Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 278 (2014) 20–40
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A PDE-constrained optimization approach to the discontinuous Petrov–Galerkin method with a trust region inexact Newton-CG solver

Tan Bui-Thanh[a,*], Omar Ghattas[b]

[a] *Department of Aerospace Engineering and Engineering Mechanics, Institute for Computational Engineering & Sciences, The University of Texas at Austin, Austin, TX 78712, USA*
[b] *Institute for Computational Engineering & Sciences, Jackson School of Geosciences, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, USA*

Received 5 October 2013; received in revised form 21 April 2014; accepted 26 April 2014
Available online 6 May 2014

Abstract

We propose a partial-differential-equation-constrained (PDE-constrained) approach to the discontinuous Petrov–Galerkin (DPG) of Demkowicz and Gopalakrishnan (2010, 2011). This view opens the door to invite all the state-of-the-art PDE-constrained techniques to be part of the DPG framework, and hence enabling one to solve large-scale and difficult (nonlinear) problems efficiently. That is, our proposed method preserves all the attractive features of the DPG framework while enjoying all advances from the PDE-constrained optimization community. The proposed approach can be considered as a Rayleigh–Ritz method for the DPG minimum residual statement. It is equipped with a trust region inexact Newton conjugate gradient (TRINCG) method which prevents over-solving when optimization iterates are far away from the optimal solution but converges quadratically otherwise for sufficiently smooth residual. The PDE-constrained approach together with the TRINCG solver is therefore a robust iterative method for the DPG framework. It is robust in the sense that the approximate solution is improved after each optimization iterate until the algorithm converges to a local minimum of the residual. As numerically shown, it is also scalable in the sense that the number of Newton iterations remains constant as either the mesh or the solution order is refined. Moreover, the proposed approach solves neither the optimal test functions nor the original PDE directly, though the discretization of the latter is still necessary. The gradient and Hessian-vector product, which are required by the TRINCG solver, are explicitly derived for both abstract variational problems and viscous Burger equation. This reveals a fact that the complexity of each Newton iteration scales like $\mathcal{O}(N \times n_{CG})$, where N and n_{CG} are the number of unknowns and the number of conjugate iterations, respectively. Optimal h- and p- convergences of the proposed approach are demonstrated for Laplace and Helmholtz equations. Numerical results for the viscous Burger equation and the Euler equation of gas dynamics are very promising.

Keywords: Discontinuous Petrov–Galerkin methods; Trust region inexact Newton conjugated gradient method; Forward equation; Adjoint equation; Incremental forward equation; PDE-constrained optimization

* Corresponding author. Tel.: +1 6178202676.
E-mail addresses: tanbui@ices.utexas.edu, tansweet@gmail.com (T. Bui-Thanh), omar@ices.utexas.edu (O. Ghattas).

http://dx.doi.org/10.1016/j.cma.2014.04.018
0045-7825/© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The discontinuous Petrov–Galerkin (DPG) framework of Demkowicz and Gopalakrishnan [1,2] has been emerging as a new numerical methodology for partial differential equations (PDEs). There have been successful attempts to apply the DPG framework to a wide range of PDEs including scalar transport [1–3], Laplace [4], convection–diffusion [2,4], Helmholtz [5–7], Burgers and Navier–Stokes [8], and linear elasticity [9] equations. The recent work in [10] has also proposed an abstract DPG framework for Friedrichs' system of PDEs that unifies most of the existing DPG methods. In particular, the abstract DPG is well-posed, namely the existence, uniqueness, and stability of solution are guaranteed. It is important to point out that there are related works on the DPG method in [11–13], but we will focus on the DPG method of Demkowicz and Gopalakrishnan [1,2].

By construction, DPG methods appear to depend on the mesh. Indeed, a DPG method starts by partitioning the domain of interest into non-overlapping elements. Variational formulations are posed for each element separately and then summed up to form a global variational statement. Elemental solutions are connected by hybrid variables, also known as fluxes or traces, that live on the skeleton of the mesh. Thus, the DPG variational formulation, particularly the number of DPG unknowns, depends on the mesh under consideration. Nevertheless, it can be shown that the DPG inf–sup constant is independent of the mesh (see, e.g., [14,10]). As a result, the DPG stability does not deteriorate as the mesh is refined.

The DPG method is a minimum residual method and can be viewed as a generalization of least squares approaches [15,16]. This suggests that one can equip the DPG method with the Euler–Lagrange approach in which one solves the optimality equation, under which the residual is minimized, for the DPG solution. The optimality condition can be solved by direct solvers, which is the approach taken by Demkowicz et al. [1,2,4–9,14,17], or by iterative ones [18,19].

In this paper, we propose a Rayleigh–Ritz approach to the DPG framework. In particular, we pose the DPG minimum residual statement as an equivalent PDE-constrained optimization problem. This opens the door to invite all the state-of-the-art PDE-constrained optimization techniques to be part of the DPG framework, and hence enabling us to solve large-scale and difficult (nonlinear) problems efficiently. To this end, we first review the standard DPG unconstrained minimum residual approach, particularly tailored to the viscous Burger equation in Section 2. The details of a PDE-constrained minimum residual approach to the DPG method is presented in Section 3. We shall show that no matter how complicated and/or nonlinear the original first order PDE under consideration is the forward equation is local, i.e. can be solved element-by-element, and is always a linear second order elliptic PDE dictated by the norm in test space. It turns out that we do not need to solve the original PDE directly though we still have to discretize it. In fact, its resulting discretized residual is the forcing for the forward equation in the PDE-constrained framework. We derive the gradient and Hessian-vector product explicitly since they are necessary for a trust region inexact Newton conjugate gradient method that is described in detail in Section 8. We also show that the adjoint equation is trivial; particularly, the adjoint and forward solutions are the negative of each other. That is, the adjoint equation can be eliminated explicitly.

Next, a connection between our PDE-constrained DPG and the standard unconstrained DPG methods will be studied in Section 4. *We shall show that the former can be viewed as an iterative solver for the DPG framework*. In Section 5 we explicitly derive the gradient and Hessian-vector production for the Burger equation. We also briefly discuss how to straightforwardly and approximately enforce conservation within our optimization framework, and this is done in Section 6. Section 7 presents a conforming discretization of the PDE-constrained DPG formulation. *A main result of this section is that within our PDE-constrained DPG method the commutation of the optimize-then-discretize and the discretize-then-optimize approaches is trivially satisfied*. This is desirable (see Remark 2), but typically feasible only for elliptic PDEs, in the PDE-constrained optimization literature. Section 8 lays out a state-of-the-art trust region inexact Newton conjugate gradient solver for our DPG method, and Section 9 presents numerical results for Laplace, Helmholtz, viscous Burger, and Euler equations. Finally, the paper is concluded in Section 10.

2. The discontinuous Petrov–Galerkin method with optimal test functions (DPG)

In this section, let us introduce the DPG framework by constructing a DPG method for the following nonlinear two-dimensional[1] viscous Burger equation [20]

[1] Extension to three-dimensional cases is straightforward.

$$\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial u}{\partial y} = \varepsilon\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad \text{in } \Omega, \tag{1}$$

with inflow boundary condition $u = u_-$ on Γ^- and outflow boundary $\Gamma^+ := \partial\Omega \setminus \Gamma^-$. Here, ε is the viscosity coefficient, which can be zero, and Ω a bounded open subset of $\mathbb{R}^2$ with Lipschitz boundary. We first convert (1) into the first order system of partial differential equations (PDEs)

$$\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial u}{\partial y} - \varepsilon\nabla\cdot\mathbf{q} = 0 \quad \text{in } \Omega, \tag{2a}$$

$$\mathbf{q} - \nabla u = 0 \quad \text{in } \Omega. \tag{2b}$$

Then, we test (2a) with v, (2b) with $\boldsymbol{\tau}$, and add the resulting equations to have

$$\left(\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial u}{\partial y} - \varepsilon\nabla\cdot\mathbf{q}, v\right)_\Omega + (\mathbf{q} - \nabla u, \boldsymbol{\tau})_\Omega = 0, \tag{3}$$

where we have introduced the notation $(\cdot,\cdot)_K$ as the L^2-inner product of two (possibly vector-valued) functions on a given domain K. Now, let us partition the domain Ω into N_{el} non-overlapping elements K_j, $j = 1, \ldots, N_{\text{el}}$ with Lipschitz boundaries such that $\Omega_h := \cup_{j=1}^{N_{\text{el}}} K_j$ and $\overline{\Omega} = \overline{\Omega}_h$. Here, h is defined as $h := \max_{j\in\{1,\ldots,N_{\text{el}}\}} diam\left(K_j\right)$. We denote the skeleton of the mesh by $\Gamma_h := \cup_{j=1}^{N_{\text{el}}} \partial K_j$; the set of all (uniquely defined) faces/edges e, each of which comes with a normal vector $\mathbf{n}^e = \left(n_x^e, n_y^e\right)$. For the sake of convenience, we also define $\Gamma_h^- := \Gamma_h \setminus \Gamma^+$, $\Gamma_{K_j}^+ := \partial K_j \setminus \Gamma^-$ and similarly $\Gamma_{K_j}^- := \partial K_j \setminus \Gamma^+$.

Now, integrating (3) by parts we have

$$\begin{aligned}
&\sum_{K_j} -\frac{1}{2}\left(u^2, \frac{\partial v}{\partial x}\right)_{K_j} - \left(u, \frac{\partial v}{\partial y}\right)_{K_j} + \varepsilon(\mathbf{q}, \nabla v)_{K_j} + \left\langle \frac{1}{2}u^2 n_x + u n_y, v\right\rangle_{\Gamma_{K_j}^+} \\
&\quad - \varepsilon\left\langle \mathbf{q}\cdot\mathbf{n}, v\right\rangle_{\Gamma_{K_j}} + (\mathbf{q}, \boldsymbol{\tau})_{K_j} + (u, \nabla\cdot\boldsymbol{\tau})_{K_j} - \left\langle u, \boldsymbol{\tau}\cdot\mathbf{n}\right\rangle_{\Gamma_{K_j}^+} \\
&\quad = -\left\langle \frac{1}{2}u_-^2 n_x + u_- n_y, v\right\rangle_{\Gamma_{K_j}^-} + \left\langle u_-, \boldsymbol{\tau}\cdot\mathbf{n}\right\rangle_{\Gamma_{K_j}^-},
\end{aligned} \tag{4}$$

where $\langle\cdot,\cdot\rangle_\Gamma$ denotes the duality pairing on a given set Γ with nontrivial one-dimensional Lebesgue measure. The next step in the DPG methodology is to seek solutions $(u, \mathbf{q})$ with $u, u^2 \in L^2(\Omega)$ and[2] $\mathbf{q} \in L^2(\Omega)$. If we do that, the trace of u and numerical flux $\mathbf{q}\cdot\mathbf{n}$ on the element boundary are, however, not meaningful. To avoid this undesirable fact, let us introduce single-valued hybrid variables $\hat{u}$ and $\hat{q}$ on the mesh skeleton.

Next, we seek $\left(u, \mathbf{q}, \hat{u}, \hat{q}\right) \in U := L^2(\Omega) \times L^2(\Omega) \times H^{\frac{1}{2}}\left(\Gamma_h^-\right) \times H^{-\frac{1}{2}}(\Gamma_h)$ such that

$$\begin{aligned}
&\sum_{K_j} -\frac{1}{2}\left(u^2, \frac{\partial v}{\partial x}\right)_{K_j} - \left(u, \frac{\partial v}{\partial y}\right)_{K_j} + \varepsilon(\mathbf{q}, \nabla v)_{K_j} + \left\langle \frac{1}{2}\hat{u}^2 n_x + \hat{u} n_y, v\right\rangle_{\Gamma_{K_j}^+} \\
&\quad - \varepsilon\left\langle \hat{q}\,\mathrm{sgn}(\mathbf{n}), v\right\rangle_{\Gamma_{K_j}} + (\mathbf{q}, \boldsymbol{\tau})_{K_j} + (u, \nabla\cdot\boldsymbol{\tau})_{K_j} - \left\langle \hat{u}, \boldsymbol{\tau}\cdot\mathbf{n}\right\rangle_{\Gamma_{K_j}^+} = \left\langle u_-, \boldsymbol{\tau}\cdot\mathbf{n}\right\rangle_{\Gamma_{K_j}^-} \\
&\quad - \left\langle \frac{1}{2}u_-^2 n_x + u_- n_y, v\right\rangle_{\Gamma_{K_j}^-}, \quad \forall (v, \boldsymbol{\tau}) \in H^1(\Omega_h) \times H(\text{div}, \Omega_h),
\end{aligned} \tag{5}$$

with the broken spaces $H^1(\Omega_h)$ and $H(\text{div}, \Omega_h)$ defined as

$$H^1(\Omega_h) = \left\{v \in L^2(\Omega) : v|_{K_j} \in H^1\left(K_j\right)\right\},$$

$$H(\text{div}, \Omega_h) = \left\{\boldsymbol{\tau} \in L^2(\Omega) : \boldsymbol{\tau}|_{K_j} \in H\left(\text{div}, K_j\right)\right\},$$

[2] Rigorously, $\mathbf{q} \in \left[L^2(\Omega)\right]^2$ but we use $\mathbf{q} \in L^2(\Omega)$ throughout the paper for simplicity in writing.

and

$$\operatorname{sgn}(\mathbf{n}) := \begin{cases} 1 & \text{if } \mathbf{n} = \mathbf{n}^e \\ -1 & \text{if } \mathbf{n} = -\mathbf{n}^e. \end{cases}$$

It should be mentioned that the well-posedness of the analogous formulation for a large family of Friedrichs' systems of linear first order PDEs has been shown in [10]. Nevertheless, it is not clear to us whether the nonlinear formulation (5) in general has a solution. Instead of struggling with the well-posedness for nonlinear problems, which may be impossible in general, we proceed with the procedure for finding a solution assuming that there is at least one. Introducing $V := H^1(\Omega_h) \times H(\mathrm{div}, \Omega_h)$ we rewrite (5) in the abstract form

$$\left\langle B\left(u, \mathbf{q}, \hat{u}, \hat{q}\right), (v, \boldsymbol{\tau})\right\rangle_{V' \times V} := b\left(\left(u, \mathbf{q}, \hat{u}, \hat{q}\right), (v, \boldsymbol{\tau})\right) = \ell\left((v, \boldsymbol{\tau})\right) =: \left\langle \ell, (v, \boldsymbol{\tau})\right\rangle_{V' \times V},$$

with $b(\cdot,\cdot)$ and $\ell(\cdot)$ obviously defined as the left- and the right-hand sides of (5). In addition, we denote by V' the topological dual of V. Next, one may realize, at least psychologically, that "there are more unknowns than the number of equations" since we have introduced two new unknown variables $\hat{u}$ and $\hat{q}$ without adding more equations. This immediately suggests that one should seek a solution using the popular least squares approach. This is essentially the main idea behind the DPG method with optimal test functions. In particular, we minimize the residual in the dual space V' using least squares, i.e.,

$$\inf_{\mathcal{U}} \mathcal{J} := \frac{1}{2}\left\| B(\mathcal{U}) - \ell \right\|_{V'}^2, \tag{6}$$

where we have defined the group variable $\mathcal{U} := \left(u, \mathbf{q}, \hat{u}, \hat{q}\right)$. At this point, one can set the first variation of $\mathcal{J}$ to zero (see, e.g., [17]) to arrive at the DPG formulation with optimal test functions, which is then discretized and solved. This is the standard approach in the DPG literature (see, e.g., [1–9,14,17]). More explicitly, taking the Gâteaux derivative of $\mathcal{J}$, $\mathscr{D}\mathcal{J}$, with respect to $\mathcal{U}$ in the direction $\tilde{\mathcal{U}}$ and then setting it to zero we obtain the standard DPG equation

$$\left\langle B(\mathcal{U}) - \ell, R_V^{-1}\mathscr{D}B\left(\mathcal{U}; \tilde{\mathcal{U}}\right)\right\rangle_{V' \times V} = 0, \quad \forall \tilde{\mathcal{U}} \in U, \tag{7}$$

which is equivalent to

$$b\left(\mathcal{U}, \mathcal{V}_{\tilde{\mathcal{U}}}\right) = \ell\left(\mathcal{V}_{\tilde{\mathcal{U}}}\right), \quad \forall \tilde{\mathcal{U}} \in U,$$

where $\mathcal{V}_{\tilde{\mathcal{U}}} := R_V^{-1}\mathscr{D}B\left(\mathcal{U}; \tilde{\mathcal{U}}\right)$ is known as the nonlinear optimal test function corresponding to $\tilde{\mathcal{U}}$ [20] (because it is a function of $\mathcal{U}$ and it makes the residual minimal at the same time, assuming that the Hessian is positive definite). Here, R_V is the Riesz map that takes a function in V to its unique corresponding functional in V'.

3. DPG as a PDE-constrained optimization problem (oDPG)

Here, we take a radically different approach, namely, a constrained optimization approach as opposed to the standard unconstrained approach in Section 2. We first define $\mathcal{V}$ as the Riesz representation of the residual $B(\mathcal{U}) - \ell$ in V and denote by $(\cdot,\cdot)_V$ the inner product of two functions in V, and then, by the isometry and definition of the Riesz map, our unconstrained optimization problem (6) becomes

$$\inf_{\mathcal{U}} \mathcal{J} := \frac{1}{2}\left\|\mathcal{V}\right\|_V^2, \tag{8a}$$

subject to

$$\left(\mathcal{V}, \mathcal{W}\right)_V = \left\langle B(\mathcal{U}), \mathcal{W}\right\rangle_{V' \times V} - \left\langle \ell, \mathcal{W}\right\rangle_{V' \times V}, \quad \forall \mathcal{W} \in V, \tag{8b}$$

which is a constrained minimization problem.

Remark 1. In the context of PDE-constrained optimization, the unknown solution $\mathcal{U} = \left(u, \mathbf{q}, \hat{u}, \hat{q}\right)$ is known as the (control) parameter while the constraint (8b) is called the forward equation. It is interesting to realize that no matter how complicated and/or nonlinear the original first order PDE under consideration is the forward equation restricted on each element K_j is always a linear PDE in $\mathcal{V}$ and the differential order is dictated by the norm in V. For example, for the inviscid Burger equation, and hence for any conservation laws, the forward PDE is always the diffusion–reaction problem on K_j since the test space is always $H^1(\Omega_h)$. It follows that we can use the standard continuous finite element to solve the forward equation in an element-by-element fashion. It is interesting to point out that though we do not need to solve the original Burger equation directly, we still have to discretize it. In fact, the resulting discretized residual of the original PDE is the forcing for the forward equation.

We solve our constrained optimization problem using the "reduced space approach", i.e. $\mathcal{U}$ is essentially the only optimization variable (as opposed to the full space approach in which both $\mathcal{U}$ and $\mathcal{V}$ are optimization variables), via the standard adjoint method [21]. To this end, we define the Lagrangian

$$\mathcal{L} = \frac{1}{2}\|\mathcal{V}\|_V^2 + (\mathcal{V},\mathcal{W})_V - \langle B(\mathcal{U}),\mathcal{W}\rangle_{V'\times V} + \langle \ell,\mathcal{W}\rangle_{V'\times V}.$$

Taking the first variation of the Lagrangian with respect to $\mathcal{W}$ in the direction $\tilde{\mathcal{W}}$ and arguing that the variation $\tilde{\mathcal{W}}$ is arbitrary yield the forward equation (8b).

Now taking the first variation the Lagrangian with respect to $\mathcal{V}$ in the direction $\tilde{\mathcal{V}}$ and arguing that the variation $\tilde{\mathcal{V}}$ is arbitrary yield the so-called adjoint equation

$$\left(\tilde{\mathcal{V}},\mathcal{V}\right)_V + \left(\tilde{\mathcal{V}},\mathcal{W}\right)_V = 0 \quad \forall \tilde{\mathcal{V}} \in V. \tag{9}$$

It follows that the adjoint equation is simply $\mathcal{W} = -\mathcal{V}$. Finally, taking the first variation the Lagrangian with respect to $\mathcal{U}$ in the direction $\tilde{\mathcal{U}}$ (satisfying $\mathcal{U}+\tilde{\mathcal{U}} \in U$) yields the gradient in the direction $\tilde{\mathcal{U}}$

$$\mathcal{G}\left(\mathcal{U};\tilde{\mathcal{U}}\right) := -\left\langle \mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right),\mathcal{W}\right\rangle_{V'\times V},$$

where $\mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right)$ denotes the first variation of $B(\mathcal{U})$ in the direction $\tilde{\mathcal{U}}$, while $\mathscr{D}B(\mathcal{U})$ is a linear operator from U to V'. Eliminating the adjoint equation (9) we can rewrite the gradient as

$$\mathcal{G}\left(\mathcal{U};\tilde{\mathcal{U}}\right) = \left\langle \mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right),\mathcal{V}\right\rangle_{V'\times V}. \tag{10}$$

The procedure for computing the cost and the gradient is now clear. We first solve the forward equation (8b) for $\mathcal{V}$, which is then used to compute the cost functional (8a) and the gradient (10) in a given direction $\tilde{\mathcal{U}}$.

We shall describe in Section 8 a trust region inexact Newton conjugate gradient (Newton-CG) method to solve the PDE-constrained optimization problem (8). It is therefore necessary to compute the Hessian of the cost functional (8a). The Hessian acting in the directions $\tilde{\mathcal{U}}$ and $\dot{\mathcal{U}}$ (satisfying $\mathcal{U}+\dot{\mathcal{U}} \in U$) is obtained by simply taking the first variation of the gradient with respect to $\mathcal{U}$ and $\mathcal{V}$ in the direction $\dot{\mathcal{U}}$ and $\dot{\mathcal{V}}$ (satisfying $\mathcal{V}+\dot{\mathcal{V}} \in V$):

$$\mathcal{H}\left(\mathcal{U};\tilde{\mathcal{U}},\dot{\mathcal{U}}\right) = \left\langle \mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right),\dot{\mathcal{V}}\right\rangle_{V'\times V} + \left\langle \mathscr{D}^2B\left(\mathcal{U};\tilde{\mathcal{U}},\dot{\mathcal{U}}\right),\mathcal{V}\right\rangle_{V'\times V}. \tag{11}$$

As mentioned at the beginning of this section, the reduced space approach is employed, and hence the variations $\dot{\mathcal{V}}$ cannot be arbitrary. In fact, it is only admissible if the forward equation (8b) is satisfied. As a direct consequence, the first variation of (8b) with respect to $\mathcal{U}$ and $\mathcal{V}$ in the directions $\dot{\mathcal{U}}$ and $\dot{\mathcal{V}}$ must vanish, that is, $\dot{\mathcal{V}}$ is the solution of the following incremental forward equation:

$$(\dot{\mathcal{V}},\mathcal{W})_V = \langle \mathscr{D}B(\mathcal{U};\dot{\mathcal{U}}),\mathcal{W}\rangle_{V'\times V}, \quad \forall \mathcal{W} \in V. \tag{12}$$

As a result, once the cost and gradient in a given direction $\tilde{\mathcal{U}}$ is computed, one solves the incremental forward equation (12) for $\dot{\mathcal{V}}$ and then evaluates the Hessian as in (11).

At this point, one may think that our approach seems to be much more complicated and disadvantageous compared to the standard unconstrained approach. This is not true. On the contrary, our approach opens the door to invite all the state-of-the-art PDE-constrained techniques to be part of the DPG framework, and hence enabling us to solve large-scale and difficult (nonlinear) problems. More on the advantage of the proposed approach is discussed in Section 5.

4. Connection between DPG and oDPG

As can be seen, the standard DPG with optimal test function described at the end of Section 2 seems to be different from our PDE-constrained approach, namely oDPG, in Section 3. In particular, we do not have the concept of optimal test function in the oDPG approach and we do not solve the original PDE directly either. In this section we would like to address the relationship between the standard DPG method and oDPG.

By construction, the cost function (8a) of oDPG is exactly the same as the cost function (6) of the standard DPG. We next show that the gradient of the DPG method, namely, the left-hand side of (7) is identical to the oDPG gradient (10). Using the definition of the Riesz map R_V, the forward equation is equivalent to

$$R_V \mathcal{V} = B(\mathcal{U}) - \ell \quad \text{in } V',$$

which can be then substituted in (10) to give

$$\mathcal{G}\left(\mathcal{U};\tilde{\mathcal{U}}\right) = \left\langle \mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right), R_V^{-1}\left(B(\mathcal{U}) - \ell\right)\right\rangle_{V'\times V},$$

which, in turn, is exactly the left-hand side of (7) using the property of the Riesz map and its inverse. In order to solve (7), one typically uses the Newton–Raphson method in which the Jacobian of the left-hand side is constructed. This is exactly the Hessian in the oDPG approach.

In summary, though both standard DPG and oDPG solve the same variational problem in exact arithmetic, the former follows the Euler–Lagrange method while the latter can be considered as a Rayleigh–Ritz approach. As shall be shown in Section 8, our optimization solver requires the evaluation of the cost, gradient, and Hessian-vector product for a given value of $\mathcal{U}$. This amounts to solving the forward equation (8b) and incremental forward equation (12) for $\mathcal{V}$ and $\tilde{\mathcal{V}}$ locally, element-by-element, by inverting elemental Riesz maps on K_j. The construction of gradient and Hessian-vector product for the field variables $u, \mathbf{q}$ is also local on each element, while it requires immediate neighboring information for $\hat{u}, \hat{q}$. That is, the oDPG approach does not construct any global matrix (e.g. the Jacobian/Hessian) explicitly. Instead, it exploits the power of the adjoint technique to compute the Hessian-vector product *exactly but implicitly without ever forming the Hessian matrix*. This important point is best illustrated on the viscous Burger equation in Section 5.

5. Explicit gradient and Hessian-vector product for the Burger equation

In this section, the abstract gradient and Hessian-vector product in Section 3 will be specified for the viscous Burger equation. To be efficient in actual implementation, we choose test functions that are localized on each element. Taking $\mathcal{W} = (w, \mathbf{0})$ such that supp$w \subseteq K_j$, the forward equation (8b) reads

$$\begin{aligned}(v, w)_{H^1(K_j)} = &-\frac{1}{2}\left(u^2, \frac{\partial w}{\partial x}\right)_{K_j} - \left(u, \frac{\partial w}{\partial y}\right)_{K_j} + \varepsilon\left(\mathbf{q}, \nabla w\right)_{K_j} \\ &+ \left\langle \frac{1}{2}\hat{u}^2 n_x + \hat{u} n_y, w\right\rangle_{\Gamma_{K_j}} - \varepsilon\left\langle \hat{q}\,\mathrm{sgn}(\mathbf{n}), w\right\rangle_{\Gamma_{K_j}}. \end{aligned} \tag{13}$$

On the other hand, taking $\mathcal{W} = (0, \boldsymbol{\sigma})$ such that supp$\boldsymbol{\sigma} \subseteq K_j$, the forward equation (8b) becomes

$$(\boldsymbol{\tau}, \boldsymbol{\sigma})_{H(\mathrm{div}, K_j)} = (\mathbf{q}, \boldsymbol{\sigma})_{K_j} + (u, \nabla\cdot\boldsymbol{\sigma})_{K_j} - \left\langle \hat{u}, \boldsymbol{\sigma}\cdot\mathbf{n}\right\rangle_{\Gamma_{K_j}}. \tag{14}$$

Thus, given $\mathcal{U}$ one can solve for the residual representation $\mathcal{V}$ in an element-by-element fashion by inverting symmetric positive definite Gram matrices generated by $H^1\left(K_j\right)$ and $H\left(\mathrm{div}, K_j\right)$.

Next, taking $\tilde{\mathcal{U}} = (\tilde{u}, \mathbf{0}, 0, \mathbf{0})$ in (10) such that $\tilde{u}$ has local support in K_j gives the gradient of the cost function (8a) with respect to u in the direction $\tilde{u}$

$$\langle \mathscr{D}\mathcal{J}, \tilde{u}\rangle := \mathcal{G}\left(\mathcal{U};\tilde{\mathcal{U}}\right) = -\left(u\tilde{u}, \frac{\partial v}{\partial x}\right)_{K_j} - \left(\tilde{u}, \frac{\partial v}{\partial y}\right)_{K_j} + (\tilde{u}, \nabla\cdot\boldsymbol{\tau})_{K_j}, \tag{15}$$

where we have omitted the subscript $V' \times V$ in the definition of the Gâteaux derivative of the cost $\mathcal{J}$ for simplicity in writing. The gradient with respect to $\mathbf{q}$ in the direction $\tilde{\mathbf{q}}$ such that $\tilde{\mathbf{q}}$ has local support in K_j can be obtained by taking $\tilde{\mathcal{U}} = \left(0, \tilde{\mathbf{q}}, 0, \mathbf{0}\right)$:

$$\langle \mathscr{D}\mathcal{J}, \tilde{\mathbf{q}} \rangle := \mathcal{G}\left(\mathcal{U}; \tilde{\mathcal{U}}\right) = \varepsilon \left(\tilde{\mathbf{q}}, \nabla v\right)_{K_j} + \left(\tilde{\mathbf{q}}, \boldsymbol{\tau}\right)_{K_j}. \tag{16}$$

Similarly, the gradient with respect to $\hat{u}$ in the direction $\tilde{\hat{u}}$ such that $\tilde{\hat{u}}$ has local support in $\Gamma^+_{K_j}$ reads

$$\left\langle \mathscr{D}\mathcal{J}, \tilde{\hat{u}} \right\rangle := \mathcal{G}\left(\mathcal{U}; \tilde{\mathcal{U}}\right) = \left\langle \hat{u}\tilde{\hat{u}}, [\![v n_x]\!] \right\rangle_{\Gamma^+_{K_j}} + \left\langle \tilde{\hat{u}}, [\![v n_y]\!] \right\rangle_{\Gamma^+_{K_j}} - \left\langle \tilde{\hat{u}}, [\![\boldsymbol{\tau}\cdot\mathbf{n}]\!] \right\rangle_{\Gamma^+_{K_j}}, \tag{17}$$

where we have used the standard jump operator for any quantity $(\cdot)$ across and edge e as $[\![(\cdot)]\!] := (\cdot)^+ + (\cdot)^-$ with superscripts $\pm$ indicating the (relative) "left" and "right" values of $(\cdot)$. Now, taking $\tilde{\mathcal{U}} = \left(0, \mathbf{0}, 0, \tilde{\hat{q}}\right)$ such that $\tilde{\hat{q}}$ has local support in Γ_{K_j} gives the gradient of the cost function (8a) with respect to $\hat{q}$ in the direction $\tilde{\hat{q}}$

$$\left\langle \mathscr{D}\mathcal{J}, \tilde{\hat{q}} \right\rangle := \mathcal{G}\left(\mathcal{U}; \tilde{\mathcal{U}}\right) = -\varepsilon \left\langle \tilde{\hat{q}}, [\![v\mathbf{n}]\!] \right\rangle_{\Gamma_{K_j}}. \tag{18}$$

The Hessian acting in directions $\tilde{\mathcal{U}}$ and $\dot{\mathcal{U}} := \left(\dot{u}, \dot{\mathbf{q}}, \dot{\hat{u}}, \dot{\hat{\mathbf{q}}}\right)$, written abstractly in (11), has the same number of components as the gradient. In particular, for the first gradient component in (15), the corresponding Hessian-vector product is given by

$$\langle \mathscr{D}\langle \mathscr{D}\mathcal{J}, \tilde{u}\rangle, \dot{\mathcal{U}} \rangle = -\left(u\tilde{u}, \frac{\partial \dot{v}}{\partial x}\right)_{K_j} - \left(\tilde{u}, \frac{\partial \dot{v}}{\partial y}\right)_{K_j} + \left(\tilde{u}, \nabla\cdot\dot{\boldsymbol{\tau}}\right)_{K_j} - \left(\dot{u}\tilde{u}, \frac{\partial v}{\partial x}\right)_{K_j},$$

for the second gradient component in (16) the corresponding Hessian is

$$\langle \mathscr{D}\langle \mathscr{D}\mathcal{J}, \tilde{\mathbf{q}}\rangle, \dot{\mathcal{U}} \rangle = \varepsilon \left(\tilde{\mathbf{q}}, \nabla \dot{v}\right)_{K_j} + \left(\tilde{\mathbf{q}}, \dot{\boldsymbol{\tau}}\right)_{K_j},$$

for the third gradient component in (17) the corresponding Hessian reads

$$\left\langle \mathscr{D}\left\langle \mathscr{D}\mathcal{J}, \tilde{\hat{u}}\right\rangle, \dot{\mathcal{U}} \right\rangle = \left\langle \hat{u}\tilde{\hat{u}}, [\![\dot{v} n_x]\!] \right\rangle_{\Gamma^+_{K_j}} + \left\langle \tilde{\hat{u}}, [\![\dot{v} n_y]\!] \right\rangle_{\Gamma^+_{K_j}} - \left\langle \tilde{\hat{u}}, [\![\dot{\boldsymbol{\tau}}\cdot\mathbf{n}]\!] \right\rangle_{\Gamma^+_{K_j}} + \left\langle \dot{\hat{u}}\tilde{\hat{u}}, [\![v n_x]\!] \right\rangle_{\Gamma^+_{K_j}},$$

and for the fourth gradient component in (18) the corresponding Hessian is given by

$$\left\langle \mathscr{D}\left\langle \mathscr{D}\mathcal{J}, \tilde{\hat{q}}\right\rangle, \dot{\mathcal{U}} \right\rangle = -\varepsilon \left\langle \tilde{\hat{q}}, [\![\dot{v}\mathbf{n}]\!] \right\rangle_{\Gamma_{K_j}},$$

where $\dot{\mathcal{V}} = (\dot{v}, \dot{\boldsymbol{\tau}})$.

Again, the variation $\dot{\mathcal{V}} = (\dot{v}, \dot{\boldsymbol{\tau}})$ cannot be arbitrary but satisfies the incremental forward equation. Similar to the forward equations (13) and (14), the incremental counterpart also has two sets, the first of which is the first variation of (13) and can be shown to be

$$\begin{aligned}(\dot{v}, w)_{H^1(K_j)} = &-\left(\dot{u}u, \frac{\partial w}{\partial x}\right)_{K_j} - \left(\dot{u}, \frac{\partial w}{\partial y}\right)_{K_j} + \varepsilon\left(\dot{\mathbf{q}}, \nabla w\right)_{K_j} \\ &+ \left\langle \dot{\hat{u}}\hat{u}n_x + \dot{\hat{u}}n_y, w \right\rangle_{\Gamma_{K_j}\setminus\Gamma^-} - \varepsilon \left\langle \dot{\hat{\mathbf{q}}}\,\mathrm{sgn}\left(\mathbf{n}\right), w \right\rangle_{\Gamma_{K_j}}, \quad \forall w \in H^1\left(K_j\right),\end{aligned}$$

while the second of which is the first variation of (14), i.e.,

$$(\dot{\boldsymbol{\tau}}, \boldsymbol{\sigma})_{H(\mathrm{div}, K_j)} = (\dot{\mathbf{q}}, \boldsymbol{\sigma})_{K_j} + (\dot{u}, \nabla\cdot\boldsymbol{\sigma})_{K_j} - \left\langle \dot{\hat{u}}, \boldsymbol{\sigma}\cdot\mathbf{n} \right\rangle_{\Gamma_{K_j}\setminus\Gamma^-}, \quad \forall \boldsymbol{\sigma} \in H\left(\mathrm{div}, K_j\right).$$

This completes the explicit description of the gradient and Hessian-vector product for the viscous Burger equation.

As can be seen, computing the cost, gradient, and Hessian-vector product is local in an element-by-element and edge-by-edge fashion. For gradient and Hessian-vector product with respect to hybrid variables $\hat{u}$ and $\hat{q}$ on an edge

e, only two neighboring elements sharing that edge are involved in the computation. Consequently, global matrices or matrix–vector products are never formed in our approach. If we denote by m_{el} the number of unknowns corresponding to u on each element K_j and by m_{ed} the number of unknowns corresponding to $\hat{u}$ on each edge e, then it is clear from above that the complexity of constructing the cost, gradient, and a Hessian-vector product behaves like $\mathcal{O}\left(m_{el}^3 \times N_{\mathrm{el}} + m_{ed}^2 \times N_{\mathrm{ed}}\right)$. Since the total number of unknowns scales as $N = \mathcal{O}\left(m_{el} \times N_{\mathrm{el}} + m_{ed} \times N_{\mathrm{ed}}\right)$, the complexity is linear in the number of unknowns N assuming $m_{el} \times N_{\mathrm{el}} \gg m_{el}^2$ and $m_{ed} \times N_{\mathrm{ed}} \gg m_{ed}$.

6. Enforcing conservation

Note that the oDPG method is general not conservative. The reason is that it is equivalent to the DPG method which is not conservative since constant function, i.e. 1, may not be a member of the optimal test functions $\mathcal{V}_{\tilde{\mathcal{U}}} := R_V^{-1}\mathscr{D}B\left(\mathcal{U};\tilde{\mathcal{U}}\right)$. However, this can be enforced explicitly. Ideally it is desirable to have the following identity

$$\left\langle \frac{1}{2}\hat{u}^2 n_x + \hat{u} n_y, 1\right\rangle_{\Gamma_{K_j}} - \varepsilon\left\langle \hat{q}\,\mathrm{sgn}\left(\mathbf{n}\right), 1\right\rangle_{\Gamma_{K_j}} = 0, \quad \forall K_j, \tag{19}$$

which is a weak statement (by setting the test function to be constant) of the following pointwise conservation:

$$\nabla \cdot \left[\left(\frac{1}{2}u^2, u\right) - \varepsilon \mathbf{q}\right] = 0.$$

Perhaps, the easiest way to approximately enforce (19) is to use the penalty method. In this approach, we augment the cost function (8a) by an amount of $\frac{\lambda}{2}C^2$ so that the augmented cost functional reads

$$\mathcal{J}' := \mathcal{J} + \frac{\lambda}{2}C^2,$$

where λ is some penalty constant and

$$C := \sum_{K_j} \left\langle \frac{1}{2}\hat{u}^2 n_x + \hat{u} n_y, 1\right\rangle_{\Gamma_{K_j}} - \varepsilon\left\langle \hat{q}\,\mathrm{sgn}\left(\mathbf{n}\right), 1\right\rangle_{\Gamma_{K_j}}. \tag{20}$$

The advantage of this approach is that the whole reduced space framework in Section 5 is not changed except for a small modification of the gradient and Hessian due to the contribution of $\frac{\lambda}{2}C^2$. The disadvantage is clearly that the conservation is not exactly satisfied.

7. Discretization

We have introduced the oDPG method on the infinite dimensional level in Section 3. Let us now approximate infinite dimensional spaces U and V by finite dimensional piecewise polynomial spaces U_h and V_h on which we solve the constrained variational problem (8a)–(8b). We define

$$U_h := \left\{\mathcal{U}_h = \left(u_h, \mathbf{q}_h, \hat{u}_h, \hat{\mathbf{q}}_h\right) \middle| \begin{array}{ll} u_h|_{K_j} \in \mathcal{P}^p\left(K_j\right), \ \mathbf{q}_h|_{K_j} \in \left[\mathcal{P}^p\left(K_j\right)\right]^2, & K_j \in \Omega_h \\ \hat{u}_h\big|_e \in \mathcal{P}^p\left(e\right), \ \hat{\mathbf{q}}_h\big|_e \in \mathcal{P}^p\left(e\right), & e \in \Gamma_h \end{array}\right\},$$

$$V_h := \left\{\mathcal{V}_h = \left(v_h, \boldsymbol{\tau}_h\right) : v_h|_{K_j} \in \mathcal{P}^{p+\Delta p}\left(K_j\right), \ \boldsymbol{\tau}_h|_{K_j} \in \left[\mathcal{P}^{p+\Delta p}\left(K_j\right)\right]^2, \quad K_j \in \Omega_h\right\}.$$

It should be mentioned that, for simplicity of the exposition, we have used uniform polynomial order $p \geq 0$ and enriched order $\Delta p \geq 1$ for all elements; the extension to nonuniform p and Δp is straightforward.

Using the standard finite element ansatz applied to the group variable $\mathcal{U}_h$ we have

$$\mathcal{U}_h := \sum_{n=1}^{N} \mathcal{U}_h^n \Phi_n,$$

where Φ_n are either modal or nodal basis functions, the total of which is N; hence, $\mathcal{U}_h^n$ are either modal or nodal coefficients. The discretized counterpart of the infinite dimensional PDE-constrained optimization problem (8) reads

$$\min_{\mathcal{U}_h} \mathcal{J}_h := \frac{1}{2} \left\|\mathcal{V}_h\right\|_{V_h}^2, \tag{21a}$$

subject to

$$(\mathcal{V}_h, \mathcal{W}_h)_{V_h} = \langle B(\mathcal{U}_h), \mathcal{W}_h\rangle_{V_h{}'\times V_h} - \langle \ell, \mathcal{W}_h\rangle_{V_h{}'\times V_h}, \quad \forall \mathcal{W}_h \in V_h. \tag{21b}$$

Using (10), the gradient of $\mathcal{J}_h$ with respect to $\mathcal{U}_h^n$ can be written as

$$\mathcal{G}_h\left(\mathcal{U}_h; \mathcal{U}_h^n\right) = \langle \mathscr{D}B\left(\mathcal{U}_h;\, \Phi_n\right), \mathcal{V}_h\rangle_{V_h{}'\times V_h}.$$

Now, applying (11), the nth component of the product of the Hessian matrix with an arbitrary (nodal or modal) vector $\left\{\dot{\mathcal{U}}_h^n\right\}_n$ reads

$$\mathcal{H}_h^n\left(\mathcal{U}_h; \mathcal{U}_h^n, \dot{\mathcal{U}}_h\right) = \left\langle \mathscr{D}B\left(\mathcal{U}_h; \mathcal{U}_h^n\right), \dot{\mathcal{V}}_h\right\rangle_{V_h{}'\times V_h} + \left\langle \mathscr{D}^2 B\left(\mathcal{U}_h; \mathcal{U}_h^n, \dot{\mathcal{U}}_h\right), \mathcal{V}_h\right\rangle_{V_h{}'\times V_h},$$

where $\dot{\mathcal{U}}_h := \sum_n \dot{\mathcal{U}}_h^n \Phi_n$ and $\dot{\mathcal{V}}_h$ satisfies the discretized incremental forward equation

$$\left(\dot{\mathcal{V}}_h, \mathcal{W}_h\right)_{V_h} = \left\langle \mathscr{D}B\left(\mathcal{U}_h; \dot{\mathcal{U}}_h\right), \mathcal{W}_h\right\rangle_{V_h{}'\times V_h}, \quad \forall \mathcal{W}_h \in V_h.$$

Remark 2. There are two discretization approaches in the PDE-constrained literature (see, e.g., [22]), namely, discretize-then-optimize and optimize-then-discretize. In the former approach, one first discretizes the cost functional and the forward equation, and then optimizes the discretized optimization problem to find the gradient and Hessian. This method always guarantees that the resulting gradient and Hessian are exact gradient and Hessian of the discrete cost functional, the so-called gradient consistency. In the latter approach, on the other hand, one first optimizes the infinite dimensional optimization problem to find gradient and Hessian, and then discretizes the resulting cost, gradient, and Hessian. The discretized gradient and Hessian in this case are close but generally not exactly those of the discretized cost functional, though they do converge to the exact ones as the mesh size approaches zero. Nevertheless, when both approaches commute with each other, the gradient consistency is in fact satisfied, and this typically happens for discretizations that lead to symmetric matrices. Our above discretization for oDPG falls into the optimize-then-discretize category. Since the DPG stiffness matrix is symmetric (see, e.g., [2]), the commutation, and hence the gradient consistency, is guaranteed.

8. A trust region inexact Newton-CG solver

Section 7 equips us with the discretized cost, gradient, and Hessian-vector product. Therefore, we are in the position to use any existing first order (requires gradient) or second order (requires Hessian) optimization techniques. Of our interest are Newton-like optimization methods that converge quadratically as optimization iterates are sufficiently close to a local minimum. However, care must be taken since our problem is inherently large-scale. For example in the above Burger equation, if $p = 2$ and a triangular mesh is used, we have $N = 3\times 6N_{\text{el}} + 2\times 3N_{\text{ed}}$ unknowns, where N_{ed} is the number of edges in the skeleton Γ_h. Consequently, it is necessary to use a scalable large-scale optimization technique for our oDPG problem. We use the state-of-the-art trust region inexact Newton-CG (TRINCG) method which we now describe.

We consider the following generic unconstrained optimization problem

$$\min_{\mathbf{z}\in\mathbb{R}^N} \mathcal{F}(\mathbf{z}),$$

which we like to solve using sequential quadratic programming. In this approach, at the kth iteration, one needs to solve the following sub-optimization problem

$$\min_{\mathbf{s}\in\mathbb{R}^N} \left\{\varphi^k(\mathbf{s}) : \|\mathbf{s}\| \le \Delta^k\right\}, \tag{22}$$

where Δ^k is the current trust region radius whose updating rule is given in Algorithm 3, and $\varphi^k(\mathbf{s})$ the merit function given by

$$\varphi^k(\mathbf{s}) = \mathbf{s}^T \nabla \mathcal{F}\left(\mathbf{z}^k\right) + \frac{1}{2}\mathbf{s}^T \nabla^2 \mathcal{F}\left(\mathbf{z}^k\right)\mathbf{s}.$$

A sketch of our TRINCG solver is given in Algorithm 1 for which Step 3 is the key. In this step, we solve the trust region subproblem (22) inexactly as in Algorithm 2. In particular, if an iterate is far from the closest minimum, and hence the gradient is typically large, the inexact CG solver returns quickly to avoid unnecessary work. Closer to the optimum, the gradient is smaller and the number of CG iterations also increases in general to attain quadratic convergence. This is possible due to the fact that the method combines the rapid locally-quadratic convergence rate properties of Newton's method, the effectiveness of trust region globalization for treating ill-conditioned problems, and the Eisenstat–Walker idea of preventing over-solving. In other words, the inexact solver automatically adapts to minimize the work. The TRINCG algorithm is defined to converge if either

$$\left|\mathcal{F}\left(\mathbf{z}^k + \mathbf{s}\right) - \mathcal{F}\left(\mathbf{z}^k\right)\right| \le \varepsilon_F\left(1 + \mathcal{F}\left(\mathbf{z}^k\right)\right), \quad \text{or} \quad \|\mathbf{s}\|_2 \le \varepsilon_X, \quad \text{or} \quad \left\|\nabla \mathcal{F}\left(\mathbf{z}^k\right)\right\| \le \varepsilon_G,$$

is true, where ε_F, ε_X and ε_G are prescribed tolerances for the cost functional value, optimization variables, and the gradient, respectively.

Algorithm 1 TRINCG solver

```
1: while not converged do
2:     At the current Newton step $\mathbf{z}^k$, compute the gradient $\nabla\mathcal{F}\left(\mathbf{z}^k\right)$.
3:     Solve the subproblem (22) using inexact-CG Algorithm 2.
4:     Compute the ratio between the actual and the predicted reductions
       $ared := \mathcal{F}\left(\mathbf{z}^k + \mathbf{s}\right) - \mathcal{F}\left(\mathbf{z}^k\right), \quad pred := \varphi^k(\mathbf{s}), \quad \rho := \frac{ared}{pred}.$
5:     Update $\mathbf{z}^{k+1} \leftarrow \mathbf{z}^k$ and adjust the trust region radius $\Delta^{k+1} \leftarrow \Delta^k$ via Algorithm 3.
6: end while
```

Algorithm 2 Inexact Newton-CG

```
Ensure: $\varepsilon_{CG}$, $\nabla\mathcal{F}$ and a subroutine for the action of Hessian $\nabla^2\mathcal{F}$ with an arbitrary vector.
 1: Set $\mathbf{r} = \nabla\mathcal{F}$, and $\eta = \min\{\varepsilon_{CG}, \|\mathbf{r}\|\} \times \|\mathbf{r}\|$.
 2: Solve the Newton equation $\nabla^2\mathcal{F}\,\mathbf{s} = -\nabla\mathcal{F}$ using the standard CG method and return as follows:
 3: if negative curvature direction is detected then
 4:     Follow the negative curvature direction and set s up to the trust region boundary.
 5:     return s
 6: else if the residual of the Newton equation is less than or equal to $\eta$ then
 7:     return s
 8: else if the number of CG iterations is equal to $N$ then
 9:     return s
10: end if
```

It should be pointed out that for linear PDE, the TRINCG is exactly the standard CG method if the tolerance η is machine zero and the number of CG iterations is N.

9. Numerical results

In this section, we demonstrate our TRINCG solver for the oDPG formulation on linear problems including the Laplace and Helmholtz equations. It will be shown that h-convergence is optimal and p-convergence is exponential. We then present some numerical results for the TRINCG solver on two-dimensional nonlinear viscous Burger equation and two-dimensional Euler equation.

Algorithm 3 Trust region radius computation

```
Ensure: $\mu = 1.e-1$, $\beta = 0.25$, $\nu = 0.75$, $\gamma_0 = 0.0625$, $\gamma_1 = 0.5$, $\gamma_2 = 2$, $\Delta^{\max}$ and $BT^{\max}$
 1: if ($\rho > \mu$ and $ared < 0$) or $ared < 0$ then
 2:   if $\rho \le \beta$ and $\|\mathbf{s}\|_2 < \gamma_1 \Delta^k$ then
 3:     $\Delta^{k+1} = \gamma_1 \Delta^k$
 4:   else if $\rho > \nu$ then
 5:     if $\|\mathbf{s}\|_2 \ge 0.8\Delta^k$ then
 6:       $\Delta^{k+1} = \min\left\{\gamma_2 \Delta^k, \Delta^{\max}\right\}$
 7:     else
 8:       $\Delta^{k+1} = \min\left\{\max\left\{\Delta^k, \gamma_2 \|\mathbf{s}\|_2\right\}, \Delta^{\max}\right\}$
 9:     end if
10:   end if
11: else
12:   $k_{BT} = 0$
13:   while $ared \ge 0$ and $k_{BT} \le BT^{\max}$ do
14:     if $\rho \le 0$ then
15:       $\Delta^{k+1} = \gamma_0 \min\left\{\Delta^k, \|\mathbf{s}\|_2\right\}$
16:     else
17:       $\Delta^{k+1} = \gamma_1 \min\left\{\Delta^k, \|\mathbf{s}\|_2\right\}$
18:     end if
19:     Solve the trust region problem (22).
20:     Compute $ared$, $pred$ and $\rho$
21:     $k_{BT} = k_{BT} + 1$
22:   end while
23: end if
24: Update $\mathbf{z}^{k+1} = \mathbf{z}^k + \mathbf{s}$
```

9.1. Laplace equation

We consider the Laplace equation with Dirichlet boundary condition

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

which can be viewed as a simplification of the viscous Burger equation (1) in which the advection term is zero and $\varepsilon = 1$. Here, $\Omega = [0, 1]^2$. The forcing f and the Dirichlet boundary value g are chosen such that the exact solution reads

$$u = \sin(5x)\cos(7y).$$

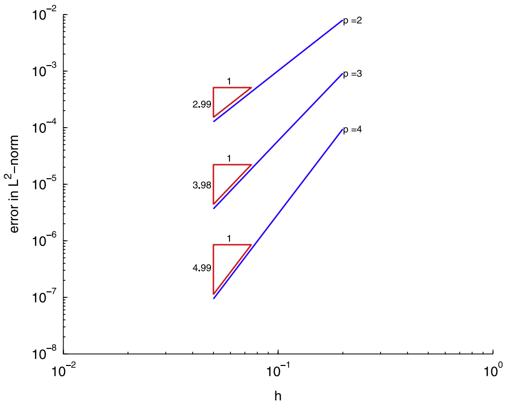
The tolerances are chosen as follows:

$$\varepsilon_F = 1 \times 10^{-12}, \qquad \varepsilon_X = 10^{-12}, \qquad \varepsilon_G = 10^{-12},$$

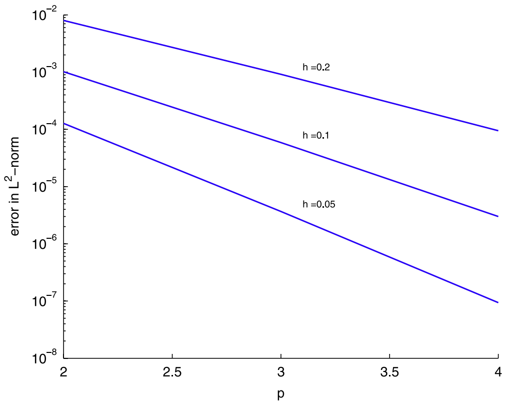
and the initial guess is chosen as $\mathcal{U} = \mathbf{0}$.

Fig. 1(a) shows the h-convergence in which we plot the error in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$, versus $h = \{0.05, 0.1, 0.2\}$ in the log–log scale for solution order in the range $p = \{2, 3, 4\}$ and $\Delta p = 1$. As expected, we obtain the optimal convergence order of $p + 1$. We also show the p-convergence in Fig. 1(b) for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$ in the linear-log scale, and in this case the convergence is exponential. Note that the solution is marginally improved for $\Delta p \ge 2$, and hence not shown here.

In order to see how the oDPG solution evolves during the optimization process, we plot the solution after one iteration together with the mesh in Fig. 2(a), the solution after four iterations in Fig. 2(b), and the solution after nine iterations in Fig. 2(c) (indistinguishable with the exact solution). In this case, we take $p = 4$ and $\Delta p = 1$. Note that the (relative) change in the cost function is less than ε_F after nine iterations. The total of unknowns in $\mathcal{U}_h$ is 2700.

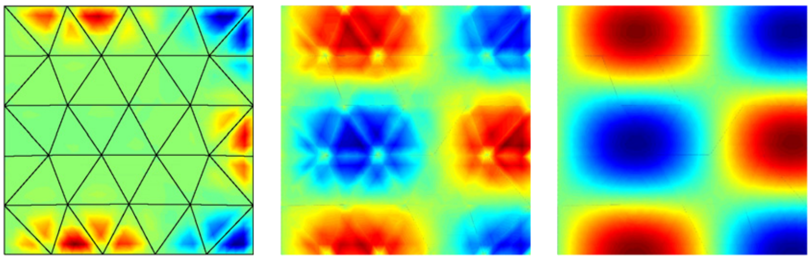


(a) h-convergence.



(b) p-convergence.

Fig. 1. h- and p-convergences for the Laplace equation: Fig. 1(a) Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. The mesh is refined in h for different polynomial orders from $p = 2$ to $p = 4$. The convergence is shown for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$. Fig. 1(b) Linear-log scale plot of the error of oDPG method in the L^2-norm. The solution p is refined, and the convergence is shown for three different mesh sizes $h \in \{0.05, 0.1, 0.2\}$.



(a) One iteration. (b) Four iterations. (c) Nine iterations.

Fig. 2. Solution of the Laplace equation using the oDPG method with $p = 4$ and $\Delta p = 1$. Fig. 2(a) is u_h after one iteration. Fig. 2(b) is u_h after four iterations. Fig. 2(c) is u_h after nine iterations.

9.2. Helmholtz equation

In this section, the Helmholtz equation is considered

$$-\Delta u - k^2 u = f \text{ in } \Omega,$$
$$u = g \text{ on } \partial\Omega,$$

which can be viewed as a variant of the viscous Burger equation (1) in which the advection term is replaced by $-k^2 u$ and $\varepsilon = 1$. Similar to the Laplace case, we take $\Omega = [0, 1]^2$. The forcing f and the Dirichlet boundary value g are chosen such that the exact solution reads

$$u = \cos\left(\tan^{-1}\left(\frac{y}{x+1}\right)\right) J_1\left(5\sqrt{(x+1)^2 + y^2}\right),$$

where J_1 is the standard Bessel function. The tolerances are chosen as

$$\varepsilon_F = 10^{-12}, \qquad \varepsilon_X = 10^{-12}, \qquad \varepsilon_G = 10^{-12}.$$

Similar to the Laplace case, we choose $\mathcal{U} = \mathbf{0}$ for the initial guess.

We choose $p = 4$ and $\Delta p = 1$ (giving a total of 2700 unknowns) and show the solution u_h after one iteration, four iterations, and eight iterations (indistinguishable with the exact solution) in Fig. 3. It turns out that the optimization stops after eight iterations when the criteria on the relative change in the cost function are met and the actual cost value (the residual square) in this case is 2.98×10^{-9}; the ℓ^2-norm of the discrete gradient is 2.8×10^{-9}. We also show h-

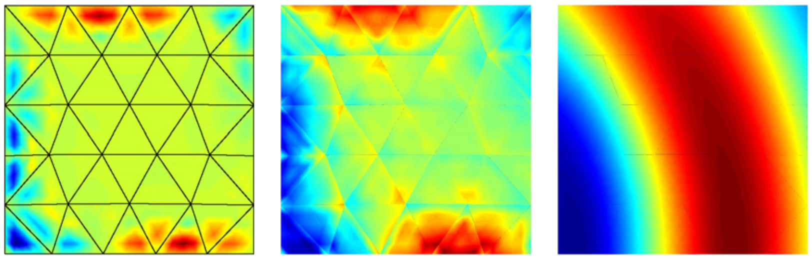
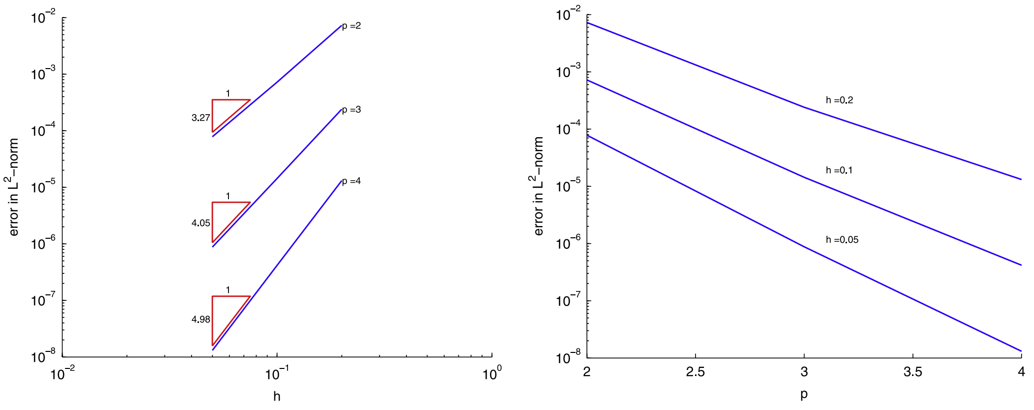

(a) One iteration. (b) Four iterations. (c) Eight iterations.

Fig. 3. Solution of the Helmholtz equation using the oDPG method with $p = 4$ and $\Delta p = 1$. Fig. 3(a) is u_h after one iteration. Fig. 3(b) is u_h after four iterations. Fig. 3(c) is u_h after eight iterations.



(a) h-convergence. (b) p-convergence.

Fig. 4. h- and p-convergences for the Helmholtz equation: Fig. 4(a) Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. Fig. 4(b) Linear-log scale plot of the error of oDPG method in the L^2-norm.

and p-convergences in Fig. 4. As can be seen, we obtain optimal convergence order in h and exponential convergence in p.

9.3. Burger equation

We next consider the viscous Burger equation with smooth exact solution. A forcing is added to the right-hand side of (1) such that the exact solution is given by

$$u_e = \frac{e^{r_1(x-1)} - e^{r_2(x-1)}}{e^{-r_1} - e^{-r_2}} \sin(\pi y), \tag{23}$$

which has a mild boundary layer at $x = 1$. Here,

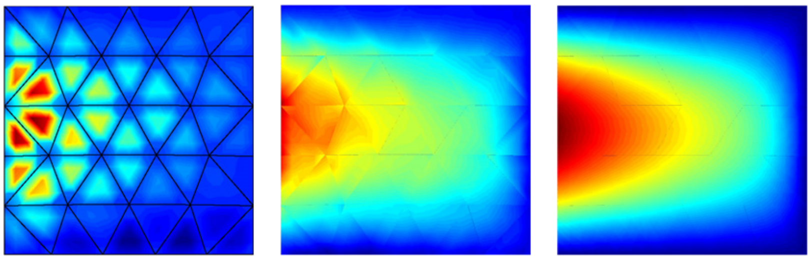
$$r_1 = -\frac{-1 + \sqrt{1 + 0.04\pi^2}}{0.2}, \quad \text{and} \quad r_2 = -\frac{-1 - \sqrt{1 + 0.04\pi^2}}{0.2}.$$

The domain under consideration is $\Omega = [0, 1]^2$. In order to enforce the Dirichlet boundary on $\partial\Omega$, we set $\hat{u}$ equal to the exact solution along $\partial\Omega$.

The tolerances for the optimization solver are chosen as follows

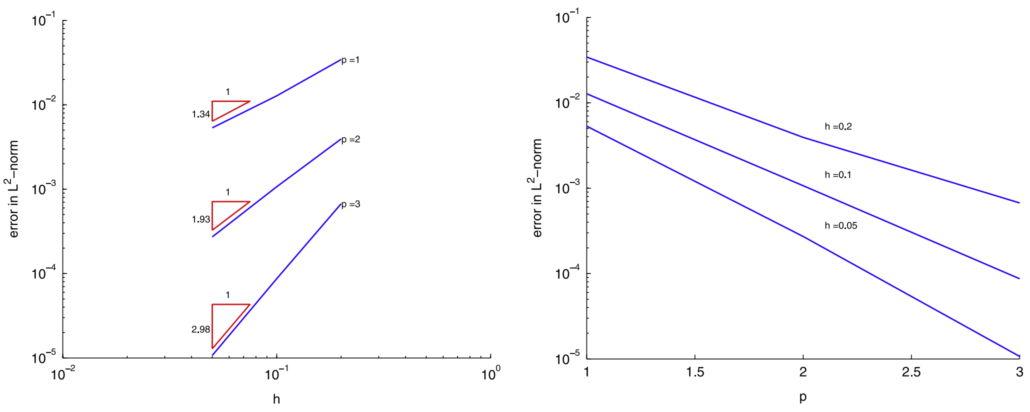
$$\varepsilon_F = 10^{-14}, \qquad \varepsilon_X = 10^{-14}, \qquad \varepsilon_G = 10^{-14},$$

and the initial guess is chosen to be $\mathcal{U} = \mathbf{0}$. Fig. 5 shows a typical evolution of the oDPG solution as the number of Newton iterations increases. In particular, we have used $p = 3$, $\Delta p = 1$, $h = 0.2$, and $\varepsilon = 10^{-1}$. For this case, the oDPG solution is almost indistinguishable to the exact solution after 14 iterations; the residual norm is 2.87×10^{-5} and the ℓ^2-norm of the discrete gradient is 1.66×10^{-9}.



(a) Solution after 2 iterations. (b) Solution after 5 iterations. (c) Solution after 12 iterations.

Fig. 5. Solution of the viscous Burger equation using the oDPG method with $p = 3$ and $\Delta p = 1$ and exact solution (23). Fig. 5(a) is the solution after 2 iterations. Fig. 5(b) is the solution after 5 iterations. Fig. 5(c) is the solution after 11 iterations.



(a) h-convergence. (b) p-convergence.

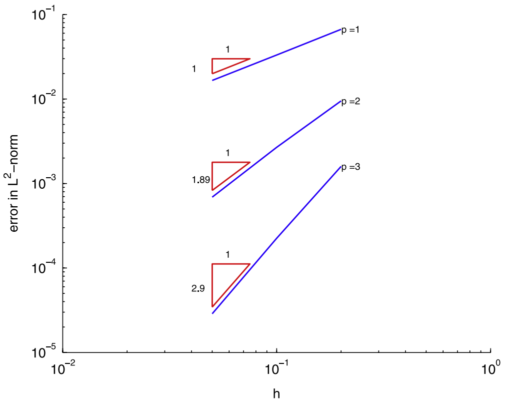
Fig. 6. h- and p-convergences for the viscous Burger equation with $\varepsilon = 10^{-1}$: Fig. 6(a) Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. The mesh is refined in h for different polynomial orders from $p = 1$ to $p = 3$. The convergence is shown for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$. Fig. 6(b) Linear-log scale plot of the error of oDPG method in the L^2-norm. The solution p is refined, and the convergence is shown for three different mesh sizes $h \in \{0.05, 0.1, 0.2\}$.

Table 1
The number of Newton iterations of oDPG method as the solution order and mesh are refined, viscous Burger equation with $\varepsilon = 10^{-1}$.

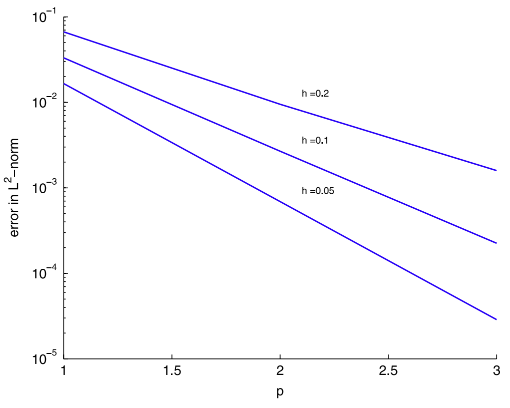
h	0.2	0.1	0.05
$p = 1$	11	13	15
$p = 2$	14	13	12
$p = 3$	13	14	13

The first question that we would like to address is how the number of Newton iterations varies as either the mesh or solution order is refined. To that end, we take $\varepsilon = 10^{-1}$. As can be seen from Table 1, the number of Newton iterations is essentially constant as the mesh is refined and/or the solution order increases. This mesh-independent property of our optimization, and hence iterative oDPG, solver is very desirable for high-fidelity large-scale problems. That is, our oDPG solver scales well with the solution resolution.

We next study h- and p-convergences. To that end, we choose two values for ε: 10^{-1} and 1. Fig. 6 shows, for $\varepsilon = 10^{-1}$, that h-convergence is suboptimal by one order, though p-convergence is still exponential. We observe similar behavior for $\varepsilon = 1$ in Fig. 7. Note that we have used $\Delta p = 1$ since $\Delta p \geq 2$ is just marginally better. Again, for all cases, the stopping criteria on relative change in the cost functional (residual) are met first. For all cases, we observe that the ℓ^2-norm of the discrete gradient is $\mathcal{O}\left(10^{-10}\right)$ when the optimization stops. It follows that the suboptimality is not due to premature convergence.



(a) h-convergence.



(b) p-convergence.

Fig. 7. h- and p-convergences for the viscous Burger equation with $\varepsilon = 1$: Fig. 7(a) Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. The mesh is refined in h for different polynomial orders from $p = 1$ to $p = 3$. The convergence is shown for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$. Fig. 7(b) Linear-log scale plot of the error of oDPG method in the L^2-norm. The solution p is refined, and the convergence is shown for three different mesh sizes $h \in \{0.05, 0.1, 0.2\}$.

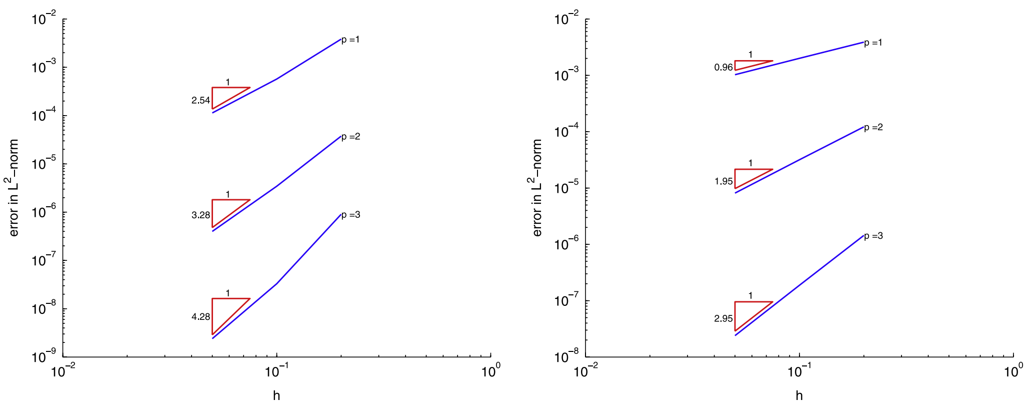
Table 2
Asymptotic conservation of the DPG method for $p = 3$, $\Delta p = 1$ and $h = \{0.05, 0.1, 0.2\}$, viscous Burger equation with $\varepsilon = 10^{-1}$.

h	0.2	0.1	0.05
Converged cost ($\lambda = 0$)	$7.67e-07$	$1.63e-08$	$2.71e-10$
Initial cost ($\lambda = 10^8$)	$6.67e-06$	$2.10e-08$	$2.73e-10$
C	$2.43e-07$	$6.86e-09$	$1.38e-10$

It should be pointed out that we do not enforce the conservation for the results in Figs. 6 and 7, and one may think that conservation may be helpful in recovering the optimal convergence. To address this, for example in the cases of $p = 3$, $\Delta p = 1$, $h = \{0.05, 0.1, 0.2\}$ and $\varepsilon = 10^{-1}$, we use the converged solutions with $\lambda = 0$ as initial guesses for the oDPG method with $\lambda = 10^8$. As can be observed from Table 2, the DPG method is not exactly conservative since C in (20) is not zero. However, C is the same order of (in fact smaller than) the residual norm at the converged DPG solutions (compared rows two, three, and fourth in Table 2). Furthermore, the conservation is improved as the mesh is refined, namely, the DPG is asymptotically conservative. We observe that for all h, our optimization solver takes at most five iterations to converge for $\lambda = 10^8$, and at the converged solution the cost and the L^2-norm $\|u - u_h\|_{L^2(\Omega_h)}$ are almost exactly the same as the converged solution for $\lambda = 0$. This implies that penalizing conservation discrepancy as we suggested does not improve the convergence rate. We also try to increase λ. However, since C^2 is machine zero, the penalty term λC^2 is essentially zero and hence does not contribute to the previous converged cost (with $\lambda = 0$). As a result, the optimization solver immediately exits, i.e., the converged solution does not change. We conclude that penalizing the conservation in DPG seems to be redundant, especially for a well-resolved DPG solution or machine zero C^2.

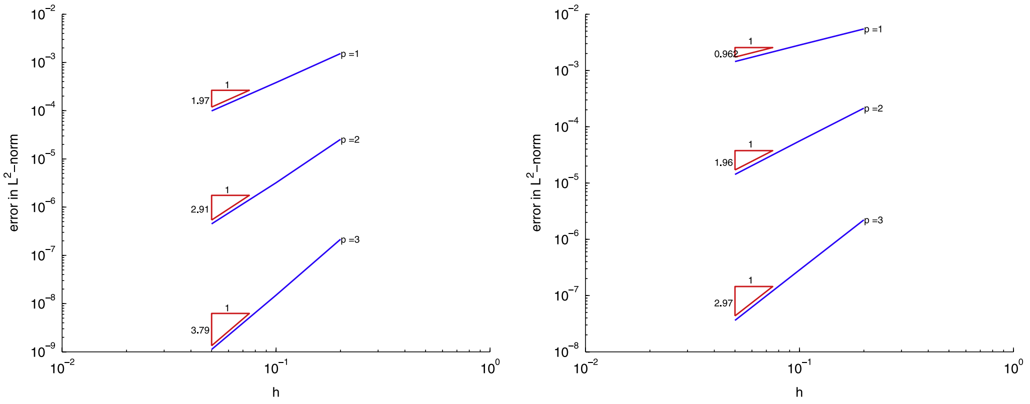
To remove the possibility of losing optimal convergence rate due to insufficiently resolving the boundary layer in the solution, let us now consider a case with very smooth solution, namely, $u = \sin(x)$. We consider $\varepsilon = 10^2$ and $\varepsilon = 1$, and study the h-convergence for these two cases. As can be seen in Fig. 8(a), the optimal convergence is achieved for $\varepsilon = 10^2$, but not for $\varepsilon = 1$ as shown in Fig. 8(b).

Therefore, the loss of optimal convergence rate is due to ε as it decreases. This is not surprising. Indeed, while the broken H^1-norm for the test space is natural and straightforward (especially for conservation laws in the conservative form), it is not robust with respect to ε. As such, it is deficiency-prone as ε decreases. This has been studied in the previous work [23,17] in which the provable robust norms involve ε. It could be that $\varepsilon < 1$ corresponds to convection-dominated and our test norm does not seem to provide the optimal convergence for convection-dominated flow. To further support this hypothesis, we carry out numerical experiments with linear convection–diffusion problem with



(a) $\varepsilon = 10^2$. (b) $\varepsilon = 1$.

Fig. 8. h-convergences for the viscous Burger equation with smooth solution $u = \sin(x)$ with $\varepsilon = 10^2$ (Fig. 8(a)) and $\varepsilon = 1$ (Fig. 8(b)). Both figures show the Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. The mesh is refined in h for different polynomial orders from $p = 1$ to $p = 3$. The convergence is shown for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$.



(a) $\varepsilon = 10^2$. (b) $\varepsilon = 1$.

Fig. 9. h-convergences for the linear convection–diffusion equation with smooth solution $u = \sin(x)$ with $\varepsilon = 10^2$ (Fig. 9(a)) and $\varepsilon = 1$ (Fig. 9(b)). Both figures show the Log–log scale plot of the error of oDPG method in the L^2-norm, i.e., $\|u - u_h\|_{L^2(\Omega_h)}$. The mesh is refined in h for different polynomial orders from $p = 1$ to $p = 3$. The convergence is shown for three different mesh sizes $h = \{0.05, 0.1, 0.2\}$.

smooth solution $u = \sin(x)$, again with $\varepsilon = 10^2$ and $\varepsilon = 1$. As shown in Fig. 9, we obtain similar results as those in Fig. 8, i.e., optimal convergence for $\varepsilon = 10^2$, but suboptimal for $\varepsilon = 1$. Our iterative DPG framework, the main focus of the paper, is clearly applicable for other norms (quasi-optimal or weighted test norms) [23,17], but for simplicity we restrict ourselves to the broken H^1-norm for the test space.

It should also be pointed out that, even for linear PDEs, the provable optimal convergence rate requires the traces/fluxes be discretized in $\mathcal{P}^{p+1}$ space, one order higher than u and $\mathbf{q}$ [4]. Our current implementation does not support different order discretization, and hence testing whether this is the cause of the loss of convergence (unlikely since optimal convergence rate has been obtained with $\varepsilon = 10^2$) is out of the scope of the current paper, whose main aim is to introduce a new and natural iterative solver for the DPG method.

9.4. Euler equation

We next consider the steady Euler equation in the conservative form

$$\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{f}, \tag{24}$$

where

$$\mathbf{F} = \mathbf{F}(\mathbf{q}) := \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ u(E+P) \end{bmatrix}, \qquad \mathbf{G} = \mathbf{G}(\mathbf{q}) := \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ v(E+P) \end{bmatrix}, \qquad \mathbf{q} := \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix},$$

where the pressure P is given by the state equation

$$E = \frac{P}{\gamma - 1} + \frac{1}{2\rho}\left[(\rho u)^2 + (\rho v)^2\right],$$

with $\gamma = 1.4$ for monoatomic gas. The domain under consideration is $\Omega = [0, 10] \times [-5, 5]$. Following the same exercise as for the viscous Burger equation, we multiply the Euler equation (24) with the test function

$$\boldsymbol{\tau} := [r, ru, rv, re]^T,$$

integrate the resulting equation by parts, and introduce single-valued hybrid variables

$$\hat{q} := \left[\widehat{\rho}, \widehat{\rho u}, \widehat{\rho v}, \widehat{E}\right]$$

on the skeleton, and hence obtain

$$\left\langle B\left(\mathbf{q}, \hat{q}\right), \boldsymbol{\tau}\right\rangle_{V' \times V} := b\left(\left(\mathbf{q}, \hat{q}\right), \boldsymbol{\tau}\right) = \ell(\boldsymbol{\tau}) =: \langle \ell, \boldsymbol{\tau}\rangle_{V' \times V}.$$

Here, the left-hand side is

$$\begin{aligned} b\left(\left(\mathbf{q}, \hat{q}\right), \boldsymbol{\tau}\right) = \sum_j -\left(\rho u, \frac{\partial r}{\partial x}\right)_{K_j} - \left(\rho v, \frac{\partial r}{\partial y}\right)_{K_j} + \left\langle \widehat{\rho u} n_x + \widehat{\rho v} n_y, r\right\rangle_{\Gamma^+_{K_j}} \\ - \left(\rho u^2 + P, \frac{\partial ru}{\partial x}\right)_{K_j} - \left(\rho uv, \frac{\partial ru}{\partial y}\right)_{K_j} + \left\langle \left(\frac{\widehat{\rho u}^2}{\widehat{\rho}} + \widehat{P}\right) n_x + \frac{\widehat{\rho u}\widehat{\rho v}}{\widehat{\rho}} n_y, ru \right\rangle_{\Gamma^+_{K_j}} \\ - \left(\rho uv, \frac{\partial rv}{\partial x}\right)_{K_j} - \left(\rho v^2 + P, \frac{\partial rv}{\partial y}\right)_{K_j} + \left\langle \frac{\widehat{\rho u}\widehat{\rho v}}{\widehat{\rho}} n_x + \left(\frac{\widehat{\rho v}^2}{\widehat{\rho}} + \widehat{P}\right) n_y, rv \right\rangle_{\Gamma^+_{K_j}} \\ - \left(u(E+P), \frac{\partial re}{\partial x}\right)_{K_j} - \left(v(E+P), \frac{\partial re}{\partial y}\right)_{K_j} + \left\langle \frac{\widehat{\rho u}}{\widehat{\rho}}\left(\widehat{E} + \widehat{P}\right) n_x + \frac{\widehat{\rho v}}{\widehat{\rho}}\left(\widehat{E} + \widehat{P}\right) n_y, rv \right\rangle_{\Gamma^+_{K_j}}, \end{aligned}$$

and the right-hand side

$$\begin{aligned} \ell(\boldsymbol{\tau}) = \sum_j (\mathbf{f}, \boldsymbol{\tau})_{K_j} - \left\langle \widehat{\rho u} n_x + \widehat{\rho v} n_y, r\right\rangle_{\Gamma^-_{K_j}} - \left\langle \left(\frac{\widehat{\rho u}^2}{\widehat{\rho}} + \widehat{P}\right) n_x + \frac{\widehat{\rho u}\widehat{\rho v}}{\widehat{\rho}} n_y, ru \right\rangle_{\Gamma^-_{K_j}} \\ - \left\langle \frac{\widehat{\rho u}\widehat{\rho v}}{\widehat{\rho}} n_x + \left(\frac{\widehat{\rho v}^2}{\widehat{\rho}} + \widehat{P}\right) n_y, rv \right\rangle_{\Gamma^-_{K_j}} - \left\langle \frac{\widehat{\rho u}}{\widehat{\rho}}\left(\widehat{E} + \widehat{P}\right) n_x + \frac{\widehat{\rho v}}{\widehat{\rho}}\left(\widehat{E} + \widehat{P}\right) n_y, rv \right\rangle_{\Gamma^-_{K_j}}, \end{aligned}$$

with

$$\widehat{E} = \frac{\widehat{P}}{\gamma - 1} + \frac{1}{2\widehat{\rho}}\left[\widehat{\rho u}^2 + \widehat{\rho v}^2\right].$$

Unlike the viscous Burger equation, we now seek solution $\left(\mathbf{q}, \hat{q}\right) \in U := L^2(\Omega) \times H^{-\frac{1}{2}}(\Gamma_h)$ and the test space $V := H^1(\Omega_h)$. The rest of the steps, namely, computing the adjoint, gradient, Hessian-vector product, and discretization are similar to those of viscous Burger equation, and hence omitted.

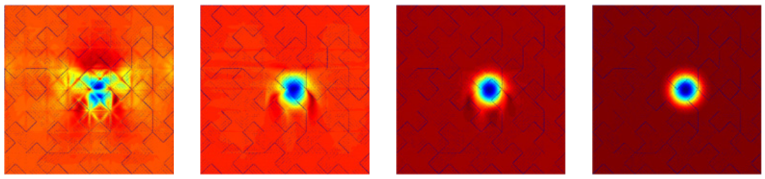

(a) 20 iterations. (b) 40 iterations. (c) 60 iterations. (d) ρ exact.

Fig. 10. Solution of the Euler equation using the oDPG method with $p = 4$ and $\Delta p = 2$: Fig. 10(a) is ρ_h after 20 iterations, Fig. 10(b) is ρ_h after 40 iterations, Fig. 10(c) is ρ_h after 60 iterations, and Fig. 10(d) is the exact ρ.

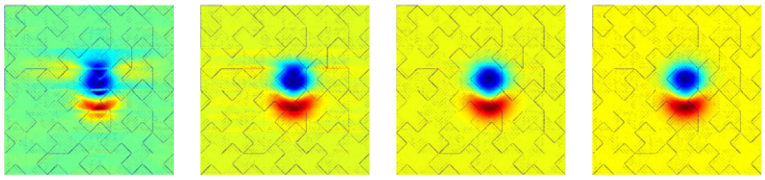

(a) 20 iterations. (b) 40 iterations. (c) 60 iterations. (d) ρu exact.

Fig. 11. Solution of the Euler equation using the oDPG method with $p = 4$ and $\Delta p = 2$: Fig. 11(a) is ρu after 20 iterations, Fig. 11(b) after 40 iterations, and Fig. 11(c) after 60 iterations, and Fig. 11(d) is the exact ρu.

For the numerical example, we choose the forcing function **f** so that the exact solution is the following vortex

$$
\begin{aligned}
\rho &= \left[1 - \frac{(\gamma-1)}{16\gamma\pi^2}\beta^2 e^{2(1-R^2)}\right]^{\frac{1}{\gamma-1}}, \\
u &= 1 - \beta e^{(1-R^2)}\frac{(y-y_0)}{2\pi}, \\
v &= \beta e^{(1-R^2)}\frac{(x-x_0)}{2\pi}, \\
P &= \rho^\gamma,
\end{aligned}
$$

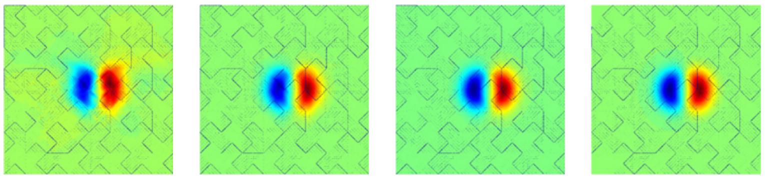
with $R^2 = (x - 0.5 - x_0)^2 + (y - y_0)^2$, $x_0 = 0.5$, $y_0 = 0$, and $\beta = 5$.

For Euler equations, we observe that $\Delta p = 2$ gives significantly more accurate results than $\Delta p = 1$ while $\Delta p \geq 3$ yields marginally better results than those with $\Delta p = 2$. Thus, to the end of this section, we take $\Delta p = 2$. The solution order is chosen to be $p = 4$. In order to challenge our trust region optimization solver, the initial guess is specified as the following

$$
\begin{aligned}
&\rho = 1, && \widehat{\rho} = 1, \\
&\rho u = 1, && \widehat{\rho u} = 1, \\
&\rho v = 0, && \widehat{\rho v} = 0, \\
&P = \rho^\gamma, && \widehat{P} = \widehat{\rho}^\gamma, \\
&E = P/(\gamma-1) + 0.5(\rho u^2 + \rho v^2)/\rho, && \widehat{E} = \widehat{P}/(\gamma-1) + 0.5(\widehat{\rho u}^2 + \widehat{\rho v}^2)/\widehat{\rho},
\end{aligned}
$$

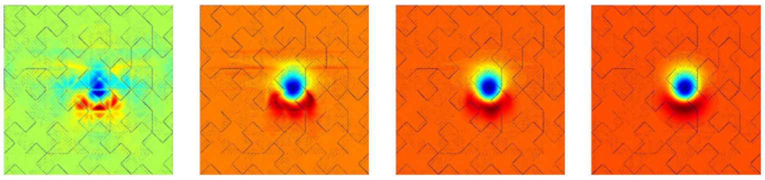
which is by no means "close" to the exact solution. Similar to other examples, we enforce Dirichlet boundary on the entire boundary $\partial\Omega$ by setting $\widehat{\rho}$, $\widehat{\rho u}$, $\widehat{\rho v}$, $\widehat{E}$ equal to the exact solutions at all times. Figs. 10–13 show the solutions ρ, ρu, ρv, E after 20, 40, and 60 iterations. It can be seen, though the initial flow is horizontal, the computed flow starts to roll up towards to the exact solution after 20 iterations. The solutions after 40 iterations are very similar to the exact solutions while those after 60 iterations are almost identical to the exact ones.

For the Euler example, Fig. 14 shows that the oDPG solutions converge towards the exact solution very fast initially, e.g., for the first 60 Newton iterations, but they converge very slowly after that. In particular, the cost function, namely the residual, decreases very slowly after 60 Newton iterations. Indeed, the tolerance for relative cost change of $\varepsilon_F = 10^{-14}$ is met at the 159th iteration though the ℓ^2-norm of the discrete gradient is $1.348e - 6$. Furthermore, above 155 Newton iterations, Newton steps are full but quite small (on the order of $\mathcal{O}(10^{-2})$). This suggests that the



(a) 20 iterations. (b) 40 iterations. (c) 60 iterations. (d) ρv exact.

Fig. 12. Solution of the Euler equation using the oDPG method with $p = 4$ and $\Delta p = 2$: Fig. 12(a) is ρv_h after 20 iterations, Fig. 12(b) is ρv_h after 40 iterations, and Fig. 12(c) is ρv_h after 60 iterations, and Fig. 12(d) is the exact ρv.



(a) 20 iterations. (b) 40 iterations. (c) 60 iterations. (d) E exact.

Fig. 13. Solution of the Euler equation using the oDPG method with $p = 4$ and $\Delta p = 2$: Fig. 13(a) is E_h after 20 iterations, Fig. 13(b) is E_h after 40 iterations, Fig. 13(c) is E_h after 60 iterations, and Fig. 13(d) is the exact E.

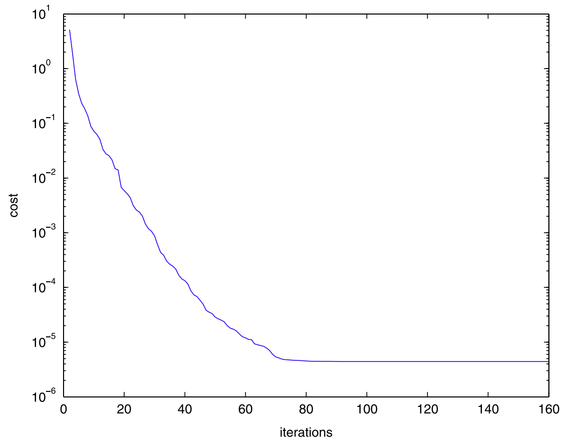


Fig. 14. Solution of the Euler equation using the oDPG method with $p = 4$ and $\Delta p = 2$: history of cost function values.

optimization problem may be difficult, i.e., the cost functional be quite flat. However, how to address this difficulty is beyond the scope of the paper.

10. Conclusions

We have presented a PDE-constrained optimization approach to the DPG framework of Demkowicz and Gopalakrishnan [1,2]. The proposed approach can be considered as a Rayleigh–Ritz approach for the DPG minimum residual statement. It is equipped with a trust region inexact Newton conjugate gradient method which prevents over-solving when optimization iterates are far away from the optimal solution but converges quadratically otherwise. The PDE-constrained approach is promising since it allows one to solve the DPG framework in an iterative manner using any of the state-of-the-art PDE-constrained techniques, and hence in principle enabling the user to solve large-scale and difficult (nonlinear) problems efficiently. We have used the viscous Burger equation as the model to derive the PDE-constrained DPG approach and showed that the proposed approach gives the same solution as that of the standard Euler–Lagrange DPG counterpart in exact arithmetic. In practice, our proposed approach neither solves the optimal test function equation or the original PDEs directly. However, it does require one to discretize the original PDE since the resulting residual is the forcing for the forward equation. We have showed that the adjoint equation turns out to be the same as the forward one, and hence eliminated. We also derive the gradient and Hessian-vector product for both abstract variational framework and the viscous Burger equation. The results show that the computing gradient and a Hessian-vector product using the adjoint technique are scalable and efficient since these tasks are local on elements or edges with linear complexity in total. Numerical results show that optimality in both h- and p-convergences is attainable for linear PDE such as Laplace and Helmholtz equations when the tolerances are set to be sufficiently small. We have also applied our method on nonlinear PDEs including the viscous Burger equation and the Euler equation of gas dynamics, and the results are very encouraging.

Our ongoing work aims at addressing the following research directions:

- We have observed that one order of h-convergence is lost for the viscous Burger equation. It could be due to the nonlinearity though we have not yet uncovered the truth. Furthermore, it is quite challenging to solve the Euler equation and this is perhaps due to the flatness of the cost function. In particular, the full Newton step is taken but essentially there is negligible improvement in the cost function. Work is in progress to address these issues.
- We have showed that the complexity of computing gradient or a Hessian-vector product scales linearly in the number of DPG unknowns N. We also observe that the number of Newton iterations is independent of mesh and solution order refinements.[3] However, for each Newton iteration, the number of CG iterations can be up to N. In fact, our numerical experiences indicate that this happens and in this case the corresponding Newton iteration requires $\mathcal{O}\left(N^2\right)$ operations, which is quite expensive. Ongoing research is to design a preconditioner so that the number of CG iterations is independent of N as well. This is very desirable since the total complexity of TRINCG would scale linearly in N in this case. Clearly, first forming the Hessian matrix and then designing a preconditioner, i.e. incomplete LU decomposition, would defeat the matrix-free nature of the method. On the other hand, the challenge here is how to construct an efficient preconditioner for our matrix-free method.
- There are a few aspects that are not addressed in the paper for the sake of simplicity. First, the domain under consideration is very simple, e.g., with polygon boundary, though our framework is applicable for domains with curved boundaries. Second, time-dependent problems are not treated. One simple approach (see, e.g., [10]) is first to perform implicit time discretization and then to apply our oDPG method. Last, but not least, we have carried out numerical simulations for two-dimensional problems in Matlab though our PDE-constrained DPG framework should be reasonably straightforward to be applied to large-scale three-dimensional problems similar to those in [24].

References

[1] Leszek Demkowicz, Jay Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part I: the transport equation, Comput. Methods Appl. Mech. Engrg. 199 (2010) 1558–1572.

[2] Leszek Demkowicz, Jay Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part II: optimal test functions, Numer. Methods Partial Differential Equations 27 (2011) 70–105.

[3] Tan Bui-Thanh, Leszek Demkowicz, Omar Ghattas, Constructively well-posed approximation method with unity inf–sup and continuity constants for partial differential equations, Math. Comput. 82 (2013) 1923–1952.

[4] Leszek Demkowicz, Jay Gopalakrishnan, Analysis of the DPG method for the Poisson equation, SIAM J. Numer. Anal. 49 (2011) 1788–1809.

[5] Leszek Demkowicz, Jay Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part IV: the optimal test norm and time-harmonic wave propagation in 1D, J. Comput. Phys. 230 (2011) 2406–2432.

[6] Jeff Zitelli, Ignacio Muga, Leszek Demkowicz, Jay Gopalakrishnan, David Pardo, Victor M. Calo, A class of discontinuous Petrov-Galerkin methods. Part IV: wave propagation, J. Comput. Phys. 230 (2011) 2046–2432.

[7] Leszek Demkowicz, Jay Gopalakrishnan, Ignacio Muga, Jeff Zitelli, Wave number explicit analysis for a DPG method for the multi-dimensional Helmholtz equation, Comput. Methods Appl. Mech. Engrg. 213 (2012) 126–138.

[8] Jesse Chan, Leszek Demkowicz, Robert Moser, Nate Roberts, A new discontinuous Petrov–Galerkin method with optimal test functions. Part V: solutions of 1D Burger and Navier–Stokes equations, Tech. Report 10–25, ICES, UT Austin, June 2010.

[9] Jamie Bramwell, Leszek Demkowicz, Jay Gopalakrishnan, Weifeng Qiu, A locking-free hp DPG method for linear elasticity with symmetric stresses, Numer. Math. 122 (2012) 671–707.

[10] Tan Bui-Thanh, Leszek Demkowicz, Omar Ghattas, A unified discontinuous Petrov-Galerkin method and its analysis for Friedrichs' systems, SIAM J. Numer. Anal. 51 (2013) 1933–1958.

[11] C.L. Bottasso, S. Micheletti, R. Sacco, The discontinous Petrov–Galerkin method for elliptic problems, Comput. Methods Appl. Mech. Eng. 191 (2002) 3391–3409.

[3] We did not perform the mesh independent property of the TRINCG approach for the Euler equation due to the above convergent difficulty.

[12] C.L. Bottasso, S. Micheletti, R. Sacco, A multiscale formulation of the discontinuous Petrov–Galerkin method for advective–diffusive problems, Comput. Methods Appl. Mech. Engrg. 194 (2005) 2819–2838.

[13] P. Causin, R. Sacco, A discontinous Petrov–Galerkin method with Lagrangian multipliers for second order elliptic problems, SIAM J. Numer. Anal. 43 (2005) 280–302.

[14] Nate Roberts, Tan Bui-Thanh, Leszek Demkowicz, The DPG method for the Stokes problem, Comput. Math. Appl. 67 (2014) 966–995.

[15] P. Bochev, M.D. Gunzburger, Least-Squares Finite Element Methods, in: Applied Mathematical Sciences, vol. 166, Spinger-Verlag, 2009.

[16] R. Cai, R. Lazarov, T.A. Manteuffel, S.F. McCormick, First-order system least squares for second-order partial differential equations, SIAM J. Numer. Anal. 31 (1994) 1785–1799.

[17] Jesse Chan, Norbert Heuer, Tan Bui-Thanh, Leszek Demkowicz, Robust DPG method for convection-dominated diffusion problems II: a natural inflow condition, Comput. Math. Appl. 67 (2014) 771–795.

[18] A.T. Barker, S.C. Brenner, E.-H. Park, L.-Y. Sung, Domain decomposition preconditioners for discontinuous Galerkin methods, Tech. Report, Dept. of Math. Luisiana State University, 2013, in preparation.

[19] S.C. Brenner, E.-H. Park, L.-Y. Sung, A balancing domain decomposition by constraints preconditioner for a discontinuous Galerkin method, Tech. Report, Dept. of Math. Luisiana State University, 2013, in preparation.

[20] D. Moro, N.C. Nguyen, J. Peraire, A hybridized discontinuous Petrov-Galerkin scheme for scalar conservation laws, Internat. J. Numer. Methods Engrg. 91 (2012) 950–970.

[21] Jorge Nocedal, Stephen J. Wright, Numerical Optimization, second ed., Springer Verlag, Berlin, Heidelberg, New York, 2006.

[22] Max D. Gunzburger, Perspectives in Flow Control and Optimization, SIAM, Philadelphia, 2003.

[23] Leszek Demkowicz, Norbert Heuer, Robust DPG method for convection-dominated diffusion problems, SIAM J. Numer. Anal. 51 (2013) 2514–2537.

[24] Tan Bui-Thanh, Carsten Burstedde, Omar Ghattas, James Martin, Georg Stadler, Lucas C. Wilcox, Extreme-scale UQ for Bayesian inverse problems governed by PDEs, in SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2012. Gordon Bell Prize finalist.