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

Ankit Chakraborty*, Stefan Henneking, Leszek Demkowicz
Oden Institute, The University of Texas at Austin

August 29, 2023

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.

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 [28] 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 [8, 13]. 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 or point and edge singularities in problems with re-entrant corners. In such problems, optimal *hp*-meshes are indispensable for achieving exponential convergence [2, 1, 26, 27, 3]. 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 [26, 24, 14, 25]. 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

*Corresponding author: ankit.chakraborty@austin.utexas.edu

accuracy and stability of the underlying numerical discretization are paramount for the effectiveness of the mesh refinement strategy.

The discontinuous Petrov–Galerkin (DPG) method with optimal test functions, first introduced by Demkowicz and Gopalakrishnan in [10, 9, 11], 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 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 ultraweak DPG method, the energy norm is *equivalent* to the L^2-error [30]. 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 [8, 13]. 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 [9] corresponding to the approximate trial space in such a way that the supremum over the continuous test space in the discrete *inf–sup* [4] 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 [6].

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

$$\begin{cases} \boldsymbol{\sigma} - \nabla u = \mathbf{0} & \text{in} \quad \Omega, \\ -\nabla \cdot \boldsymbol{\sigma} = f & \text{in} \quad \Omega, \\ u = u_0 & \text{on} \quad \Gamma_u, \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} = \sigma_0 & \text{on} \quad \Gamma_\sigma, \end{cases} \tag{2.1}$$

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:

$$\begin{aligned} L^2(\Omega) &= \{u : \Omega \to \mathbb{R} : \|u\| < \infty\}, \\ H^1(\Omega) &= \left\{v : \Omega \to \mathbb{R} : v \in L^2(\Omega), \nabla v \in \left(L^2(\Omega)\right)^3\right\}, \\ \boldsymbol{H}(\mathrm{div}, \Omega) &= \left\{\boldsymbol{w} : \Omega \to \mathbb{R}^3 : \boldsymbol{w} \in (L^2(\Omega))^3, \nabla \cdot \boldsymbol{w} \in L^2(\Omega)\right\}. \end{aligned} \tag{2.2}$$

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):

$$\begin{aligned} H^1(\Omega_h) &:= \left\{v : \Omega \to \mathbb{R} : v|_K \in H^1(K) \quad \forall\, K \in \Omega_h\right\}, \\ \boldsymbol{H}(\mathrm{div}, \Omega_h) &:= \left\{\boldsymbol{w} : \Omega \to \mathbb{R}^3 : \boldsymbol{w}|_K \in \boldsymbol{H}(\mathrm{div}, K) \quad \forall\, K \in \Omega_h\right\}, \end{aligned} \tag{2.3}$$

where $K \in \Omega_h$ represents an element of the finite element mesh. Use of the broken test spaces [6] leads to the introduction of additional trace unknowns on the mesh skeleton. The traces spaces are defined as:

$$\begin{aligned} H^{1/2}(\Gamma_h) &:= \left\{\hat{u} : \exists\, u \in H^1(\Omega) \,\text{such that}\, \hat{u} = \gamma^K(u|_K) \,\text{on}\, \partial K \quad \forall\, K \in \Omega_h\right\}, \\ H^{-1/2}(\Gamma_h) &:= \left\{\hat{\sigma}_n : \exists\, \boldsymbol{\sigma} \in \boldsymbol{H}(\mathrm{div}, \Omega) \,\text{such that}\, \hat{\sigma}_n = \gamma_n^K(\boldsymbol{\sigma}|_K) \,\text{on}\, \partial K \quad \forall\, K \in \Omega_h\right\}, \end{aligned} \tag{2.4}$$

where γ^K and γ_n^K represent continuous and normal trace operators, respectively [15].

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

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

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:

$$b(\mathfrak{u}, \mathfrak{v}) + \hat{b}(\hat{\mathfrak{u}}, \mathfrak{v}) = l(\mathfrak{v}) \quad \forall\, \mathfrak{v} \in V, \tag{2.5}$$

where

$$\begin{aligned}
\mathfrak{u} &= (u, \boldsymbol{\sigma}) \in L^2(\Omega) \times (L^2(\Omega))^3, \\
\hat{\mathfrak{u}} &= (\hat{u}, \hat{\sigma}_n) \in H^{1/2}(\Gamma_h) \times H^{-1/2}(\Gamma_h) : \hat{u} = u_0 \,\text{on}\, \Gamma_u, \hat{\sigma}_n = \sigma_0 \,\text{on}\, \Gamma_\sigma, \\
\mathfrak{v} &= (v, \boldsymbol{\tau}) \in H^1(\Omega_h) \times \boldsymbol{H}(\text{div}, \Omega_h), \\
b(\mathfrak{u}, \mathfrak{v}) &= (\boldsymbol{\sigma}, \nabla v)_{\Omega_h} + (\boldsymbol{\sigma}, \boldsymbol{\tau})_{\Omega_h} + (u, \nabla \cdot \boldsymbol{\tau})_{\Omega_h}, \\
\hat{b}(\hat{\mathfrak{u}}, \mathfrak{v}) &= -\langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n} \rangle_{\Gamma_h} - \langle \hat{\sigma}_n, v \rangle_{\Gamma_h}, \\
l(\mathfrak{v}) &= (f, v)_{\Omega_h}.
\end{aligned} \tag{2.6}$$

In 2.6, $\langle \cdot, \cdot \rangle_{\Gamma_h}$ represents duality pairings defined over mesh skeleton Γ_h,

$$\begin{aligned}
\langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n} \rangle_{\Gamma_h} &:= \sum_{K \in \Omega_h} \langle \hat{u}, \boldsymbol{\tau} \cdot \boldsymbol{n}_K \rangle_{\partial K}\,, \\
\langle \hat{\sigma}_h, v \rangle_{\Gamma_h} &:= \sum_{K \in \Omega_h} \langle \hat{\sigma}_h, v \rangle_{\partial K}\,,
\end{aligned} \tag{2.7}$$

and

$$(\cdot, \cdot)_{\Omega_h} = \sum_{K \,\in\, \Omega_h} (\cdot, \cdot)_{L^2(K)}\,. \tag{2.8}$$

The broken test space is equipped with the adjoint graph norm[17, 7]:

$$\|\mathfrak{v}\|_V^2 := \|A_h^\star \mathfrak{v}\|^2 + \alpha \|\mathfrak{v}\|^2 \tag{2.9}$$

where $\alpha > 0$ is a scaling constant, and $A_h^\star \mathfrak{v} = (\nabla \cdot \boldsymbol{\tau}, \nabla v + \boldsymbol{\tau})_{\Omega_h}$ is the (formal) adjoint operator of $A_h \mathfrak{u} = (\boldsymbol{\sigma} - \nabla u, -\nabla \cdot \boldsymbol{\sigma})_{\Omega_h}$ computed element-wise. In this paper, 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 $V(K)$, and $(U_h, \hat{U}_h) \subset (U, \hat{U})$ 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 2.6, we construct the following matrices for an element $K \in \Omega_h$,

$$\begin{aligned}
\mathrm{G}_{K,lj} &= (\varphi_l, \varphi_j)_V\,, \\
\mathrm{B}_{K,ij} &= b_K(\varphi_i, \psi_j)\,, \\
\hat{\mathrm{B}}_{k,ij} &= \hat{b}_K(\varphi_i, \hat{\psi}_j)\,, \\
\mathrm{l}_{K,i} &= l_K(\varphi_i),
\end{aligned} \tag{2.10}$$

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 $b(\mathfrak{u}, \mathfrak{v}), \hat{b}(\hat{\mathfrak{u}}, \mathfrak{v})$,

and linear form $l(\mathfrak{v})$, 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 [9, 29].

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

$$\|(\mathfrak{u},\hat{\mathfrak{u}})-(\mathfrak{u}_h,\hat{\mathfrak{u}}_h)\|^2_{E,K} := \|{R_V}^{-1}\left(l_K(\cdot)-b_K(\mathfrak{u}_h,\cdot)-\hat{b}_K(\hat{\mathfrak{u}}_h,\cdot)\right)\|^2_{V(K)} \tag{2.11}$$

where

$$R_V : V(K) \to (V(K))' \tag{2.12}$$

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:

$$\eta_K := \|\mathrm{G}^{-1}(\mathrm{l}_K-\mathrm{B}_K\mathrm{u}_\mathrm{h}-\hat{\mathrm{B}}_K\hat{\mathrm{u}}_\mathrm{h})\|^2_{V(K)}\,. \tag{2.13}$$

3 Determining Optimal *hp* Refinements

The *hp*-algorithm described in this section is *exactly* the algorithm from [8, 13], but specialized to the L^2-energy space. The corresponding algorithms for the H^1, $H(\mathrm{curl})$, and $H(\mathrm{div})$ 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 $H(\mathrm{curl})$ 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 [8, 13]. 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*.

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 global reduction over the elements preselected for refinement by the DPG residual.

3.1 Step 1: Staging a Competition of Refinements

In the first step of the algorithm, we stage a competition between *p*- and various anisotropic *h*-refinements, by computing the so-called guaranteed error reduction rate. The comparison between the various candidate refinements is based on the *error reduction rate* (e_{hp}) defined as:

$$e_{hp} := \frac{\|u-P_{\mathrm{old}}u\|^2-\|u-P_{\mathrm{new}}u\|^2}{N_{\mathrm{new}}-N_{\mathrm{old}}}\,, \tag{3.14}$$

[3] Dependent upon the parameter in the Dörfler strategy [18].

[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.

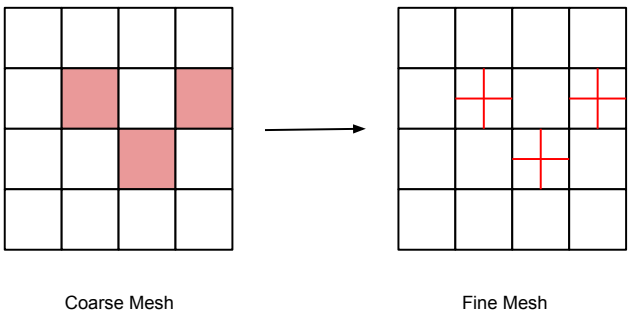


Figure 1: Isotropic *hp*-refinement of the marked elements: the elements marked for refinement are shaded in red on the coarse mesh.

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 the dimensions of the new and old spaces (number of dofs), respectively, 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 hexahedral elements considered in this paper, there are eight possibilities: no h-refinement (i.e. p-refinement only), three anisotropic h_2-refinements, three anisotropic h_4-refinements, and the isotropic h_8-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 for *hp*-refinement. With $p_x, p_y, p_z \in \{1, \ldots, 10\}$, there are "only" 10^3 scenarios for the just p-refined element, but a staggering total of 10^{24} cases for the h_8-refined element. Clearly, a simple search through all possible cases is not feasible. Instead we rely on the classical p-refinement strategy, see e.g. [12], 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 (by one order), 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:

$$\phi_j(x) = \frac{1}{\text{jac}}\hat{\phi}_j(\xi), \qquad \text{jac} = \left|\frac{\partial x_i}{\partial \xi_j}\right|, \tag{3.15}$$

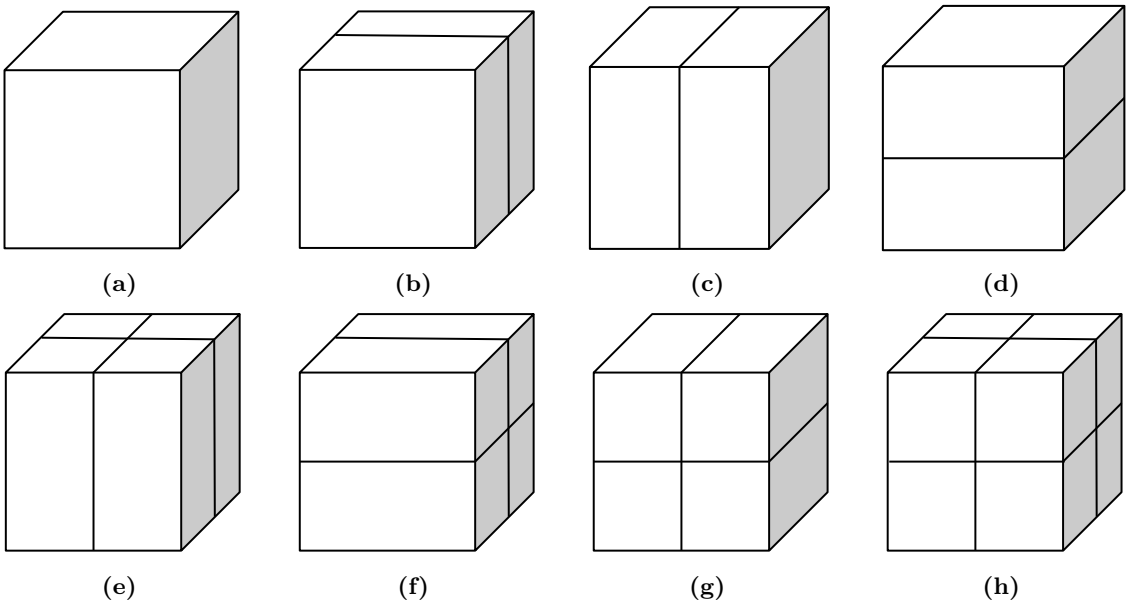


Figure 2: Various possible h-refinements for a hexahedral element, depicted in (a): (b–d) anisotropic h_2-refinements; (e–g) anisotropic h_4-refinements; and (h) isotropic h_8-refinement.

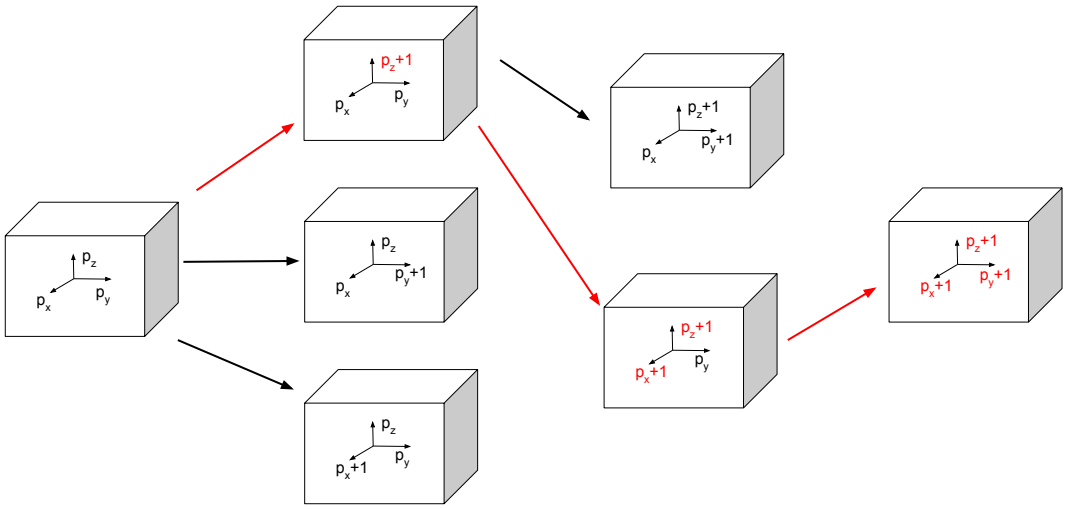


Figure 3: Maximum error reduction path for the p-refined element: traversing from (p_x, p_y, p_z) to $(p_x + 1, p_y + 1, p_z + 1)$ for a hexahedral element.

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,

$$M_{ij} := \int_K \phi_i \phi_j \, dx = \frac{1}{\text{jac}} \int_{\hat{K}} \hat{\phi}_i \hat{\phi}_j \, d\xi \,, \tag{3.16}$$

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:

$$P_N u = \sum_{j=1}^{N} u_j \phi_j, \qquad u_j = \frac{1}{M_{jj}} \int_K u \phi_j \, dx \,. \tag{3.17}$$

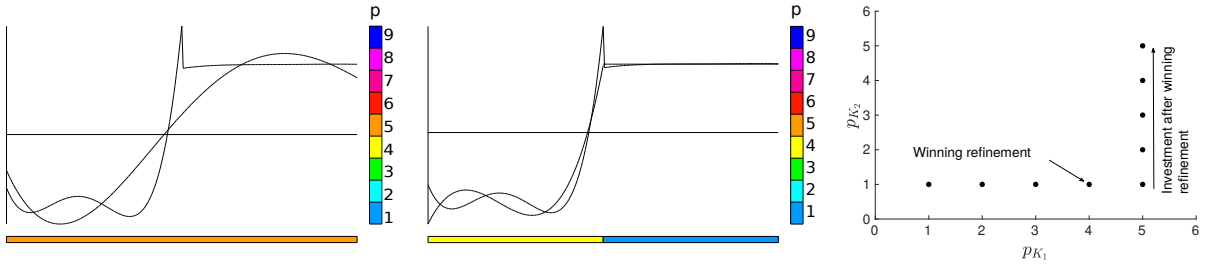
Raising the polynomial order in one direction amounts to adding extra orthogonal shape functions ϕ_{N+l} with $l = 1, ..., n$. Consequently, evaluation of the error reduction rate reduces to:

$$\begin{aligned} \frac{\|u - P_N u\|^2 - \|u - P_{N+1} u\|^2}{n} &= \frac{\|P_{N+1} u\|^2 - \|P_N u\|^2}{n} \\ &= \frac{1}{n} \sum_{l=1}^{n} |u_{N+l}|^2 M_{N+l,N+l} = \frac{1}{n} \sum_{l=1}^{n} \left(\frac{\int_K u \phi_{N+l} \, dx}{M_{N+l,N+l}} \right)^2 M_{N+l,N+l} \\ &= \frac{1}{n} \sum_{l=1}^{n} M^{-1}_{N+l,N+l} \left(\int_K u \phi_{N+l} \, dx \right)^2 . \end{aligned} \tag{3.18}$$

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 [13, 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.

Figure 4 shows the simple case of a 1D element K, starting with polynomial order $p_K = 4$. The subelements of the h-refined element K are denoted K_1 and K_2, and their respective polynomial orders p_{K_1} and p_{K_2}. The maximum error reduction path for this case (illustrated in Figure 4c) leads to the winning refinement $(p_{K_1}, p_{K_2}) = (4, 1)$ with the approximate solution shown in Figure 4b.



(a) Purely p-refined element:
$p_K = 5$
$e_{hp} = 0.01952$

(b) Winning h-refined element:
$(p_{K_1}, p_{K_2}) = (4, 1)$
$e_{hp} = 0.23196$

(c) Refinement path of the h-refined element

Figure 4: Staging a competition between the p-refined and h-refined element. The maximum error reduction path for the h-refined element traverses from $(p_{K_1}, p_{K_2}) = (1, 1)$ to the winning refinement $(p_{K_1}, p_{K_2}) = (4, 1)$.

The optimal refinement. The selection of the optimal refinement is carried out by comparing the best error reduction rates delivered by the eight differently h-refined meshes. The highest error reduction rate, delivered by the optimal refinement, is called the *guaranteed error reduction rate* and denoted by e^*_{hp}.

3.2 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* $e^*_{hp,\max}$. In principle, one could then refine only this one element. However, to accelerate the refinements (i.e. reduce the number of refinement steps), a greedy strategy is employed selecting all elements that deliver a rate greater than or equal to 25% of the best guaranteed error reduction rate. Note that this strategy implies that there may be elements initially marked for refinement by the DPG residual which ultimately remain unrefined.

3.3 Step 3: Determining the Final Refinements

For each element selected for refinement in Step 2, we could simply execute the corresponding optimal refinement determined in Step 1; and we do this indeed for the purely p-refined elements. However, when performing h-refinements we typically choose to invest additional dofs by considering the already-performed p-refinements that followed the optimal refinement while investigating error reduction rates in Step 1.

In particular, in Step 1 we recorded the error reduction rates for all subelement meshes following the maximum error reduction path. On this path, we select the maximum investment (in terms of new dofs) that still delivers 25% of the best guaranteed error reduction rate (meaning it would still satisfy the Step 2 criterion). The rational for doing so is to reduce the overall number of outer-loop iterations (number of refinement steps) by maximizing the investment in each step as long as it pays off sufficiently (delivering a sufficiently high error reduction rate, as determined by Step 2).

For example, in the 1D case illustrated in Figure 4, the refinement shown in Figure 4b won the competition with the p-refinement (Figure 4a) but, dependent upon the threshold value used in the greedy strategy, we may choose to invest additional dofs in one of the subelements.

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

3.4 Mesh Closure

The *hp* algorithm is implemented in *hp*3D, a general-purpose finite element 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 [22, 23]. *hp*3D supports MPI/OpenMP parallelism [21] and is available under BSD-3 license.[5] In the code, any 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.

Closure 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 ([9, p. 71]) to ensure that no face is refined unless the face[6] is unconstrained.

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

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

Algorithm 1 Mesh Optimization Algorithm

Start with an initial trial mesh
while $\eta_{\Omega_h} >$ 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 ($e^*_{hp,K}$).
    Step 2: Determine the best guaranteed error reduction rate ($e^*_{hp,\max}$).
    Unrefine the mesh.
    Step 3: For each element K marked for refinement:
    if $e^*_{hp,K} \geq 0.25\, e^*_{hp,\max}$ **then**
        Perform the optimal *hp*-refinement.
    end if
end while

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).

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

We refer to [22] for a more formal exposition of the algorithms. In the end, in both steps, a number of additional, *unwanted* refinements may be 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 retain their respective polynomial order. In principle, one could attempt to find the corresponding optimal distribution of polynomial orders, but this has been not done in our current implementation. Hence, the presented meshes may be slightly overrefined.

4 Numerical Results

4.1 A Boundary Layer Problem

Sharp boundary layers are among the most commonly encountered flow features in computational fluid dynamics. Our first 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 [19]. In particular, we solve

$$\begin{aligned} -\nabla^2 u &= f(x,y,z) \quad \text{in} \quad \Omega := (0,1)^3, \\ u &= 0 \qquad\qquad\ \ \text{on} \quad \Gamma_u, \\ \nabla u \cdot \boldsymbol{n} &= g(x,y,z) \quad \text{on} \quad \Gamma_\sigma, \end{aligned} \tag{4.19}$$

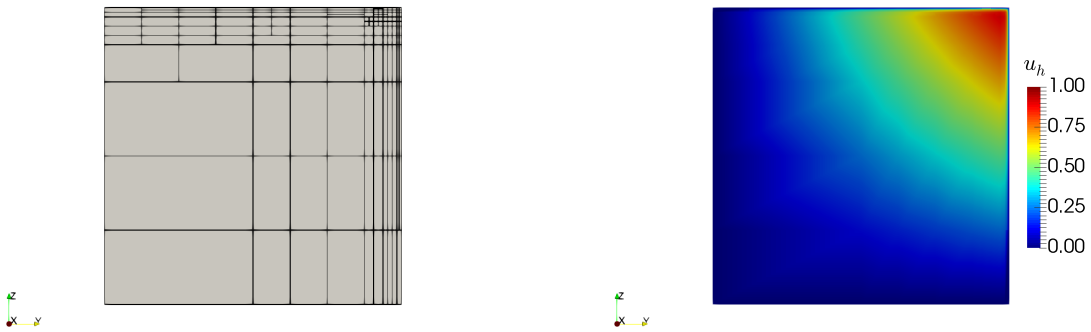
where

$$\begin{aligned} \Gamma_u &= ([0,1)\times[0,1)\times\{0\}) \cup ([0,1)\times\{0\}\times[0,1)) \cup (\{0\}\times[0,1)\times[0,1))\,, \\ \Gamma_\sigma &= ([0,1]\times[0,1]\times\{1\}) \cup ([0,1]\times\{1\}\times[0,1]) \cup (\{1\}\times[0,1]\times[0,1])\,. \end{aligned} \tag{4.20}$$

In (4.19), $\boldsymbol{n}$ is the outward normal, and f and g are generated using the exact solution. The exact solution is given by

$$u(x,y,z) = \left(x + \frac{e^{x/\epsilon}-1}{1-e^{1/\epsilon}}\right)\left(y + \frac{e^{y/\epsilon}-1}{1-e^{1/\epsilon}}\right)\left(z + \frac{e^{z/\epsilon}-1}{1-e^{1/\epsilon}}\right). \tag{4.21}$$

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]



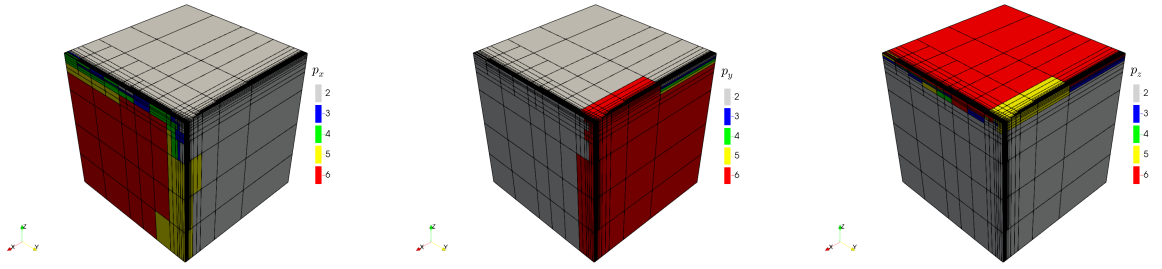
(a) Cross-section of the mesh at $x = 0.95$ **(b)** Contour plot of the solution at $x = 0.95$

Figure 5: 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.

Figures 5a and 5b display the cross-section of an adapted mesh and the corresponding solution contour, respectively. Figure 6 depicts the polynomial distribution around the boundary layers on

[7] In hp3D, we employ exact-sequence spaces [16]. Hence, an order of (p_x, p_y, p_z) denotes L^2 shape functions of order $(p_x - 1, p_y - 1, p_z - 1)$.

an anisotropically adapted *hp*-mesh. Figure 7 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 7, the depicted error is the combined relative error in all L^2 variables.



(a) Polynomial order p_x **(b)** Polynomial order p_y **(c)** Polynomial order p_z

Figure 6: Boundary layer problem: an adapted mesh with 855 532 dofs; coloring indicates the polynomial distributions p_x, p_y, p_z in x-, y-, z-direction, respectively. The algorithm prescribes higher-order polynomials anisotropically corresponding to each boundary layer along the x-, y-, and z-axis.

Figure 6 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 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 7, it is evident that anisotropic *hp*-refinements outperform isotropic *h*-refinements by orders of magnitude. The convergence plots show the error and the residual against $\sqrt[3]{\text{ndof}}$ (where ndof represents the number of degrees of freedom), verifying exponential convergence. In Figure 7, 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 strategy. The variant of the Fichera cube problem being solved here is given by:

$$\begin{aligned} \nabla^2 u &= 0 && \text{in} \quad \Omega := (-1,1)^3 \setminus [0,1]^3, \\ u &= 0 && \text{on} \quad \Gamma_u, \\ \nabla u \cdot \boldsymbol{n} &= g(x,y,z) && \text{on} \quad \Gamma_\sigma. \end{aligned} \tag{4.22}$$

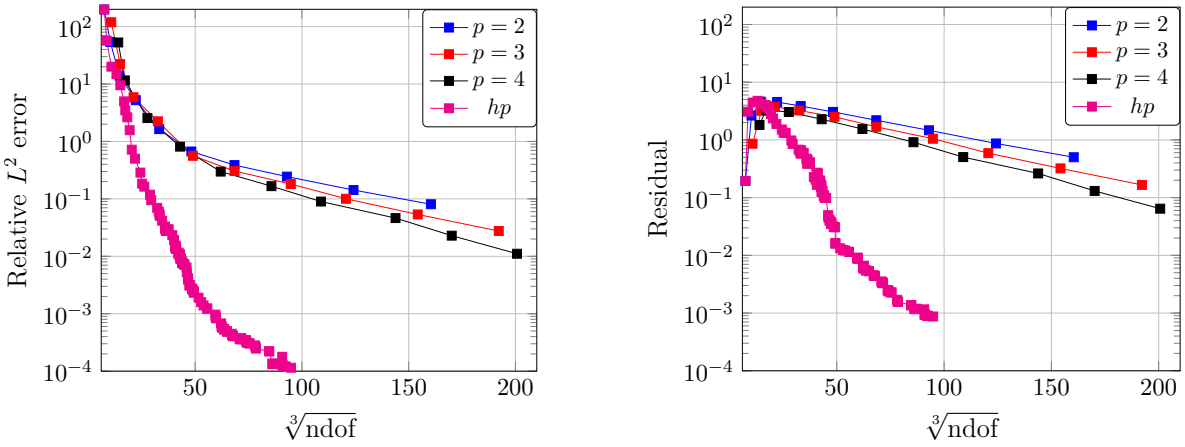


Figure 7: Boundary layer problem: convergence of relative L^2 error and DPG residual. Even though there is a marginal decrease in the rate of convergence for the hp-refinements, both the error and the residual are 2–3 orders of magnitude lower compared to the h-refinements for approximately the same number of dofs.

The domain is created by subdividing a large cube $(-1, 1)^3$ 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 of coordinate axes, i.e.

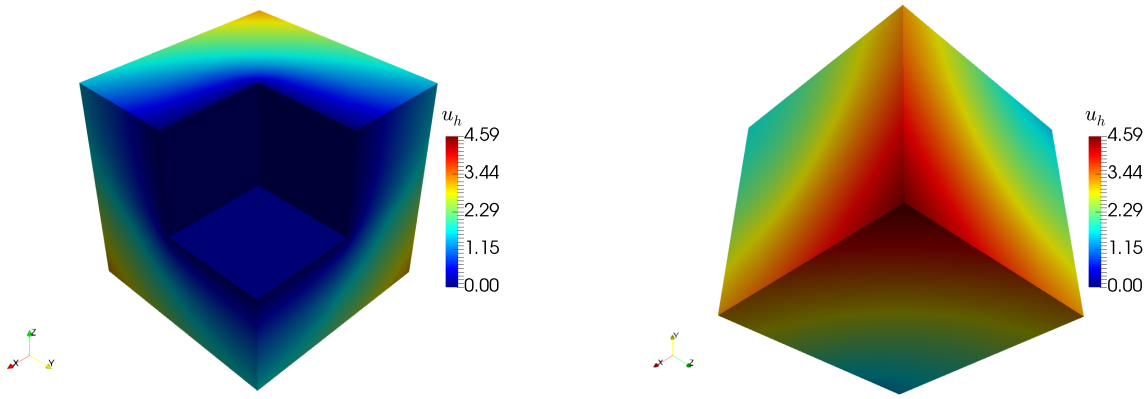
$$\Gamma_u = ([0,1] \times [0,1] \times \{0\}) \cup ([0,1] \times \{0\} \times [0,1]) \cup (\{0\} \times [0,1] \times [0,1]) \,. \tag{4.23}$$

The volumetric load for the problem is zero. The problem is driven by the Neumann boundary condition on Γ_σ composed of the remaining faces of the cube. The data g correspond 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:

$$u_{\eta,\xi} = r^{\frac{2}{3}} cos(\theta), \quad r = \sqrt{\eta^2 + \xi^2}, \quad \theta = \tan^{-1}\left(\frac{\xi}{\eta}\right), \tag{4.24}$$

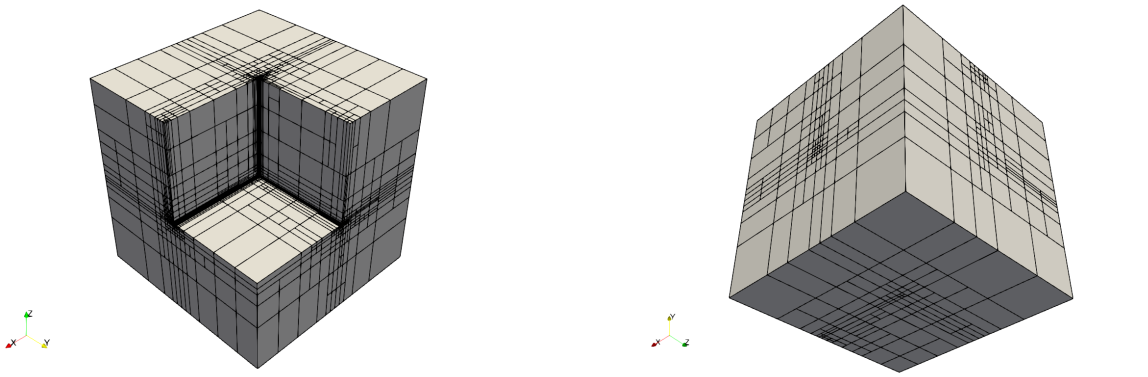
where (η, ξ) denote (x, y), (y, z), or (x, z) axes, respectively. These boundary conditions generate a solution with features analogous to an L-shaped domain problem but comprising multiple edge and vertex singularities. While the exact solution for the problem is unknown, the convergence of the DPG residual is shown in Figure 12.

Figures 8 and 9 depict the solution contour and the corresponding adapted mesh, respectively. Figures 10 and 11 illustrate the polynomial distribution associated with the adapted mesh. Figure 9 shows that the refinement algorithm performs highly anisotropic h-refinements along the edge singularities, generating 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. Figures 10 and 11 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 12, 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.



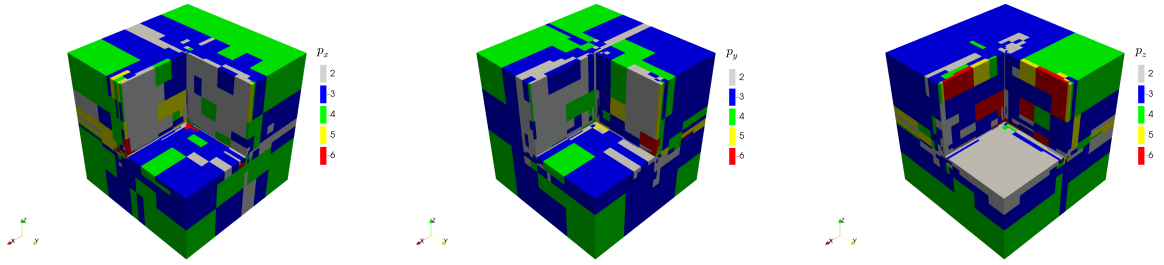
(a) Isometric view along $(-1,-1,-1)$ **(b)** Isometric view along $(1,1,1)$

Figure 8: 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.



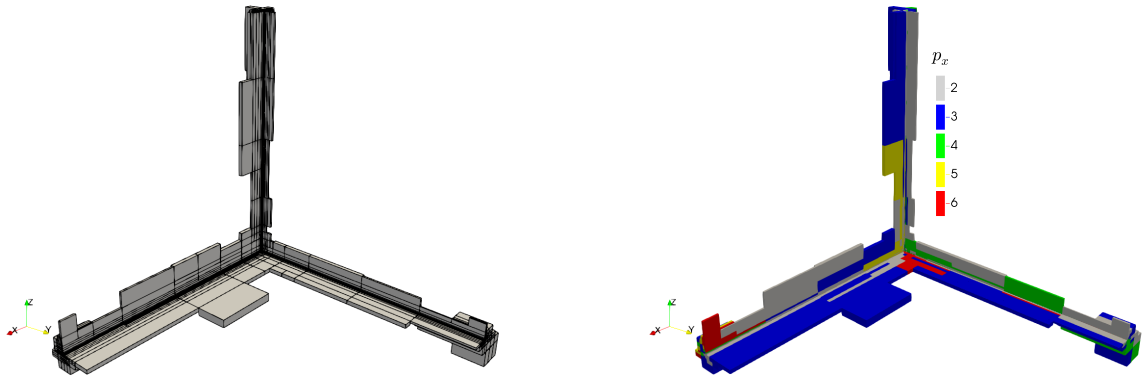
(a) Isometric view along $(-1,-1,-1)$ **(b)** Isometric view along $(1,1,1)$

Figure 9: Fichera cube problem: an anisotropically adapted *hp*-mesh with 1.3M dofs.



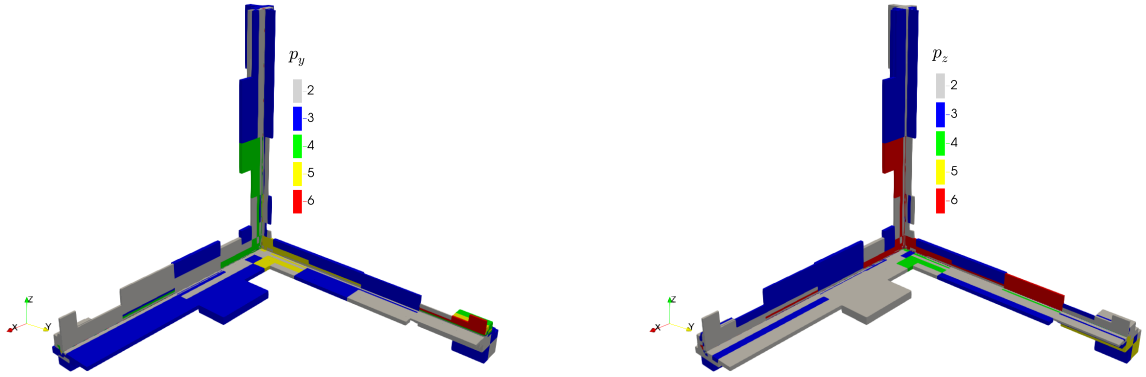
(a) Polynomial order p_x **(b)** Polynomial order p_y **(c)** Polynomial order p_z

Figure 10: 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 11 presents a magnified view of the polynomial distribution and anisotropic mesh elements around the singularities.



(a) Adapted mesh

(b) Polynomial order p_x



(c) Polynomial order p_y

(d) Polynomial order p_z

Figure 11: Fichera cube problem: magnified view of the mesh and the polynomial distribution near the edge and vertex singularities.

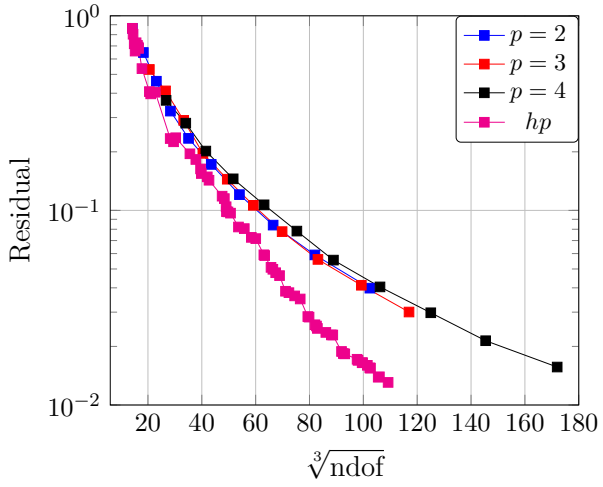


Figure 12: Fichera cube problem: convergence of the DPG residual.

4.3 Eriksson–Johnson Problem

We consider a convection-dominated diffusion problem motivated by the Eriksson–Johnson model problem [20]. Here, we extend the exact solution of the two-dimensional problem by multiplying it

with a sinusoidal term along z. In particular, we solve

$$\begin{aligned} \frac{\partial u}{\partial x} - \epsilon \nabla^2 u &= f(x,y,z) && \text{in} \quad \Omega := (0,1)^3, \\ u &= 0 && \text{on} \quad \Gamma_{u_a}, \\ u &= \sin(\pi y)\sin(\pi z) && \text{on} \quad \Gamma_{u_b}, \end{aligned} \tag{4.25}$$

where

$$\Gamma_{u_a} = \partial\Omega \setminus \{0\} \times [0,1] \times [0,1] \quad \text{and} \quad \Gamma_{u_b} = \{0\} \times [0,1] \times [0,1]. \tag{4.26}$$

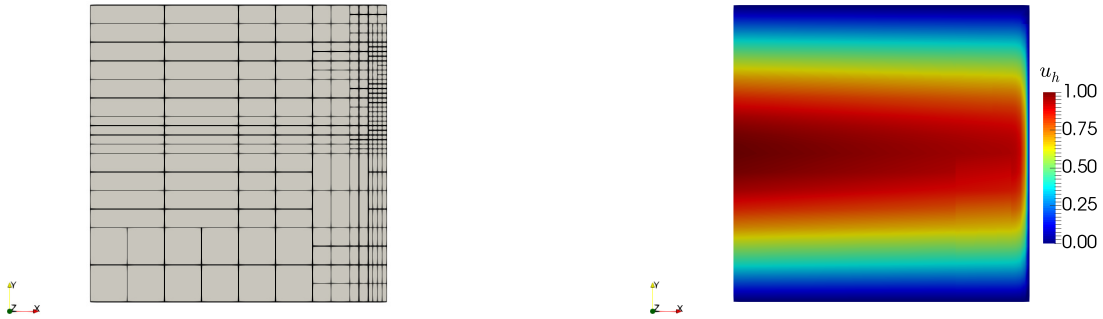
The source f and the boundary conditions are computed using the exact solution given by

$$u(x,y,z) = \frac{e^{s_1(x-1)} - e^{s_2(x-1)}}{e^{s_1} - e^{s_2}} \sin(\pi y)\sin(\pi z), \tag{4.27}$$

where

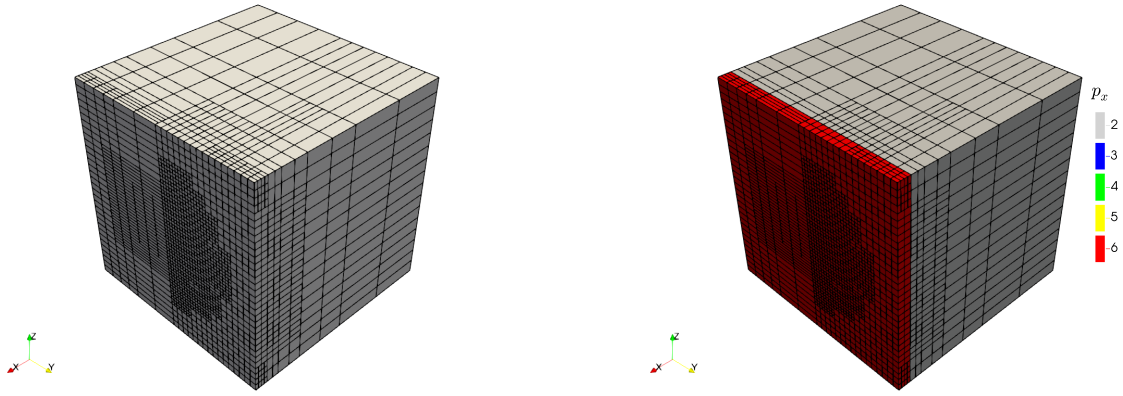
$$s_1 = \frac{1 + \sqrt{1 + 4\pi^2\epsilon^2}}{2\epsilon} \text{ and } s_2 = \frac{1 - \sqrt{1 + 4\pi^2\epsilon^2}}{2\epsilon}. \tag{4.28}$$

In this numerical experiments, $\epsilon = 0.01$. Figure 13 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 14 illustrates the adapted mesh with the polynomial distribution along the x-axis. Finally, Figure 15 presents the convergence plots for the relative L^2 error and the residual, demonstrating the efficacy of the proposed hp-refinement strategy for this problem.



(a) Cross-section of an adapted mesh at $z = 0.5$ **(b)** Solution contour at $z = 0.5$

Figure 13: Eriksson–Johnson problem: an adapted mesh and solution contour.



(a) Isometric view of the mesh.

(b) Polynomial order along x direction: p_x

Figure 14: Eriksson–Johnson problem: an adapted mesh with 209 737 dofs; coloring indicates the corresponding polynomial distribution along the x-axis.

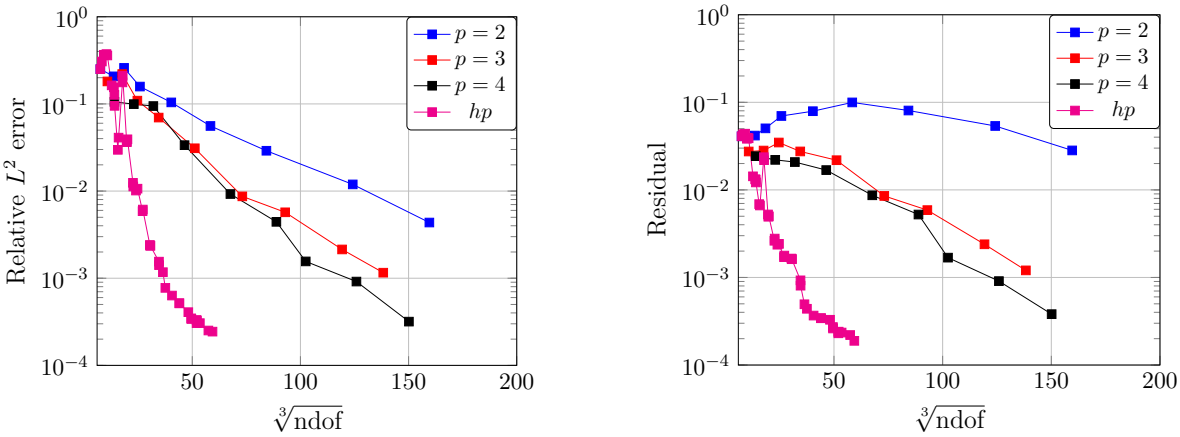


Figure 15: Eriksson–Johnson problem: convergence of relative L^2 error and DPG residual.

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 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 scalable DPG-MG solver [5]. 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] I. Babuška and W. Gui. "The h, p and *hp*-versions of the finite element method in 1 Dimension. Part III. The adaptive *hp*-version". In: *Numerische Mathematik* 49 (1986), pp. 659–684.

[2] I. Babuška and B.Q. Guo. "The h, p and *hp*-version of the finite element method; basis theory and applications". In: *Advances in Engineering Software* 15.3 (1992), pp. 159–174. ISSN: 0965-9978.

[3] I. Babuška, T. Strouboulis, and K. Copps. "*hp* Optimization of finite element approximations: Analysis of the optimal mesh sequences in one dimension". In: *Computer Methods in Applied Mechanics and Engineering* 150.1 (1997), pp. 89–108.

[4] Ivo Babuška. "Error-bounds for finite element method". In: *Numerische Mathematik* 16 (1971), pp. 322–333.

[5] Jacob Badger et al. "Scalable DPG Multigrid Solver for Helmholtz Problems: A Study on Convergence". In: *Computers & Mathematics with Applications* 148 (2023), pp. 81–92.

[6] C. Carstensen, L. Demkowicz, and J. Gopalakrishnan. "Breaking spaces and forms for the DPG method and applications including Maxwell equations". In: *Computers & Mathematics with Applications* 72.3 (2016), pp. 494–522.

[7] Jesse Chan et al. "A robust DPG method for convection-dominated diffusion problems II: Adjoint boundary conditions and mesh-dependent test norms". In: *Computers & Mathematics with Applications* 67.4 (2014), pp. 771–795.

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

[9] L. Demkowicz and J. Gopalakrishnan. "A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions". In: *Numerical Methods for Partial Differential Equations* 27.1 (2011), pp. 70–105.

[10] L. Demkowicz and J. Gopalakrishnan. "A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation". In: *Computer Methods in Applied Mechanics and Engineering* 199.23 (2010), pp. 1558–1572.

[11] L. Demkowicz, J. Gopalakrishnan, and A.H. Niemi. "A class of discontinuous Petrov–Galerkin methods. Part III: Adaptivity". In: *Applied Numerical Mathematics* 62.4 (2012), pp. 396–427.

[12] L. Demkowicz, J. T. Oden, and T. Strouboulis. "Adaptive Finite Elements for Flow Problems with Moving Boundaries. Part 1: Variational Principles and a Posteriori Estimates". In: *Computer Methods in Applied Mechanics and Engineering* 46 (1984), pp. 217–251.

[13] L Demkowicz et al. *Computing with hp-Adaptive Finite Elements. Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman and Hall/CRC, 2007.

[14] L. Demkowicz et al. "Toward a universal *hp* adaptive finite element strategy. Part 1: constrained approximation and data structure". In: *Computer Methods in Applied Mechanics and Engineering* 77.1 (1989), pp. 79–112.

[15] Leszek Demkowicz. *Energy Spaces*. Lecture notes; The University of Texas at Austin. 2018.

[16] Leszek Demkowicz. *Mathematical theory of finite elements*. Lecture notes; The University of Texas at Austin. 2023.

[17] Leszek Demkowicz and Norbert Heuer. "Robust DPG Method for Convection-Dominated Diffusion Problems". In: *SIAM Journal on Numerical Analysis* 51.5 (2013), pp. 2514–2537.

[18] Willy Dörfler. "A Convergent Adaptive Algorithm for Poisson's Equation". In: *SIAM Journal on Numerical Analysis* 33.3 (1996), pp. 1106–1124. (Visited on 08/01/2023).

[19] Herbert Egger and Joachim Schöberl. "A hybrid mixed discontinuous Galerkin finite element method for convection–diffusion problems". In: *IMA Journal of Numerical Analysis* 30.4 (July 2009), pp. 1206–1234. ISSN: 0272-4979.

[20] Kenneth Eriksson and Claes Johnson. "Adaptive Streamline Diffusion Finite Element Methods for Stationary Convection-Diffusion Problems". In: *Mathematics of Computation* 60.201 (1993), pp. 167–188. (Visited on 07/17/2023).

[21] Stefan Henneking. "A scalable *hp*-adaptive finite element software with applications in fiber optics". PhD thesis. The University of Texas at Austin, 2021.

[22] Stefan Henneking and Leszek Demkowicz. *Computing with hp Finite Elements. III. Parallel hp3D Code*. In preparation, 2023.

[23] Stefan Henneking and Leszek Demkowicz. "*hp*3D User Manual". In: *arXiv:2207.12211* (2022).

[24] J.T. Oden et al. "Toward a universal *hp* adaptive finite element strategy. Part 2: a posteriori error estimation". In: *Computer Methods in Applied Mechanics and Engineering* 77.1 (1989), pp. 113–180.

[25] W. Rachowicz, J.T. Oden, and L. Demkowicz. "Toward a universal *hp* adaptive finite element strategy. Part 3: design of *hp* meshes". In: *Computer Methods in Applied Mechanics and Engineering* 77.1 (1989), pp. 181–212.

[26] W. Rachowicz, D. Pardo, and L. Demkowicz. "Fully automatic *hp*-adaptivity in three dimensions". In: *Computer Methods in Applied Mechanics and Engineering* 195.37 (2006), pp. 4816–4842.

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

[28] Dan Stanzione et al. "Frontera: The Evolution of Leadership Computing at the National Science Foundation". In: *Practice and Experience in Advanced Research Computing*. Association for Computing Machinery, 2020, pp. 106–111.

[29] Ali Vaziri Astaneh et al. "High-order polygonal discontinuous Petrov–Galerkin (PolyDPG) methods using ultraweak formulations". In: *Computer Methods in Applied Mechanics and Engineering* 332 (2018), pp. 686–711.

[30] J. Zitelli et al. "A class of discontinuous Petrov–Galerkin methods. Part IV: the optimal test norm and time-harmonic wave propagation in 1D". In: *Journal of Computational Physics* 230.7 (2011), pp. 2406–2432.