (Dörfler strategy).
    Isotropically hp-refine marked elements to generate the fine mesh.
    Compute the reference solution u using the fine mesh.
    Step 1: For each refined element K :
        Determine the best possible p-refinement using the maximum error reduction path.
        Determine the best possible h-refinement using the maximum error reduction path.
        Use error reduction rates to decide between p- and h-refinement.
        Determine the element guaranteed error reduction rate ($r_{g,K}$).
    Step 2: Loop over all marked elements to determine the best guaranteed error reduction rate (r_m).
    Unrefine the mesh.
    Step 3: For each element K marked for refinement:
        if $r_{g,K} \geq 0.25\, r_m$ then
            perform the optimal hp-refinement.
        end if
endwhile

3.1. Mesh Closure

The *hp* algorithm has been implemented in *hp*3D, a general-purpose FE code supporting hybrid meshes consisting of elements of all shapes (hexas, tets, prisms, pyramids), conforming discretizations of the exact sequence spaces (H^1-, $H(\text{curl})$-, $H(\text{div})$-, and L^2-conforming elements), solution of coupled multiphysics problems, and *anisotropic hp*-refinements [24, 25]. *hp*3D is available under BSD-3 license.[5] In the code, any

[5] `https://github.com/Oden-EAG/hp3d`

h-refinement is executed in two steps. Given a list of elements to refine (along with the requested, possibly anisotropic, h-refinement flags), we proceed as follows.

Step 1 (local): Refine the elements from the list in the provided order, enforcing two rules:

- **compatibility with existing face refinements:** upgrade the requested element refinement flag to accommodate *existing face refinements.*
- **one-irregularity rule for faces:** employ the standard *shelf* or *queue* algorithm ([13], p.71) to ensure that no face is refined unless the face[6] is unconstrained.

If one of the element faces is constrained, the element is placed on the shelf, and a necessary refinement of the neighbor across the face is executed, to eliminate the constraint. If the one-irregularity rule for faces prohibits the refinement, the corresponding neighbor is placed on the shelf and so on. Once the refinement of the processed element is possible, it is executed and the process resumes with the last element from the shelf. The algorithm proceeds until the shelf is empty. All mesh manipulations (refinements) are supported for meshes that satisfy the one-irregularity rule for faces (not necessary for edges and vertices).

Step 2 (global): loop through all elements and perform additional necessary refinements to eliminate edges and vertices with multiple constraints.

We refer to [24] for a more formal exposition of the algorithms. In the end, in both steps, a number of additional, *unwanted* refinements are executed. These refinements can be *isotropic* or *anisotropic*, reflecting minimal requirements to eliminate the nodes with multiple constraints. In the 'global' hp-refinement driven by the DPG residual, all unwanted refinements are chosen to be isotropic. This is motivated by the fact that an unwillingly refined element (in Step 1) may, in fact, be on the DPG list of wanted refinements. However, once the optimal hp-refinements are determined, all unwanted refinements are executed in a minimal, anisotropic way.

All unwillingly h-refined elements inherit their polynomial order from the father element. In principle, one could attempt to find the corresponding optimal distribution of polynomial order p, but this has been not done in our current implementation. Hence, the presented meshes may be slightly overrefined.

[6]More precisely, the mid-face node.

4. Numerical Results

4.1. A Boundary Layer Problem

Sharp boundary layers are among the most commonly encountered flow features in computational fluid dynamics. This numerical experiment demonstrates the proposed algorithm's efficacy in resolving such boundary layers. In this test case, we solve a Poisson problem with a manufactured solution containing boundary layers. The manufactured solution is a three-dimensional extension of the solution of the Egger-Schöberl problem [26]. In particular, we solve,

$$
\begin{aligned}
-\nabla^2 u &= f(x,y,z) && \text{in} \quad \Omega := (0,1)^3, \\
u &= 0 && \text{on} \quad \Gamma_u, \\
\nabla u \cdot \boldsymbol{n} &= g(x,y,z) && \text{on} \quad \Gamma_\sigma,
\end{aligned}
\tag{4.9}
$$

where

$$
\begin{aligned}
\Gamma_u &= ([0,1) \times [0,1) \times \{0\}) \cup ([0,1) \times \{0\} \times [0,1)) \cup (\{0\} \times [0,1) \times [0,1)) \quad \text{and} \\
\Gamma_\sigma &= ([0,1] \times [0,1] \times \{1\}) \cup ([0,1] \times \{1\} \times [0,1]) \cup (\{1\} \times [0,1] \times [0,1]) \, .
\end{aligned}
$$

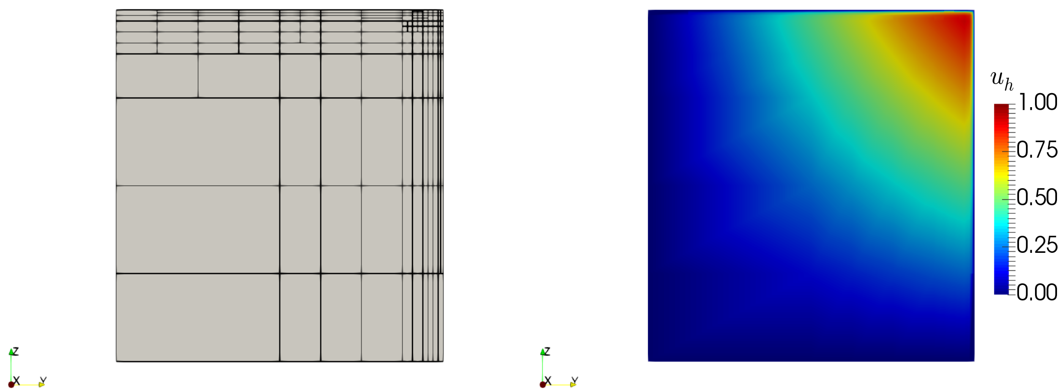
In 4.9, $\boldsymbol{n}$ is the outward normal, $f(x,y,z)$ and $g(x,y,z)$ are generated using the exact solution. The exact solution is given by

$$
u(x,y,z) = \left(x + \frac{e^{x/\epsilon} - 1}{1 - e^{1/\epsilon}} \right) \left(y + \frac{e^{y/\epsilon} - 1}{1 - e^{1/\epsilon}} \right) \left(z + \frac{e^{z/\epsilon} - 1}{1 - e^{1/\epsilon}} \right) .
$$

The solution exhibits a boundary layer near $x \approx 1$, $y \approx 1$ and $z \approx 1$. The strength of the boundary layer is inversely proportional to ϵ. In this numerical experiment, $\epsilon = 0.005$. The *hp*-adaptation is initialized with a mesh comprising only eight elements with a constant polynomial order of $(2,2,2)$.[7]

Figure 6a and Figure 6b display the cross-section of an adapted mesh and the corresponding solution contour, respectively. Figure 7 depicts the polynomial distribution around the boundary layers on an anisotropically adapted *hp*-mesh. Figure 8 presents the convergence results, comparing isotropic *h*-adaptation and the proposed *hp*-refinement strategy. The Dörfler parameter for both isotropic and *hp*-refinement is 0.75. In Figure 8, the relative error denotes combined relative error in all L^2 variables. Figure 7 clearly illustrates the strong anisotropy and grading in the element size and the polynomial distribution. The anisotropy and the grading in element size are paramount for resolving strong boundary layers efficiently. The algorithm also prescribes an anisotropic polynomial distribution in the boundary layers instead of an isotropic one. This directional

[7] In *hp*3D, we employ exact-sequence spaces [27]. Hence, an order of (p_x, p_y, p_z) denotes L^2 shape functions of order $(p_x - 1, p_y - 1, p_z - 1)$.



(a) Cross-section of the mesh at $x = 0.95$ **(b)** Contour plot of the solution at $x = 0.95$

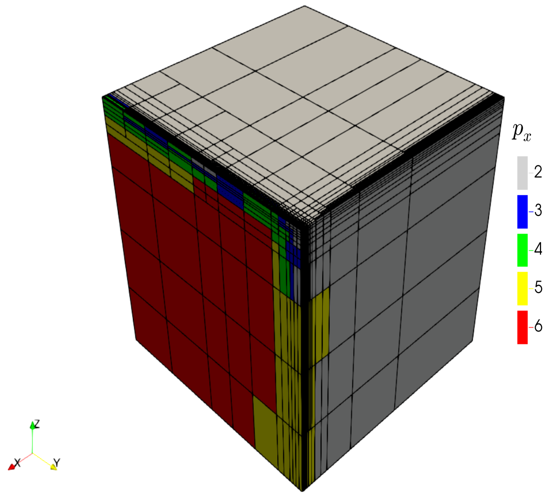
Figure 6: Boundary layer problem: (a) cross-section of the mesh showing anisotropic elements required to resolve the boundary layers, and (b) contour plot illustrating the boundary layers on the yz-plane. The boundary layers are along right and top faces of the cross-section.

preference of prescribing polynomial orders showcases a significant advantage of the proposed hp-refinement strategy: the ability to complement an anisotropic h-refinement with an anisotropic p-refinement. This approach makes the refinement strategy highly efficient in terms of allocating dofs when the solution exhibits strong anisotropic features. The algorithm does not waste any dofs in directions where the solution variables do not exhibit significant variations.

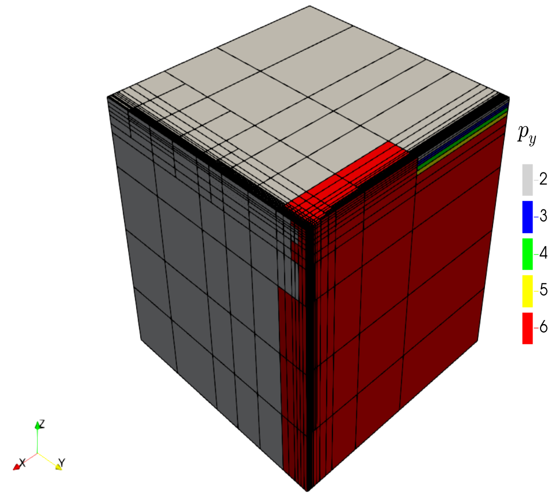
From Figure 8, it is evident that anisotropic hp-refinements outperform isotropic h-refinements by orders of magnitude. The convergence plots shows the error and the residual against $\sqrt[3]{\text{ndof}}$ (where ndof represents the number of degrees of freedom), verifying exponential convergence. In Figure 8, a reduction in the convergence rate for the hp-refinement can be observed. The slowdown in convergence occurs due to the limiting of the highest polynomial order in the numerical experiments to $p = 6$. The adaptation cycles are initially dominated by h-refinements. Once the boundary layers are resolved, the algorithm starts preferring both p-refinements along with h-refinements. This behavior is expected, since, increasing the polynomial order on coarse meshes while approximating solutions with high gradients can induce spurious oscillations.

4.2. Fichera Cube Problem

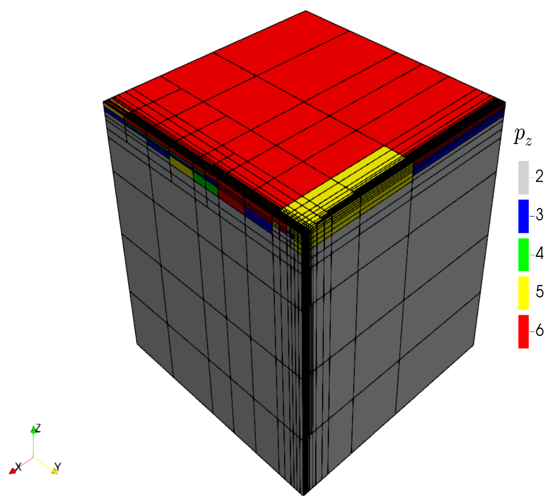
To demonstrate the efficacy of the proposed refinement strategy in the presence of multiple singularities, we solve the well-known Fichera cube problem and perform hp-adaptations using the proposed refinement



(a) Polynomial order along x direction: p_x



(b) Polynomial order along y direction: p_y



(c) Polynomial order along z direction: p_z

Figure 7: Boundary layer problem: an adapted mesh with 855 532 dof and the corresponding polynomial distribution. The algorithm prescribes higher-order polynomials anisotropically corresponding to each boundary layer along x,y, and z-axis.

strategy. The variant of the Fichera cube problem being solved here is given by:

$$\begin{aligned} \nabla^2 u &= 0 && \text{in} \quad \Omega := (-1,1)^3 \setminus [0,1]^3, \\ u &= 0 && \text{on} \quad \Gamma_u, \\ \nabla u \cdot \boldsymbol{n} &= g(x,y,z) && \text{on} \quad \Gamma_\sigma. \end{aligned} \tag{4.10}$$

The domain is created by subdividing a large cube (denoted by $(-1,1)^3$) into eight smaller cubes and then removing one of the cubes. The Dirichlet data $u = 0$ is imposed on the three square faces aligned with planes

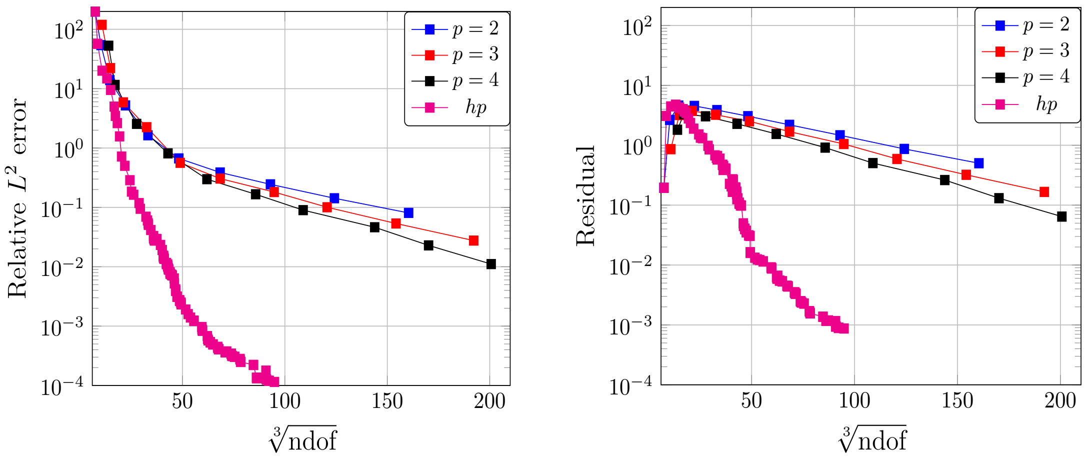


Figure 8: Boundary layer problem: convergence of relative L^2 error and DPG residual. Even though there is a marginal decrease in the rate of convergence for the hp-refinements, both the error and the residual are 2-3 orders of magnitude lower compared to the h-refinements for approximately same number of dof.

of coordinate axes, i.e.

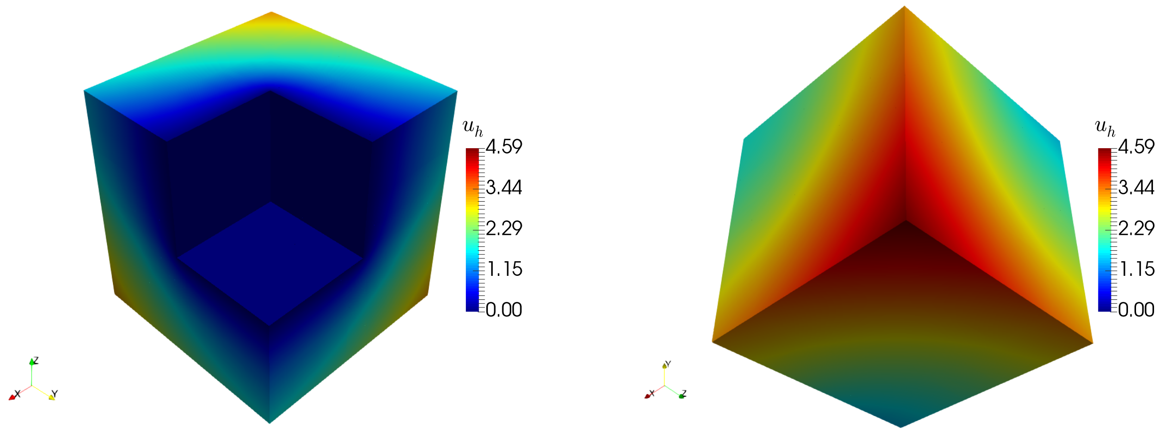
$$\Gamma_u = ([0,1]\times[0,1]\times\{0\}) \cup ([0,1]\times\{0\}\times[0,1]) \cup (\{0\}\times[0,1]\times[0,1])\,.$$

The volumetric load for the problem is 0. The problem is driven by the Neumann boundary conditions on Γ_σ composed of the remaining faces of the cube. The data $g(x, y, z)$ corresponds to the sum of two-dimensional exact solutions of the L-shaped domain problem on xy, yz and xz planes. The exact solution of the L-shaped domain problem is given by:

$$u_{\eta,\xi} = r^{\frac{2}{3}} cos(\theta), \quad r = \sqrt{\eta^2 + \xi^2}, \quad \theta = \tan^{-1}\left(\frac{\xi}{\eta}\right), \tag{4.11}$$

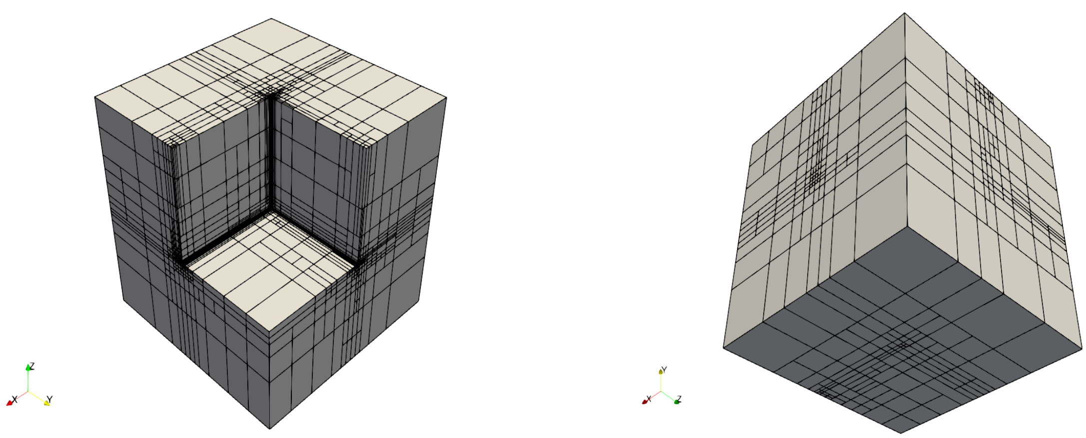
where (η, ξ) denote (x, y) or (y, z), or (x, z) axes, respectively. These boundary conditions generate a solution with features analogous to an L-shaped domain problem but comprising multiple edge and vertex singularities. While the exact solution for the problem is unknown, the convergence of the DPG residual is shown in Figure 13.

Figure 9 and Figure 10 depict the solution contour and the corresponding adapted mesh, respectively. Figure 11 and Figure 12 illustrate the polynomial distribution associated with the adapted mesh. Figure 10 shows that the refinement algorithm performs highly anisotropic h-refinements along the edge singularities to generate graded meshes. The anisotropic refinements propagate through the volume to the opposing boundary faces on Γ_σ. The propagation of refinements happens in conjunction to the singularities arising from the faces with Neumann boundary conditions. Figure 11 and Figure 12 clearly indicate that the algorithm chooses lowest order polynomials around the singularities. Moving away from the singularities, the algorithm prescribes higher order polynomials underscoring the smoothness of the solution variables. In Figure 13, one can observe the exponential convergence of the residual on performing hp-refinements, whereas isotropic h-refinements suffer from a loss of convergence due to the lack of required grading in size and polynomial distribution.



(a) Isometric view along $(-1, -1, -1)$ **(b)** Isometric view along $(1, 1, 1)$

Figure 9: Fichera cube problem: solution contour. The problem is driven by the Neumann boundary conditions on the L-shaped faces in (a) and the three visible square faces in (b). The faces aligned along the coordinate planes in (a) have the Dirichlet boundary conditions.



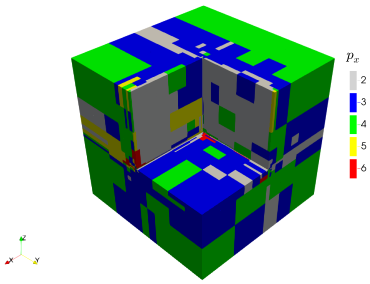
(a) Isometric view along $(-1, -1, -1)$ **(b)** Isometric view along $(1, 1, 1)$

Figure 10: Fichera cube problem: an anisotropically adapted hp-mesh with 1.3M dofs.

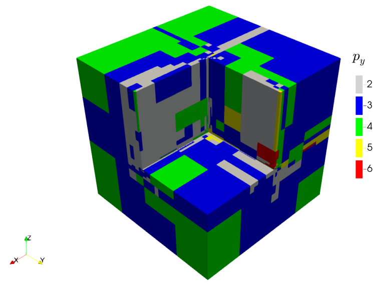
4.3. Eriksson–Johnsson Problem

Next, we present our final numerical experiment. We consider a convection-dominated diffusion problem motivated by the Eriksson–Johnson model problem [28]. Here, we extend the exact solution of the two-dimensional problem by multiplying it with a sinusoidal term along z. In particular, we solve,

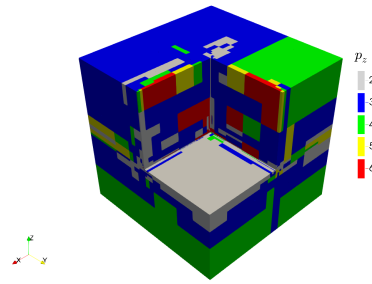
$$\begin{aligned} \frac{\partial u}{\partial x} - \epsilon \nabla^2 u &= f(x, y, z) && \text{in} \quad \Omega := (0,1)^3, \\ u &= 0 && \text{on} \quad \Gamma_{u_a}, \\ u &= \sin(\pi y)\sin(\pi z) && \text{on} \quad \Gamma_{u_b}, \end{aligned} \tag{4.12}$$



(a) Polynomial order along x direction: p_x

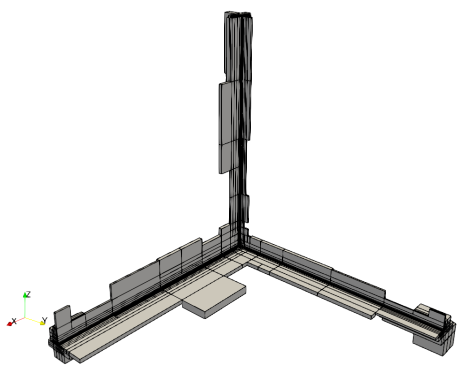


(b) Polynomial order along y direction: p_y

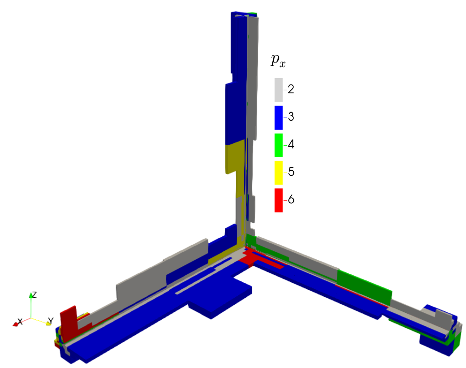


(c) Polynomial order along z direction: p_z

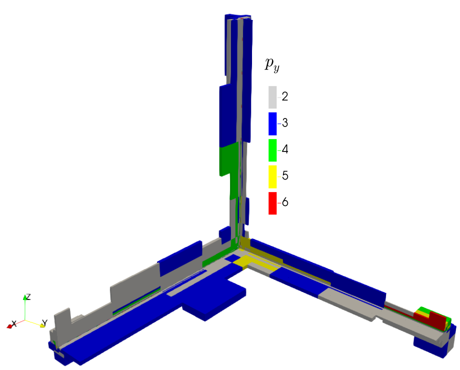
Figure 11: Fichera cube problem: polynomial distribution on the adapted hp-mesh. The algorithm prescribes low-order polynomials anisotropically around each edge singularity along x, y and z-axis. Figure 12 presents a magnified view of the polynomial distribution and anisotropic mesh elements around the singularities.



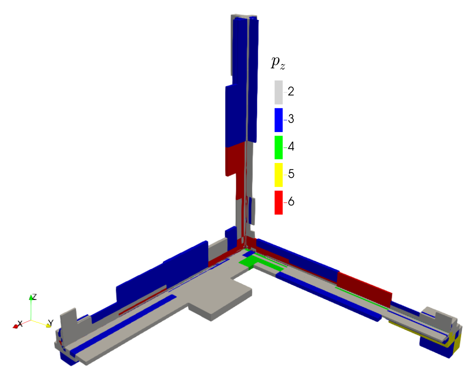
(a) Adapted mesh



(b) Polynomial order along x direction: p_x



(c) Polynomial order along y direction: p_y



(d) Polynomial order along z direction: p_z

Figure 12: Fichera cube problem: magnified view of the mesh and the polynomial distribution near the edge and vertex singularities.

where

$$\Gamma_{u_a} = \partial\Omega \setminus \{0\} \times [0,1] \times [0,1] \quad \text{and} \quad \Gamma_{u_b} = \{0\} \times [0,1] \times [0,1].$$

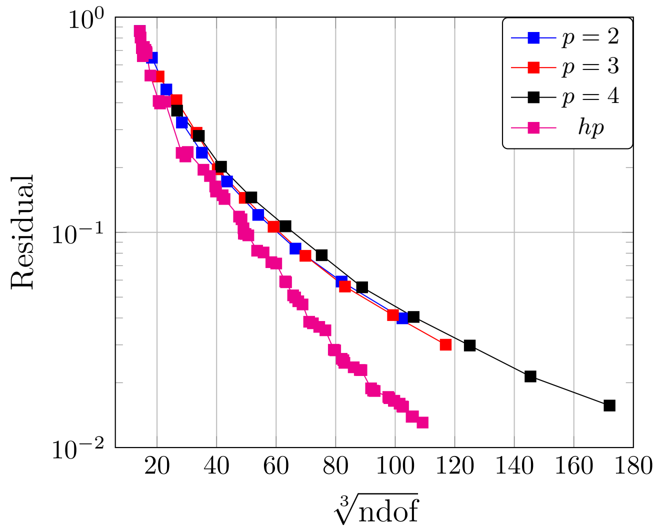


Figure 13: Fichera cube problem: convergence of DPG residual.

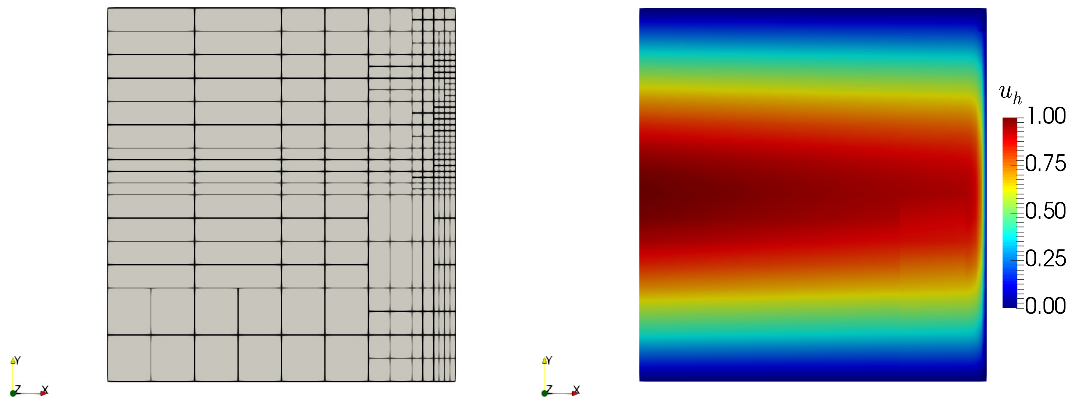
The source $f(x, y, z)$ and the boundary conditions are computed using the exact solution. The exact solution is given by

$$u(x, y, z) = \frac{e^{s_1(x-1)} - e^{s_2(x-1)}}{e^{s_1} - e^{s_2}} \sin(\pi y) \sin(\pi z), \quad \text{where } s_1 = \frac{1 + \sqrt{1 + 4\pi^2\epsilon^2}}{2\epsilon} \quad \text{and} \quad s_2 = \frac{1 - \sqrt{1 + 4\pi^2\epsilon^2}}{2\epsilon}.$$

In this numerical experiments, $\epsilon = 0.01$. Figure 14 depicts the cross-section of an adapted mesh and the corresponding solution contour at $z = 0.5$. The solution exhibits a boundary layer along the x-axis with sinusoidal variations along y and z. The variation in the solution is also reflected in the hp-refinements executed by the algorithm. In order to capture the boundary layer, the algorithm generates anisotropic elements parallel to the yz-plane and assigns the highest polynomial order along the x-axis inside the boundary layer. Since the boundary layer is weighted with sinusoidal variations in y and z, the majority of the h-refined elements in the boundary layer are positioned near $y = 0.5$ and $z = 0.5$. Figure 15 illustrates the adapted mesh with polynomial distribution along the x-axis. Finally, Figure 16 presents the convergence plots for the relative L^2 error and the residual, demonstrating the efficacy of the proposed hp-refinement strategy.

5. Conclusion

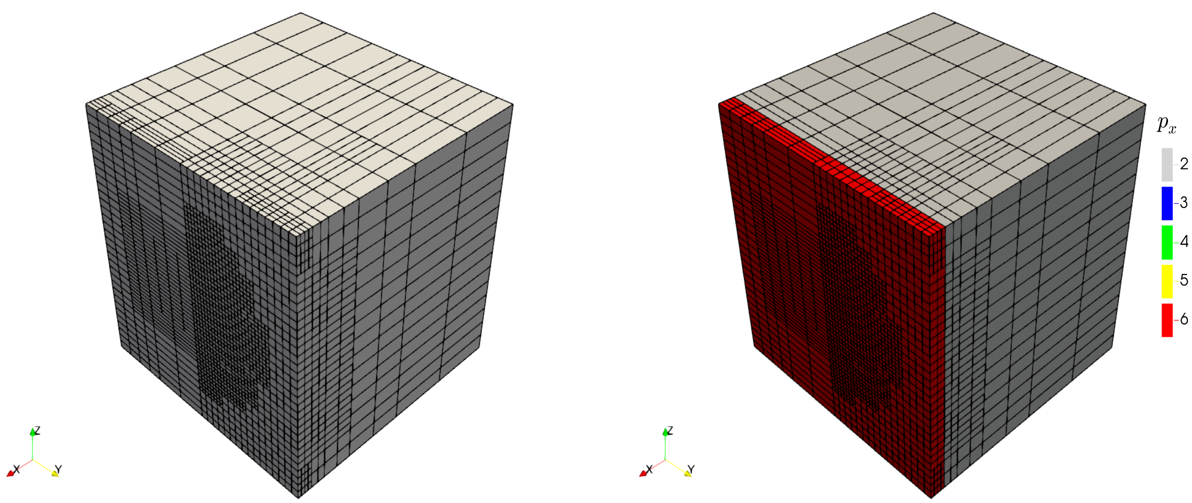
The anisotropic hp-refinement strategy presented in this article utilizes the built-in DPG error-estimator and L^2 projection-based error estimates for the ultraweak variational formulation. The efficacy of the proposed algorithm is demonstrated through numerical experiments containing boundary layers and singularities. The algorithm is able to generate a sequence of meshes that provide exponential convergence. Since we have capped the maximum polynomial order in our numerical experiments to $p = 6$ for practical reasons, we observe a slight loss of optimal convergence rate. Nonetheless, the accuracy of the solutions on the anisotropically



(a) Cross-section of an adapted mesh at $z = 0.5$

(b) Solution contour at $z = 0.5$

Figure 14: Eriksson–Johnsson problem: an adapted mesh and solution contour.



(a) Isometric view of the mesh.

(b) polynomial order along x direction: p_x

Figure 15: Eriksson–Johnson problem: an adapted mesh with 209 737 dofs and corresponding polynomial distribution along x-axis.

refined hp-meshes remains orders of magnitude better than that on isotropically refined meshes for nearly same number of dof. The proposed hp-refinement strategy complements anisotropic h-refinements with anisotropic p-refinements, which allows the algorithm to avoid any superfluous investment (in terms of dofs).

Future work: To accelerate the computation of the fine-grid solution and apply the hp-refinement strategy to large-scale multiphysics problems, we intend to integrate the proposed hp-refinement strategy with the

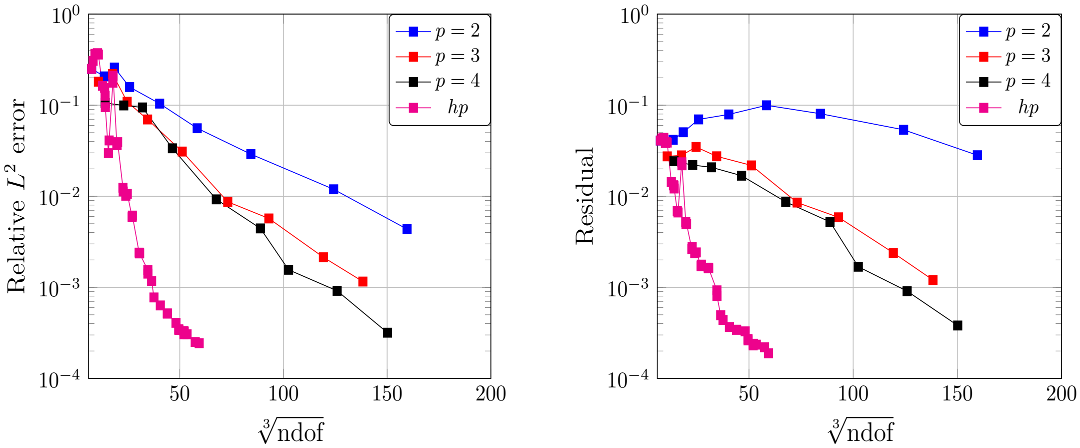


Figure 16: Eriksson–Johnson problem: convergence of relative L^2 error and DPG residual.

scalable DPG-MG solver [29]. Additionally, we aim to extend the proposed refinement strategy to other element types, such as tets, prisms, and pyramids, in order to leverage *hp*3D's capability to handle hybrid meshes.

References

[1] D. Stanzione, J. West, R. T. Evans, T. Minyard, O. Ghattas, D. K. Panda, Frontera: The evolution of leadership computing at the national science foundation, PEARC '20, Association for Computing Machinery, 2020.

[2] L. Demkowicz, Computing with *hp*-Adaptive Finite Elements. Vol. I: One and Two Dimensional Elliptic and Maxwell Problems, Chapman and Hall/CRC, 2006.

[3] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszynski, W. Rachowicz, A. Zdunek, Computing with *hp*-Adaptive Finite Elements. Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications, Chapman and Hall/CRC, 2007.

[4] I. Babuška, B. Guo, The h, p and *hp*-version of the finite element method; basis theory and applications, Advances in Engineering Software 15 (3) (1992) 159–174.

[5] I. Babuška, W. Gui, The h, p and *hp*-versions of the finite element method in 1 dimension. Part III. The adaptive *hp*-version, Numerische Mathematik 49 (1986) 659–684.

[6] W. Rachowicz, D. Pardo, L. Demkowicz, Fully automatic *hp*-adaptivity in three dimensions, Computer Methods in Applied Mechanics and Engineering 195 (37) (2006) 4816–4842.

[7] C. Schwab, *p*- and *hp*-finite element methods : theory and applications in solid and fluid mechanics, Clarendon press, 1998.

[8] I. Babuška, T. Strouboulis, K. Copps, *hp* optimization of finite element approximations: Analysis of the optimal mesh sequences in one dimension, Computer Methods in Applied Mechanics and Engineering 150 (1) (1997) 89–108.

[9] J. Oden, L. Demkowicz, W. Rachowicz, T. Westermann, Toward a universal *hp*-adaptive finite element strategy, part 2. a posteriori error estimation, Computer Methods in Applied Mechanics and Engineering 77 (1) (1989) 113–180.

[10] L. Demkowicz, J. Oden, W. Rachowicz, O. Hardy, Toward a universal h-p adaptive finite element strategy, part 1. constrained approximation and data structure, Computer Methods in Applied Mechanics and Engineering 77 (1) (1989) 79–112.

[11] W. Rachowicz, J. Oden, L. Demkowicz, Toward a universal h-p adaptive finite element strategy part 3. design of h-p meshes, Computer Methods in Applied Mechanics and Engineering 77 (1) (1989) 181–212.

[12] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation, Computer Methods in Applied Mechanics and Engineering 199 (23) (2010) 1558–1572.

[13] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions, Numerical Methods for Partial Differential Equations 27 (1) (2011) 70–105.

[14] L. Demkowicz, J. Gopalakrishnan, A. Niemi, A class of discontinuous Petrov–Galerkin methods. Part III: Adaptivity, Applied Numerical Mathematics 62 (4) (2012) 396–427.

[15] J. Zitelli, I. Muga, L. Demkowicz, J. Gopalakrishnan, D. Pardo, V. Calo, A class of discontinuous Petrov–Galerkin methods. part iv: The optimal test norm and time-harmonic wave propagation in 1d, Journal of Computational Physics 230 (7) (2011) 2406–2432.

[16] I. Babuška, Error-bounds for finite element method, Numerische Mathematik 16 (1971) 322–333.

[17] C. Carstensen, L. Demkowicz, J. Gopalakrishnan, Breaking spaces and forms for the DPG method and applications including Maxwell equations, Computers & Mathematics with Applications 72 (3) (2016) 494–522.

[18] L. Demkowicz, Energy Spaces, Lecture notes, The University of Texas at Austin (2018).

[19] L. Demkowicz, N. Heuer, Robust DPG method for convection-dominated diffusion problems, SIAM Journal on Numerical Analysis 51 (5) (2013) 2514–2537.

[20] J. Chan, N. Heuer, T. Bui-Thanh, L. Demkowicz, A robust DPG method for convection-dominated diffusion problems II: Adjoint boundary conditions and mesh-dependent test norms, Computers & Mathematics with Applications 67 (4) (2014) 771–795.

[21] A. Vaziri Astaneh, F. Fuentes, J. Mora, L. Demkowicz, High-order polygonal discontinuous Petrov–Galerkin (PolyDPG) methods using ultraweak formulations, Computer Methods in Applied Mechanics and Engineering 332 (2018) 686–711.

[22] W. Dörfler, A convergent adaptive algorithm for Poisson's equation, SIAM Journal on Numerical Analysis 33 (3) (1996) 1106–1124.

[23] L. Demkowicz, J. T. Oden, T. Strouboulis, Adaptive finite elements for flow problems with moving boundaries. Part 1: Variational principles and a posteriori estimates, Computer Methods in Applied Mechanics and Engineering 46 (1984) 217–251.

[24] S. Henneking, L. Demkowicz, Computing with *hp* Finite Elements III. Parallel *hp*3D Code, 2023, in preparation, available upon request.

[25] S. Henneking, L. Demkowicz, hp3d user manual, arXiv preprint arXiv:2207.12211.

[26] H. Egger, J. Schöberl, A hybrid mixed discontinuous Galerkin finite element method for convection–diffusion problems, IMA Journal of Numerical Analysis 30 (4) (2009) 1206–1234.

[27] L. Demkowicz, Mathematical Theory of Finite Elements , Lecture notes, The University of Texas at Austin (2023).

[28] K. Eriksson, C. Johnson, Adaptive streamline diffusion finite element methods for stationary convection-diffusion problems, Mathematics of Computation 60 (201) (1993) 167–188.

[29] J. Badger, S. Henneking, S. Petrides, L. Demkowicz, Scalable DPG multigrid solver for Helmholtz problems: A study on convergence, arXiv preprint, arXiv:2304.01728, Accepted for publication in Computers & Mathematics with Applications. (2023).