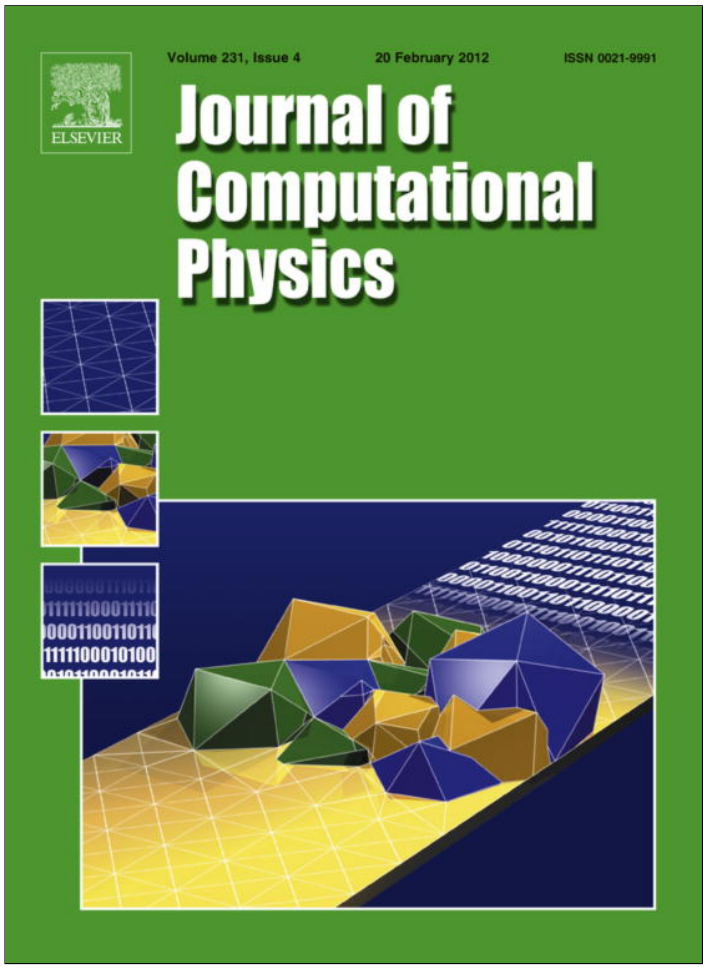

Journal of Computational Physics 231 (2012) 1230–1243

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Recovery of normal derivatives from the piecewise L^2 projection

Yunqing Huang [a], Hailiang Liu [b,*], Nianyu Yi [a]

[a] Hunan Key Laboratory for Computation and Simulation in Science and Engineering, School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, PR China
[b] Iowa State University, Mathematics Department, Ames, IA 50011, United States

ARTICLE INFO

Article history:
Received 18 January 2011
Received in revised form 7 June 2011
Accepted 2 October 2011
Available online 19 October 2011

Keywords:
Recovery
L^2 projection
DDG methods
Numerical flux

ABSTRACT

In this paper, we propose a novel approach to recover normal derivatives for a smooth function based on its piecewise L^2 projection. For each polynomial (of degree up to 4) projection, another polynomial of same degree (at least degree 1) is constructed over a subdomain centered at the interface separating two polynomials, its normal derivative at interface is taken to be the recovered normal derivative. The pointwise accuracy of the recovery is shown to be of order $2\left[\frac{m+1}{2}\right]+2$. From such a recovery algorithm we obtain a set of numerical flux formulae for solution derivatives. Following the direct discontinuous Galerkin (DDG) method introduced by Liu and Yan [H. Liu, J. Yan, The direct discontinuous Galerkin (DDG) method for diffusion with interface corrections, Commun. Comput. Phys. 8 (3) (2010) 541–564] for parabolic equations, we apply these flux formulae to some elliptic problems using polynomial elements of degree up to 4. Some adaptation of these numerical fluxes is adopted for even high order elements. Both one and two-dimensional numerical results are provided to demonstrate the good qualities of the recovery algorithm when combined with the DDG methods.

1. Introduction

This work is concerned with recovery of normal derivatives for a smooth function based on its piecewise L^2 projection. Here normal refers to the normal direction of the interface (for example, element nodal points in one dimension or element edges in two dimension) separating two polynomials. The problem can be formulated as follows: let u be a smooth function on a domain I consisting of two sub-domains $I^{\pm}$ with an interface $K=\partial I^{-}\cap\partial I^{+}$, and its L^2 projection on $I^{\pm}$, respectively. We want to recover the normal derivative $R[u_n]$ on K so that

$$\max_K\left|\frac{\partial u}{\partial n}-R[u_n]\right|=O(h^r),$$

for some $r>0$. Here h is the size of the partitioned domain. The challenge of this problem lies in that the two projected polynomials are not necessarily continuous on K, so jumps of their derivatives may be significant in the recovery of u_n.

This problem, interesting in its own sake, is mainly motivated by its potential application in the recently developed DDG (Direct Discontinuous Galerkin) methods [16,17] for diffusion problems. To illustrate the idea, we look at the linear elliptic equation

$$-\Delta u=f,$$

* Corresponding author.
E-mail addresses: huangyq@xtu.edu.cn (Y. Huang), hliu@iastate.edu (H. Liu), yinianyu365109@126.com (N. Yi).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.10.001

in a bounded domain Ω, subject to a proper boundary condition. If the domain is partitioned into computational cells $\Omega = \cup I_j$, then the DDG method proposed in [17] is to find a numerical solution so that in each computational cell I_j it satisfies

$$\int_{I_j} \nabla u \cdot \nabla v dx - \int_{\partial I_j} \hat{u}_n v ds + \frac{1}{2} \int_{\partial I_j} (u^{ext} - u) v_n ds = \int_{I_j} v f dx, \tag{1.1}$$

for any test functions in the same finite dimensional space as for the numerical solution. Here u^{ext} is the numerical solution in the neighboring cell. This scheme is complete as long as a formula for $\hat{u}_n$ is provided. In [16,17] a general consistent numerical flux ansatz is proposed, followed by an admissibility concept with which one can verify whether a selected numerical flux leads to a stable scheme. Several flux formulae are identified in [16] by using symmetric pointwise interpolation and further used in [17] for the refined DDG method. In this work we present a different recovery, yet consistent with the L^2-projection. We illustrate the capacity of our recovery algorithm through the DDG method (1.1) with

$$\hat{u}_n = R[u_n].$$

Our recovery algorithm is for C_{-1} element or discontinuous polynomials with a target application in DDG methods to solve diffusion problems. In contrast, gradient recovery for C_0 finite element has been well developed in the context of finite element methods, see, e.g. [7,13,14,22,24,25], and the references cited therein. The main difference is that our recovery is to provide a way to formulate the numerical flux for the DDG method, instead of a post-processing technique, but finite element gradient recovery methods are post-processing techniques that reconstruct improved gradient approximations from finite element solutions as well as to explore their use in adaptive computations. Therefore, the new method cannot be judged inferior or superior through a comparison with the finite element gradient recovery methods.

The choice of discontinuous finite elements empowers the DG (Discontinuous Galerkin) method with more attractive features as have been observed since the work [19] for neutron transport equations, yet one often has to identify an appropriate numerical flux to make it work. This issue becomes subtly difficult for diffusion problems, see, e.g. [15,21]. There have been several DG methods suggested in literature to solve the problem, including the method originally proposed by Bassi and Rebay [6] for compressible Navier–Stokes equations, its generalization called the local discontinuous Galerkin (LDG) method introduced in [9] by Cockburn and Shu, the method by Baumann–Oden [5,18], the DDG method by Liu and Yan [16,17], and the methods introduced in [10,12,15], as well as the hybridizable discontinuous Galerkin (HDG) method recently introduced by Cockburn et al. [8]. Also in the 1970s, Galerkin methods for elliptic and parabolic problems using discontinuous finite elements, called the interior penalty (IP) methods, were independently introduced and studied; see, e.g. [1,3,23]. For analysis of DG methods for elliptic problems, we refer to [2,4] and references therein.

Using recovery based DG methods for diffusion began with van Leer and Norman [15]. In particular, van Raalte and van Leer derived bilinear forms for the recovery-based discontinuous Galerkin method introduced in [20], where they introduced the concept of a local "recovery polynomial basis". For each piecewise continuous polynomial of degree m defined on two adjacent cells, they construct a continuous polynomial of degree $2m + 1$ and then compute its value and derivative at the cell interface. Its drawback is the high complexity when higher order polynomials are used. In [11], Franz et al. proposed a postprocessing recovery technique for discontinuous Galerkin methods by projecting a discontinuous, piecewise polynomial solution into a higher order polynomial space on a macro mesh, and then improve the solution accuracy.

The recovery algorithm presented in this work when applied to the DDG method [17] offers a class of simple, easy to implement DG methods for elliptic problems. Illustration of this will be the main goal of this work. The rest of the paper is organized as follows: in Section 2, we describe the formulation of recovery algorithm for the one dimensional case, followed by a discussion of optimality of the recovery and several extensions. The corresponding set of numerical flux formulae is thus derived. Numerical results of both one and two dimensions are presented in Section 3. Finally, in Section 4, concluding remarks are given.

2. The recovery algorithm

For simplicity, we present recovery algorithms based on a one-dimensional master domain $I = [-h, h]$ with mesh size h. Set $I^- = [-h, 0]$, $I^+ = [0, h]$. Let u be a smooth function in I, and $\Pi_m u = p_1 \in P^m(I^-)$ and $\Pi_m u = p_2 \in P^m(I^+)$ be the standard L^2 projection of u in I^- and I^+, respectively. More precisely,

$$\int_{I^-} (p_1(x) - u(x)) v(x) dx = 0, \quad \forall v \in P^m(I^-)$$

and

$$\int_{I^+} (p_2(x) - u(x)) v(x) dx = 0, \quad \forall v \in P^m(I^+).$$

Our goal is to recover $u_x(0)$ from only $p_1(x)$ and $p_2(x)$. The idea is to identify a sub-domain $I_k = [-kh, kh]$ for some $k \in (0, 1]$ such that a polynomial of same degree can be recovered, based on which the interface derivative follows. The algorithm goes as follows.

Recovery algorithm: Let $\Pi_m u$ be the piecewise L^2 projection of u defined above, we find $p \in P^{\max\{1,m\}}(I_k)$ such that

$$\int_{I_k} p v dx = \int_{I_k} \Pi_m u v dx, \quad \forall v \in P^{\max\{1,m\}}(I_k). \tag{2.1}$$

We then difine the recovered interface derivative by

$$R[u_x](0) := p'(0). \tag{2.2}$$

Existence of such a sub-domain and the recovery accuracy is summarized in the following.

Theorem 2.1. *Let u be a smooth function in $[-h,h]$, its L^2 projection on $[-h,0]$ be $\Pi_m u = p_1$ and $\Pi_m u = p_2$ on $[0,h]$. For each m with $0 \leqslant m \leqslant 4$, there exists $k \in (0,1)$ such that $u_x(0)$ can be recovered on $I_k = [-kh,kh]$ via recovery algorithm (2.1) and (2.2). Moreover,*

$$|R[u_x](0) - u_x(0)| = O(h^r), \quad r = 2\left[\frac{m+1}{2}\right] + 2. \tag{2.3}$$

Proof. Let $u_0 = u(0)$, $u_d(x) = u(x) - u(-x)$, by *Taylor* expansion,

$$u_d(x) = 2xu_0' + \frac{x^3}{3}u_0''' + \frac{x^5}{60}u_0^{(5)} + \frac{x^7}{2520}u_0^{(7)} + \cdots = 2\sum_{i=0}^{\infty} \frac{x^{2i+1}}{(2i+1)!}u_0^{(2i+1)}. \tag{2.4}$$

For $m = 0$, $p_1 = a_1 \in P^0(I^-)$, $p_2 = a_2 \in P^0(I^+)$, with

$$a_1 = \frac{1}{h}\int_{-h}^{0} u dx, \quad a_2 = \frac{1}{h}\int_{0}^{h} u dx.$$

From (2.1) and (2.2), we obtain

$$R[u_x](0) = \frac{3}{4k}\frac{a_2 - a_1}{h} = \frac{3}{4kh^2}\int_0^h u_d dx = \frac{3}{4k}u_0' + \frac{h^2}{16k}u_0''' + O(h^4). \tag{2.5}$$

Consistency requires $k = \frac{3}{4}$, so that

$$R[u_x](0) = u_0' + \frac{h^2}{12}u_0''' + O(h^4),$$

which approximates u_0' with second order accuracy.

Using the notations

$$[a] = a_2 - a_1, \quad \bar{a} = \frac{a_1 + a_2}{2},$$

we proceed to the case $m = 1$, for which, $p_1 = a_1 + b_1(x + \frac{h}{2})$, $p_2 = a_2 + b_2(x - \frac{h}{2})$, where

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & 0 \\ \frac{6}{h^2} & \frac{12}{h^3} \end{bmatrix}\begin{bmatrix} \int_{-h}^0 u dx \\ \int_{-h}^0 ux dx \end{bmatrix}, \quad \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & 0 \\ -\frac{6}{h^2} & \frac{12}{h^3} \end{bmatrix}\begin{bmatrix} \int_{0}^h u dx \\ \int_{0}^h ux dx \end{bmatrix}.$$

From (2.1) and (2.2) it follows that

$$R[u_x](0) = \frac{3}{4k}\frac{[a]}{h} + \left(1 - \frac{3}{4k}\right)\bar{b} = \left(\frac{3}{kh^2} - \frac{3}{h^2}\right)\int_0^h u_d dx + \left(\frac{6}{h^3} - \frac{9}{2kh^3}\right)\int_0^h u_d x dx = u_0' + h^2\left(\frac{3}{20} - \frac{1}{20k}\right)u_0''' + O(h^4). \tag{2.6}$$

For any k this yields a recovery of second order accuracy. An improved accuracy is possible only when $k = \frac{1}{3}$, leading to

$$R[u_x](0) = u_0' + O(h^4).$$

For $m = 2$, $p_1 = a_1 + b_1(x + \frac{h}{2}) + c_1(x + \frac{h}{2})^2$, $p_2 = a_2 + b_2(x - \frac{h}{2}) + c_2(x - \frac{h}{2})^2$ with

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2h} & -\frac{15}{h^2} & -\frac{15}{h^3} \\ \frac{6}{h^2} & \frac{12}{h^3} & 0 \\ \frac{30}{h^3} & \frac{180}{h^4} & \frac{180}{h^5} \end{bmatrix}\begin{bmatrix} \int_{-h}^0 u dx \\ \int_{-h}^0 ux dx \\ \int_{-h}^0 ux^2 dx \end{bmatrix}, \quad \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2h} & \frac{15}{h^2} & -\frac{15}{h^3} \\ -\frac{6}{h^2} & \frac{12}{h^3} & 0 \\ \frac{30}{h^3} & -\frac{180}{h^4} & \frac{180}{h^5} \end{bmatrix}\begin{bmatrix} \int_{0}^h u dx \\ \int_{0}^h ux dx \\ \int_{0}^h ux^2 dx \end{bmatrix}.$$

Recovery (2.1) and (2.2) gives

$$\begin{aligned} R[u_x](0) &= \frac{3}{4k}\frac{[a]}{h} + \left(1 - \frac{3}{4k}\right)\bar{b} + \left(\frac{3}{8}kh - \frac{h}{2} + \frac{3h}{16k}\right)[c] \\ &= \left(\frac{54}{8kh^2} - \frac{18}{h^2} + \frac{45k}{4h^2}\right)\int_0^h u_d dx + \left(\frac{96}{h^3} - \frac{27}{kh^3} - \frac{135k}{2h^3}\right)\int_0^h u_d x dx + \left(\frac{45}{2kh^4} - \frac{90}{h^4} + \frac{135k}{2h^4}\right)\int_0^h u_d x^2 dx \\ &= u_0' + h^2\left(\frac{1}{80k} - \frac{1}{10} + \frac{3k}{16}\right)u_0''' + h^4\left(\frac{3}{2240k} - \frac{1}{112} + \frac{5k}{448}\right)u_0^{(5)} + O(h^6). \end{aligned} \tag{2.7}$$

In order for this recovery to approximate u_0' with 4th-order accuracy, it suffices to take k such that

$$\frac{1}{80k}-\frac{1}{10}+\frac{3k}{16}=0,$$

which holds for $k=\frac{1}{3}$ or $k=\frac{1}{5}$, in either case we have

$$R[u_x](0)=u_0'+O(h^4).$$

For $m=3$, $p_1=a_1+b_1\left(x+\frac{h}{2}\right)+c_1\left(x+\frac{h}{2}\right)^2+d_1\left(x+\frac{h}{2}\right)^3$, $p_2=a_2+b_2\left(x-\frac{h}{2}\right)+c_2\left(x-\frac{h}{2}\right)^2+d_2\left(x-\frac{h}{2}\right)^3$, the L^2 projection links these to the moments of function u as follows

$$\begin{bmatrix}a_1\\ b_1\\ c_1\\ d_1\end{bmatrix}=\begin{bmatrix}-\frac{3}{2h} & -\frac{15}{h^2} & -\frac{15}{h^3} & 0\\ -\frac{15}{h^2} & -\frac{240}{h^3} & -\frac{630}{h^4} & -\frac{420}{h^5}\\ \frac{30}{h^3} & \frac{180}{h^4} & \frac{180}{h^5} & 0\\ \frac{140}{h^4} & \frac{1680}{h^5} & \frac{4200}{h^6} & \frac{2800}{h^7}\end{bmatrix}\begin{bmatrix}\int_{-h}^{0}u\,dx\\ \int_{-h}^{0}ux\,dx\\ \int_{-h}^{0}ux^2dx\\ \int_{-h}^{0}ux^3dx\end{bmatrix}$$

and

$$\begin{bmatrix}a_2\\ b_2\\ c_2\\ d_2\end{bmatrix}=\begin{bmatrix}-\frac{3}{2h} & \frac{15}{h^2} & -\frac{15}{h^3} & 0\\ \frac{15}{h^2} & -\frac{240}{h^3} & \frac{630}{h^4} & -\frac{420}{h^5}\\ \frac{30}{h^3} & -\frac{180}{h^4} & \frac{180}{h^5} & 0\\ -\frac{140}{h^4} & \frac{1680}{h^5} & -\frac{4200}{h^6} & \frac{2800}{h^7}\end{bmatrix}\begin{bmatrix}\int_{0}^{h}u\,dx\\ \int_{0}^{h}ux\,dx\\ \int_{0}^{h}ux^2dx\\ \int_{0}^{h}ux^3dx\end{bmatrix},$$

respectively. From (2.1) and (2.2) we obtain

$$\begin{aligned}R[u_x](0)&=\frac{45}{32kh}[a]+\left(1-\frac{45}{32k}\right)\bar{b}+\left(\frac{5}{32}kh-\frac{1}{2}h+\frac{45}{128k}h\right)[c]+\left(-\frac{15}{32}kh^2+\frac{3}{4}h^2-\frac{45}{128k}h^2\right)\bar{d}\\&=\left(\frac{45}{2kh^2}-\frac{60}{h^2}+\frac{75k}{2h^2}\right)\int_0^h u_d\,dx+\left(\frac{600}{h^3}-\frac{675}{4kh^3}-\frac{3375k}{8h^3}\right)\int_0^h u_d x\,dx\\&\quad+\left(\frac{675}{2kh^4}-\frac{1350}{h^4}+\frac{2025k}{2h^4}\right)\int_0^h u_d x^2dx+\left(\frac{840}{h^5}-\frac{1575}{8kh^5}-\frac{2625k}{4h^5}\right)\int_0^h u_d x^3dx\\&=u_0'+h^4\left(\frac{5}{1008}-\frac{1}{1344k}-\frac{25k}{4032}\right)u_0^{(5)}+h^6\left(\frac{1}{4752}-\frac{1}{29568k}-\frac{k}{4224}\right)u_0^{(7)}+O(h^8).\end{aligned}\tag{2.8}$$

Letting the coefficient in the second term equal zero, we have

$$25k^2-20k+3=0.$$

That is, $k=\frac{1}{5}$ or $k=\frac{3}{5}$, in either case, we have

$$R[u_x](0)=u_0'+O(h^6).$$

For $m=4$,

$$p_1=a_1+b_1\left(x+\frac{h}{2}\right)+c_1\left(x+\frac{h}{2}\right)^2+d_1\left(x+\frac{h}{2}\right)^3+e_1\left(x+\frac{h}{2}\right)^4,$$

$$p_2=a_2+b_2\left(x-\frac{h}{2}\right)+c_2\left(x-\frac{h}{2}\right)^2+d_2\left(x-\frac{h}{2}\right)^3+e_2\left(x-\frac{h}{2}\right)^4,$$

where

$$\begin{bmatrix}a_1\\ b_1\\ c_1\\ d_1\\ e_1\end{bmatrix}=\begin{bmatrix}\frac{15}{8h} & \frac{105}{2h^2} & \frac{1155}{4h^3} & \frac{945}{2h^4} & \frac{945}{4h^5}\\ -\frac{15}{h^2} & -\frac{240}{h^3} & -\frac{630}{h^4} & -\frac{420}{h^5} & 0\\ -\frac{105}{h^3} & -\frac{2520}{h^4} & -\frac{11970}{h^5} & -\frac{18900}{h^6} & -\frac{9450}{h^7}\\ \frac{140}{h^4} & \frac{1680}{h^5} & \frac{4200}{h^6} & \frac{2800}{h^7} & 0\\ \frac{630}{h^5} & \frac{12600}{h^6} & \frac{56700}{h^7} & \frac{88200}{h^8} & \frac{44100}{h^9}\end{bmatrix}\begin{bmatrix}\int_{-h}^{0}u\,dx\\ \int_{-h}^{0}ux\,dx\\ \int_{-h}^{0}ux^2dx\\ \int_{-h}^{0}ux^3dx\\ \int_{-h}^{0}ux^4dx\end{bmatrix},$$

$$\begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \\ e_2 \end{bmatrix} = \begin{bmatrix} \frac{15}{8h} & -\frac{105}{2h^2} & \frac{1155}{4h^3} & -\frac{945}{2h^4} & \frac{945}{4h^5} \\ \frac{15}{h^2} & -\frac{240}{h^3} & \frac{630}{h^4} & -\frac{420}{h^5} & 0 \\ -\frac{105}{h^3} & \frac{2520}{h^4} & -\frac{11970}{h^5} & \frac{18900}{h^6} & -\frac{9450}{h^7} \\ -\frac{140}{h^4} & \frac{1680}{h^5} & -\frac{4200}{h^6} & \frac{2800}{h^7} & 0 \\ \frac{630}{h^5} & -\frac{12600}{h^6} & \frac{56700}{h^7} & -\frac{88200}{h^8} & \frac{44100}{h^9} \end{bmatrix} \begin{bmatrix} \int_0^h u dx \\ \int_0^h u x dx \\ \int_0^h u x^2 dx \\ \int_0^h u x^3 dx \\ \int_0^h u x^4 dx \end{bmatrix}.$$

From (2.1) and (2.2) it follows that

$$\begin{aligned} R[u_x](0) &= \frac{75}{8k^3h^3}\int_{I_k} \Pi_4 u x dx - \frac{105}{8k^5h^5}\int_{I_k} \Pi_4 u x^3 dx \\ &= \frac{45}{32kh}[a] + \left(1 - \frac{45}{32k}\right)\bar{b} + \left(\frac{45h}{128k} - \frac{h}{2} + \frac{5kh}{32}\right)[c] + \left(-\frac{45h^2}{128k} + \frac{3h^2}{4} - \frac{15kh^2}{32}\right)\bar{d} \\ &\quad + \left(\frac{45h^3}{512k} - \frac{h^3}{4} + \frac{15kh^3}{64} - \frac{5k^3h^3}{64}\right)[e] \\ &= u_0' + h^4 u_0^{(5)}\left\{\frac{1}{10752k} - \frac{1}{1008} + \frac{5}{2304}k - \frac{5}{1536}k^3\right\} \\ &\quad + h^6 u_0^{(7)}\left\{\frac{5}{473088k} - \frac{1}{9504} + \frac{7}{33792}k - \frac{35}{202752}k^3\right\} + O(h^8). \end{aligned} \tag{2.9}$$

In order to achieve a recovery with 6th order accuracy, it suffices to take k so that

$$\frac{1}{10752k} - \frac{1}{1008} + \frac{5}{2304}k - \frac{5}{1536}k^3 = 0.$$

The only real root is $k = \frac{35 - 2\sqrt[3]{1225}}{105} \approx \frac{719}{5551}$, with which

$$R[u_x](0) = u_0' + \frac{59142787625}{20871851348343996}h^6 u_0^{(7)} + O(h^8). \qquad \square$$

Remark 2.1. The above result indicates the following

- Better accuracy is obtained for recovery polynomials of odd degree.
- k is not unique to achieve same accuracy. The sub-domain tends to be more compact for higher order projection.
- For recovery of derivatives at a point (on interface), only a partial set of test functions is needed. For example, in the case $m = 0,1,2$ only $v = \{x\}$ is actually used, and the recovery (2.1) and (2.2) can be simplified as

$$R(u_x)(0) = \frac{\int_{I_k} x \Pi_m u dx}{\int_{I_k} x^2 dx}. \tag{2.10}$$

- For $m = 3$ and $m = 4$, the proposed calculation is equivalent to the following formula

$$R[u_x](0) = \frac{\left(\int_{I_k} x \Pi_m u dx\right)\left(\int_{I_k} x^6 dx\right) - \left(\int_{I_k} x^3 \Pi_m u dx\right)\int_{I_k} x^4 dx}{\int_{I_k} x^2 dx \int_{I_k} x^6 dx - \left(\int_{I_k} x^4 dx\right)^2}, \tag{2.11}$$

which only uses two test functions $v = \{x, x^3\}$.

2.1. Optimality

For $m \geqslant 2$ there are more $k's$ to obtain the same accuracy of the recovery. A natural question is which k is an optimal choice. The recovery formula can be expressed as

$$R[u_x](0) = u_0' + \Phi(k) u_0^{(r)} h^{r-1},$$

where $\Phi(k)$ is the coefficient depending on k. We shall take the recovery such that $|\Phi(k)|$ is smaller. In fact, for $m = 2$,

$$\Phi(k) = \frac{3}{2240k} - \frac{1}{112} + \frac{5k}{448},$$

hence

$$\left|\Phi\left(\frac{1}{5}\right)\right| = 0, \quad \left|\Phi\left(\frac{1}{3}\right)\right| = \frac{1}{840}.$$

For $m = 3$,

$$\Phi(k) = \frac{1}{4752} - \frac{1}{29568k} - \frac{k}{4224},$$

then we have

$$\left|\Phi\left(\frac{1}{5}\right)\right| = \frac{1}{166320}, \quad \left|\Phi\left(\frac{3}{5}\right)\right| = \frac{1}{83160}.$$

In summary, we list the choice of optimal k for piecewise L^2 projection in Table 1.

Our numerical tests also show that the recovery scheme with $k = \frac{1}{5}$ produces smaller errors for $m = 2,3$.

2.2. Numerical flux

In order to apply the above recovery result to the DDG method (1.1), we shall replace the numerical flux $\hat{u}_x$ by the recovered derivatives, i.e.,

$$\hat{u}_x(0) = R[u_x](0).$$

Recall in [17] that for consistency the numerical flux necessarily has the following form

$$\widehat{u_x}(0) = \beta_0 \frac{[u]}{h} + \overline{u_x} + \sum_{i=1}^{[m/2]} \beta_i h^{2i-1} \left[\partial_x^{2i} u\right]. \tag{2.12}$$

Here the following notations have been used:

$$u^{\pm} = u(x \pm 0), \quad [u] = u^+ - u^-, \quad \bar{u} = \frac{u^+ + u^-}{2}.$$

In order to apply the above recovered derivatives to the DDG method, we now reformulate them into the form of numerical flux (2.12).

We assume that two projections are of the form

$$u|_{I^-} = p_1(x + h/2), \quad u|_{I^+} = p_2(x - h/2),$$

with $p_i(x) \in P^m(I_{1/2})$ for $i = 1,2$.

For $m = 0$, $p_i(x) = a_i$, hence $[u] = [a]$. From (2.5) it follows

$$\widehat{u_x}(0) := \frac{3(a_2 - a_1)}{4kh} = \frac{3}{4k}\frac{[u]}{h}.$$

For $k = \frac{3}{4}$, we have

$$\widehat{u_x}(0) = \frac{[u]}{h}. \tag{2.13}$$

For $m = 1$, we have $p_i(x) = a_i + b_i x$, then

$$[u] = p_2(-h/2) - p_1(h/2) = [a] - h\bar{b}, \quad \overline{u_x} = \bar{b}.$$

It follows from (2.6) that

$$\widehat{u_x}(0) := \frac{3}{4k}\frac{[u]}{h} + \overline{u_x}.$$

For $k = \frac{1}{3}$ we have

$$\widehat{u_x}(0) = \frac{9}{4}\frac{[u]}{h} + \overline{u_x}. \tag{2.14}$$

Table 1
The choice of k for p^m polynomials.

m	0	1	2	3	4
k	$\frac{3}{4}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{719}{5551}$

For $m = 2$, we have $p_i(x) = a_i + b_i x + c_i x^2$. A direct calculation gives

$$[u] = [a] - h\bar{b} + [c]\frac{h^2}{4}, \quad \overline{u_x} = \bar{b} - [c]\frac{h}{2}, \quad [u_{xx}] = 2[c].$$

From (2.7) we obtain

$$\widehat{u_x}(0) := \frac{3}{4k}\frac{[u]}{h} + \overline{u_x} + \frac{3}{16}kh[u_{xx}].$$

For $k = \frac{1}{5}$ we have

$$\widehat{u_x}(0) = \frac{15}{4}\frac{[u]}{h} + \overline{u_x} + \frac{3}{80}h[u_{xx}]. \tag{2.15}$$

For $m = 3$, we have $p_i(x) = a_i + b_i x + c_i x^2 + d_i x^3$, then

$$[u] = [a] - h\bar{b} + \frac{h^2}{4}[c] - \frac{h^3}{4}\bar{d}, \quad \overline{u_x} = \bar{b} - \frac{h}{2}[c] + \frac{3}{4}h^2\bar{d},$$
$$[u_{xx}] = 2[c] - 6h\bar{d}, \quad \overline{u_{xxx}} = 6\bar{d}.$$

By (2.8) we have

$$\widehat{u_x}(0) := \frac{45}{32k}\frac{[u]}{h} + \overline{u_x} + \frac{5}{64}kh[u_{xx}] + 0\overline{u_{xxx}}.$$

For $k = \frac{1}{5}$ we have

$$\widehat{u_x}(0) = \frac{225}{32}\frac{[u]}{h} + \overline{u_x} + \frac{1}{64}h[u_{xx}]. \tag{2.16}$$

For $m = 4$, $p_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 + e_i x^4$, then

$$[u] = [a] - h\bar{b} + \frac{h^2}{4}[c] - \frac{h^3}{4}\bar{d} + \frac{h^4}{16}[e], \quad \overline{u_x} = \bar{b} - \frac{h}{2}[c] + \frac{3}{4}h^2\bar{d} - \frac{h^3}{4}[e],$$
$$[u_{xx}] = 2[c] - 6h\bar{d} + 3h^2[e], \quad \overline{u_{xxx}} = 6\bar{d} - 6h[e], \quad [u_{xxxx}] = 24[e].$$

From (2.9) we obtain

$$\widehat{u_x}(0) := \frac{45}{32k}\frac{[u]}{h} + \overline{u_x} + \frac{5}{64}kh[u_{xx}] + 0\overline{u_{xxx}} - \frac{5}{1536}k^3h^3[u_{xxxx}].$$

For $k = \frac{719}{5551}$ we have

$$\widehat{u_x}(0) = \frac{32355}{177632}\frac{[u]}{h} + \overline{u_x} + \frac{3595}{355264}h[u_{xx}] - \frac{1858474795}{262727115495936}h^3[u_{xxxx}]. \tag{2.17}$$

The choices of the coefficients $\beta_0, \beta_1, \ldots, \beta_{[m/2]}$ with different k in numerical flux are summarized in Table 2.

2.3. Some extensions

It is clear that for polynomial elements of degree higher than 4, the corresponding recovery formula can become rather complicated. Hence we seek to exploit the recovered formulae listed above to the case with elements of higher degree. The idea is to adapt the obtained recovery formula, for example (2.16) with an adapted parameter,

$$\widehat{u_x} = \alpha \times \frac{225}{32}\frac{[u]}{h} + \overline{u_x} + \frac{1}{64}h[u_{xx}]. \tag{2.18}$$

In Example 3.4 of Section 3, we test the p^m, $m = 3,4,5$, approximation with different α values sampled from interval $[0,2]$. Our numerical results show that most of α gives the desired accuracy except for a few α in the middle of the sampled values.

The recovery algorithm can also be extended to non-uniform meshes and to multi-dimensional problems. For non-unifrom mesh in one-dimensional case, one may replace h by the average of meshes of two neighboring cells. In 2D case,

Table 2
The choice of β_i for p^m polynomials.

m	0	1	2	3	4
β_0	1	$\frac{9}{4}$	$\frac{15}{4}$	$\frac{225}{32}$	$\frac{32355}{177632}$
β_1	–	–	$\frac{3}{80}$	$\frac{1}{64}$	$\frac{3595}{355264}$
β_2	–	–	–	–	$-\frac{1858474795}{262727115495936}$

dimension by dimension extension can be used for rectangle mesh. For triangle mesh, $\hat{u}_n$ is replaced by the normal derivative on interface, and h is replaced by $l/2$ with l being the length of the interface.

3. Numerical examples

In this section we present a few numerical examples including one-dimensional and two-dimensional problems to show the accuracy of the DDG method with the obtained numerical flux.

Example 3.1. We solve the equation of the form

$$-u_{xx} + bu_x + cu = f, \quad u(0) = u(1) = 0 \tag{3.1}$$

in the domain $[0,1]$. We partition the domain $[0,1]$ into computational elements $I_j = (x_{j-1/2}, x_{j+1/2})$ with $x_{1/2} = 0$ and $x_{N+1/2} = 1$. The DDG scheme for this problem is to find $u \in P^m(I_j)$ such that

$$\frac{1}{2}[u](v_x)^-_{j+\frac{1}{2}} + \frac{1}{2}[u](v_x)^+_{j-\frac{1}{2}} - v\hat{u}_x|_{\partial I_j} + b\hat{u}v|_{\partial I_j} + \int_{I_j} u_x v_x dx - \int_{I_j} u(bv)_x dx + \int_{I_j} cuv dx = \int_{I_j} fv dx, \tag{3.2}$$

for all $v \in P^m(I_j)$. The numercal flux for $b\hat{u}$ is 'upwinding' as

$$b\hat{u} = \frac{b+|b|}{2}u^- + \frac{b-|b|}{2}u^+.$$

Boundary condition is built into the scheme in such a way that the boundary data are used when available, otherwise the value of the numerical solution in corresponding end cells will be used.

We consider a specific case with

$$b = 2, \quad c = -\left(1 + \frac{\pi^2}{4}\right), \quad f = -\pi \sin\frac{\pi x}{2} - \cos\frac{\pi x}{2}.$$

The exact solution is

$$u = \cos\frac{\pi x}{2}(1 - e^x).$$

Test case 1. For $m = 1$, the numerical flux $\widehat{u_x}$ is

$$\widehat{u_x} = \frac{9}{4}\frac{[u]}{h} + \overline{u_x}.$$

The results are reported in Table 3.

Test case 2. For $m = 2$, the numerical flux $\widehat{u_x}$ is

$$\widehat{u_x} = \frac{15}{4}\frac{[u]}{h} + \overline{u_x} + \frac{3}{80}h[u_{xx}].$$

The results are reported in Table 4.

Test case 3. For $m = 3$, the numerical flux $\widehat{u_x}$ is

$$\widehat{u_x} = \frac{225}{32}\frac{[u]}{h} + \overline{u_x} + \frac{1}{64}h[u_{xx}].$$

The results are reported in Table 5.

Test case 4. For $m = 4$, the numerical flux $\widehat{u_x}$ is

$$\widehat{u_x} = \frac{32355}{177632}\frac{[u]}{h} + \overline{u_x} + \frac{3595}{355264}h[u_{xx}] - \frac{1858474795}{262727115495936}h^3[u_{xxxx}].$$

The results are reported in Table 6.

In this example we test the performance of the recovery schemes for the DDG method with a one-dimensional problem. The DDG method based on p^m polynomial approximations with $m = 1,2,3,4$ are tested and $(m+1)$th order of accuracy is obtained.

Table 3
Errors for p^1 approximation.

$m = 1$	$N = 10$	$N = 20$		$N = 40$		$N = 80$		$N = 160$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u - u_h\|_{L^2}$	3.8117e−03	8.9959e−04	2.08	2.1576e−04	2.06	5.2603e−05	2.04	1.2970e−05	2.02
$\|u - u_h\|_{H^1}$	1.4410e−01	7.0238e−02	1.04	3.4629e−02	1.02	1.7190e−02	1.01	8.5633e−03	1.01

Table 4
Errors for p^2 approximation.

$m = 2$	$N = 10$	$N = 20$		$N = 40$		$N = 80$		$N = 160$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u - u_h\|_{L^2}$	1.0419e−04	1.3091e−05	2.99	1.6426e−06	2.99	2.0578e−07	3.00	2.5753e−08	3.00
$\|u - u_h\|_{H^1}$	4.1428e−03	1.0095e−03	2.04	2.4933e−04	2.02	6.1962e−05	2.01	1.5445e−05	2.00

Table 5
Errors for p^3 approximation.

$m = 3$	$N = 10$	$N = 20$		$N = 40$		$N = 80$		$N = 160$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u - u_h\|_{L^2}$	1.9116e−06	1.2768e−07	3.90	8.2475e−09	3.95	5.2409e−10	3.98	3.3027e−11	3.99
$\|u - u_h\|_{H^1}$	5.8872e−05	6.3846e−06	3.20	7.0354e−07	3.18	8.0377e−08	3.13	9.5033e−09	3.08

Example 3.2. We consider the same problem as the one in Example 3.1. Here the partition of the domain $[0,1]$ consists of a repeated pattern of $0.5\Delta x$ and $1.5\Delta x$ for odd and even numbers of indexes $i = 1,\ldots,N$, respectively, where $\Delta x = 1/N$ with even number N. We obtain similar results as those in Example 3.1. Errors and orders are reported in Table 7.

Example 3.3. We slove the same problem as the one in Example 3.1 with the adapted numerical flux in the DDG scheme (3.2),

$$\widehat{u_x} = 1.5 \times \frac{225}{32}\frac{[u]}{h} + \overline{u_x} + \frac{1}{64}h[u_{xx}]. \tag{3.3}$$

In this example, we test the DDG schemes with higher order polynomial approximations p^m with $m = 4,5,6,7$. The results are listed in Table 8. We obtain $(m+1)$th order accuracy for p^m approximations.

Example 3.4. We consider the same problem as the one in Example 3.1, using the following numerical flux in the DDG scheme (3.2),

$$\widehat{u_x} = \alpha \times \frac{225}{32}\frac{[u]}{h} + \overline{u_x} + \frac{1}{64}h[u_{xx}], \tag{3.4}$$

with a larger range of α.

Firstly, for the p^3 approximation we use the numerical flux (3.4) with different α values. Table 9 list the L^2 errors and orders. We observe that almost all α gives 4th order convergence for p^3 elements except $\alpha = 0.4, 0.5, 0.6, 0.7$.

Secondly, we investigate the p^4 approximation with numerical flux (3.4) and different α values. The L^2 errors and orders are reported in Table 10. Again, almost all α gives the desired convergence order for p^4 elements except $\alpha = 0.8, 0.9, 1.0, 1.2$.

Finally, we test the p^5 approximation. In Table 11 we list the L^2 errors and orders with different α values. It shows that almost all α gives 6th order convergence for p^5 elements except $\alpha = 1.1, 1.8$.

These tests show that a large range of α gives the desired accuracy, but the scheme indeed fails to deliver desired accuracy for some α.

Example 3.5. We solve the equation of the form

$$-\Delta u + u = f, \quad (x,y) \in \Omega = [0,1]^2, \quad u|_{\partial\Omega} = g. \tag{3.5}$$

Let a partition of Ω be denoted by rectangular meshes

Table 6
Errors for p^4 approximation.

$m = 4$	$N = 10$	$N = 20$		$N = 40$		$N = 80$		$N = 160$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u - u_h\|_{L^2}$	2.7735e−08	8.9032e−10	4.96	2.8200e−11	4.98	8.8720e−13	4.99	2.7807e−14	5.00
$\|u - u_h\|_{H^1}$	1.2740e−06	7.4590e−08	4.09	4.4924e−09	4.05	2.7527e−10	4.03	1.7030e−11	4.01

Table 7
Errors on non-uniform mesh, p^m approximations with $m = 0,1,2,3,4$.

m		$N = 10$	$N = 20$		$N = 40$		$N = 80$		$N = 160$	
		Error	Error	Order	Error	Order	Error	Order	Error	Order
1	$\|u-u_h\|_{L^2}$	1.0164e−02	2.5719e−03	1.98	6.4509e−04	2.00	1.6141e−04	2.00	4.0359e−05	2.00
	$\|u-u_h\|_{H^1}$	1.9835e−01	9.4851e−02	1.06	4.6291e−02	1.03	2.2859e−02	1.02	1.1358e−02	1.01
2	$\|u-u_h\|_{L^2}$	3.8217e−04	4.5306e−05	3.08	5.4633e−06	3.05	6.6880e−07	3.03	8.2665e−08	3.02
	$\|u-u_h\|_{H^1}$	8.6382e−03	2.0282e−03	2.09	4.9197e−04	2.04	1.2132e−04	2.02	3.0140e−05	2.01
3	$\|u-u_h\|_{L^2}$	7.7435e−06	5.5262e−07	3.81	3.6898e−08	3.90	2.3834e−09	3.95	1.5143e−10	3.98
	$\|u-u_h\|_{H^1}$	1.8374e−04	2.0729e−05	3.15	2.3493e−06	3.14	2.7362e−07	3.10	3.2746e−08	3.06
4	$\|u-u_h\|_{L^2}$	2.8482e−07	9.1665e−09	4.96	2.9050e−10	4.98	9.1395e−12	4.99	2.8646e−13	5.00
	$\|u-u_h\|_{H^1}$	7.0852e−06	3.8836e−07	4.19	2.2040e−08	4.14	1.2966e−09	4.09	7.8305e−11	4.05

$$\Omega = \sum_{j,k}^{N,M} I_{j,k}, \quad I_{j,k} = I_j \times I_k = \left[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}\right] \times \left[y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}\right] \tag{3.6}$$

of uniform mesh sizes $\Delta = \max\{\Delta x, \Delta y\}$.

The DDG scheme as introduced in [17] is to find $u|_{I_{j,k}} \in P^m$ such that

$$\iint_{I_{j,k}} (\nabla u \nabla v + u v) dx dy - \int_{I_k} \widehat{J}_1 v|_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} dy - \int_{I_j} \widehat{J}_2 v|_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} dx + B = \iint_{I_{j,k}} f v dx dy,$$

where

$$B = \frac{1}{2}\left\{ \int_{I_k} ([u]v_x^-)_{x_{j+\frac{1}{2}}} + ([u]v_x^+)_{x_{j-\frac{1}{2}}} dy + \int_{I_j} ([u]v_y^-)_{x_{k+\frac{1}{2}}} + ([u]v_y^+)_{x_{k-\frac{1}{2}}} dx \right\},$$
$$[u]_{x_{j+\frac{1}{2}}} := u\left(x_{j+\frac{1}{2}}^+, y\right) - u\left(x_{j+\frac{1}{2}}^-, y\right), \quad [u]_{y_{k+\frac{1}{2}}} := u\left(x, y_{k+\frac{1}{2}}^+\right) - u\left(x, y_{k+\frac{1}{2}}^-\right).$$

We shall test the case with $f = (8\pi^2 + 1)\sin 2\pi x \sin 2\pi y$ and $g = 0$, for which the exact solution is $u = \sin 2\pi x \sin 2\pi y$.

In this example we test the performance of the recovery schemes for the DDG method with a two-dimensional equation. Similar to 1D problems, $(m+1)$th order of accuracy is obtained with piecewise p^m polynomial approximations. The testes are performed on $N \times N$ rectangular mesh.

Test case 1. For $m = 1$, the numerical flux $\widehat{J}_i$ is

$$\widehat{J}_1|_{x_{j+\frac{1}{2}}} = \frac{9}{4}\frac{[u]}{\Delta x} + \overline{u_x},$$
$$\widehat{J}_2|_{y_{k+\frac{1}{2}}} = \frac{9}{4}\frac{[u]}{\Delta y} + \overline{u_y}.$$

The errors are reported in Table 12.

Test case 2. For $m = 2$, the numerical flux $\widehat{J}_i$ is

$$\widehat{J}_1|_{x_{j+\frac{1}{2}}} = \frac{15}{4}\frac{[u]}{\Delta x} + \overline{u_x} + \frac{3}{80}\Delta x[u_{xx}], \ \textit{and}$$
$$\widehat{J}_2|_{y_{k+\frac{1}{2}}} = \frac{15}{4}\frac{[u]}{\Delta y} + \overline{u_y} + \frac{3}{80}\Delta y[u_{yy}].$$

The errors are reported in Table 13.

Test case 3. For $m = 3$, the numerical flux $\widehat{J}_i$ is

$$\widehat{J}_1|_{x_{j+\frac{1}{2}}} = \frac{225}{32}\frac{[u]}{\Delta x} + \overline{u_x} + \frac{1}{64}\Delta x[u_{xx}], \quad \textit{and}$$
$$\widehat{J}_2|_{y_{k+\frac{1}{2}}} = \frac{225}{32}\frac{[u]}{\Delta y} + \overline{u_y} + \frac{1}{64}\Delta y[u_{yy}].$$

The errors are reported in Table 14.

Example 3.6. In this example, we study the same problem as the one in Example 3.5 on $N \times N$ triangular mesh. The DDG scheme [17] is to find $u|_K \in P^m$ such that

Table 8
Errors with numerical flux (3.3), p^m approximations with $m = 4,5,6,7$.

m		$N = 4$	$N = 8$		$N = 12$		$N = 16$		$N = 20$	
		Error	Error	Order	Error	Order	Error	Order	Error	Order
4	$\|u-u_h\|_{L^2}$	2.9923e−06	1.0203e−07	4.87	1.3821e−08	4.93	3.3265e−09	4.95	1.0993e−09	4.96
	$\|u-u_h\|_{H^1}$	6.4207e−05	3.3404e−06	4.26	6.0390e−07	4.22	1.8124e−07	4.18	7.1660e−08	4.16
5	$\|u-u_h\|_{L^2}$	1.7470e−07	2.2853e−09	6.26	1.8836e−10	6.16	3.2450e−11	6.11	8.3390e−12	6.09
	$\|u-u_h\|_{H^1}$	1.7495e−06	3.6140e−08	5.60	3.9912e−09	5.43	8.5715e−10	5.35	2.6323e−10	5.29
6	$\|u-u_h\|_{L^2}$	8.5990e−07	8.1534e−09	6.72	5.0511e−10	6.86	6.945e−11	6.90	1.4926e−11	6.89
	$\|u-u_h\|_{H^1}$	1.6015e−05	2.6451e−07	5.92	2.3299e−08	5.99	4.1416e−09	6.00	1.0866e−09	6.00
7	$\|u-u_h\|_{L^2}$	9.3358e−12	3.5815e−14	8.03	1.5348e−15	7.77	2.9920e−16	-	2.0407e−16	-
	$\|u-u_h\|_{H^1}$	3.0006e−10	2.0433e−12	7.20	1.1447e−13	7.11	1.6803e−14	-	1.1706e−14	-

Table 9
L^2 errors with numerical flux (3.4), p^3 approximations.

α	$N = 4$	$N = 8$		$N = 12$		$N = 16$		$N = 20$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
0.0	5.1512e−05	3.0476e−06	4.08	5.9246e−07	4.04	1.8601e−07	4.03	7.5844e−08	4.02
0.1	6.3110e−05	3.6665e−06	4.11	7.1038e−07	4.05	2.2272e−07	4.03	9.0736e−08	4.02
0.2	8.5010e−05	4.7345e−06	4.17	9.1081e−07	4.07	2.8465e−07	4.04	1.1576e−07	4.03
0.3	1.5641e−04	7.2520e−06	4.43	1.3632e−06	4.12	4.2193e−07	4.08	1.7065e−07	4.06
0.4	5.5423e−04	2.7621e−03	−2.32	1.3121e−05	13.19	3.8627e−06	4.25	2.1211e−06	2.69
0.5	4.1448e−04	5.0422e−05	3.04	4.2760e−06	6.09	3.6221e−06	0.58	7.7456e−07	6.91
0.6	4.2588e−04	1.9669e−05	4.44	4.2417e−05	−1.90	1.5366e−06	11.53	6.6733e−07	3.74
0.7	1.5641e−03	9.4358e−06	7.37	2.7085e−05	−2.60	2.3158e−06	8.55	9.3922e−07	4.04
0.8	1.5520e−04	1.3765e−05	3.50	3.1325e−06	3.65	1.0656e−06	3.75	4.5611e−07	3.80
0.9	8.3495e−05	6.5797e−06	3.67	1.3991e−06	3.82	4.5890e−07	3.87	1.9204e−07	3.90
1.0	6.0099e−05	4.5111e−06	3.74	9.4264e−07	3.86	3.0661e−07	3.90	1.2768e−07	3.93
1.1	4.7681e−05	3.4876e−06	3.77	7.2324e−07	3.88	2.3441e−07	3.92	9.7412e−08	3.94
1.2	3.9884e−05	2.8723e−06	3.80	5.9319e−07	3.89	1.9190e−07	3.92	7.9661e−08	3.94
1.3	3.4540e−05	2.4622e−06	3.81	5.0723e−07	3.90	1.6391e−07	3.93	6.7996e−08	3.94
1.4	3.0670e−05	2.1708e−06	3.82	4.4644e−07	3.90	1.4416e−07	3.93	5.9776e−08	3.94
1.5	2.7761e−05	1.9544e−06	3.83	4.0143e−07	3.90	1.2955e−07	3.93	5.3703e−08	3.95
1.6	2.5512e−05	1.7885e−06	3.84	3.6699e−07	3.91	1.1838e−07	3.93	4.9061e−08	3.95
1.7	2.3736e−05	1.6581e−06	3.84	3.3996e−07	3.91	1.0962e−07	3.93	4.5419e−08	3.95

Table 10
L^2 errors with numerical flux (3.4), p^4 approximations.

α	$N = 4$	$N = 8$		$N = 12$		$N = 16$		$N = 20$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
−0.2	1.3552e−06	4.3911e−08	4.95	5.8583e−09	4.97	1.3992e−09	4.98	4.6023e−10	4.98
−0.1	1.4434e−06	4.6920e−08	4.94	6.2673e−09	4.96	1.4977e−09	4.98	4.9282e−10	4.98
0.0	1.5536e−06	5.0678e−08	4.94	6.7780e−09	4.96	1.6208e−09	4.97	5.3352e−10	4.98
0.1	1.6952e−06	5.5496e−08	4.93	7.4331e−09	4.96	1.7787e−09	4.97	5.8572e−10	4.98
0.2	1.8840e−06	6.1901e−08	4.93	8.3039e−09	4.95	1.9886e−09	4.97	6.5510e−10	4.98
0.3	2.1498e−06	7.0860e−08	4.92	9.5218e−09	4.95	2.2821e−09	4.97	7.5213e−10	4.97
0.4	2.5584e−06	8.4424e−08	4.92	1.1365e−08	4.95	2.7260e−09	4.96	8.9885e−10	4.97
0.5	3.2992e−06	1.0803e−07	4.93	1.4563e−08	4.94	3.4957e−09	4.96	1.1531e−09	4.97
0.6	5.3722e−06	1.6432e−07	5.03	2.2090e−08	4.95	5.2978e−09	4.96	1.7465e−09	4.97
0.7	5.8520e−04	1.4937e−06	8.61	1.1995e−07	6.22	2.4906e−08	5.46	7.7370e−09	5.24
0.8	1.8766e−05	3.9012e−07	5.59	5.3262e−07	−0.77	1.3637e−08	12.74	6.1085e−09	3.60
0.9	1.2267e−04	2.2222e−07	9.11	2.3400e−08	5.55	9.1700e−09	3.26	1.0910e−07	-
1.0	3.3363e−06	7.3360e−08	5.51	1.2641e−08	4.34	2.9288e−08	−2.92	1.0624e−09	14.86
1.1	8.1982e−06	2.0538e−07	5.32	2.5964e−08	5.10	6.0353e−09	5.07	1.9525e−09	5.06
1.2	5.7204e−05	6.1750e−06	3.21	3.3442e−06	1.51	1.5298e−06	2.72	1.8972e−07	9.35
1.3	7.5704e−06	2.6754e−07	4.82	3.6655e−08	4.90	8.8750e−09	4.93	2.9437e−09	4.95
1.4	4.2323e−06	1.4513e−07	4.87	1.9695e−08	4.93	4.7447e−09	4.95	1.5688e−09	4.96
1.5	2.9923e−06	1.0203e−07	4.87	1.3821e−08	4.93	3.3265e−09	4.95	1.0993e−09	4.96
1.6	2.3401e−06	7.9709e−08	4.88	1.0794e−08	4.93	2.5974e−09	4.95	8.5822e−10	4.96
1.7	1.9377e−06	6.6004e−08	4.88	8.9385e−09	4.93	2.1508e−09	4.95	7.1062e−10	4.96

Table 11
L^2 errors with numerical flux (3.4), p^5 approximations.

α	$N=4$	$N=8$		$N=12$		$N=16$		$N=20$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
0.0	2.8905e−08	4.1198e−10	6.13	3.4981e−11	6.08	6.1196e−12	6.06	1.5874e−12	6.05
0.1	3.0479e−08	4.3377e−10	6.13	3.6809e−11	6.08	6.4374e−12	6.06	1.6695e−12	6.05
0.2	3.2280e−08	4.5867e−10	6.14	3.8900e−11	6.09	6.8008e−12	6.06	1.7634e−12	6.05
0.3	3.4358e−08	4.8742e−10	6.14	4.1313e−11	6.09	7.2204e−12	6.06	1.8719e−12	6.05
0.4	3.6786e−08	5.2097e−10	6.14	4.4130e−11	6.09	7.7103e−12	6.06	1.9985e−12	6.05
0.5	3.9663e−08	5.6072e−10	6.14	4.7467e−11	6.09	8.2906e−12	6.07	2.1485e−12	6.05
0.6	4.3144e−08	6.0875e−10	6.15	5.1497e−11	6.09	8.9916e−12	6.07	2.3296e−12	6.05
0.7	4.7482e−08	6.6853e−10	6.15	5.6512e−11	6.09	9.8635e−12	6.07	2.5550e−12	6.05
0.8	5.3190e−08	7.4703e−10	6.15	6.3084e−11	6.10	1.1006e−11	6.07	2.8500e−12	6.05
0.9	6.1743e−08	8.6471e−10	6.16	7.2855e−11	6.10	1.2697e−11	6.07	3.2860e−12	6.06
1.0	1.0455e−07	1.2580e−09	6.38	1.0228e−10	6.19	1.7482e−11	6.14	4.4776e−12	6.10
1.1	4.4269e−08	5.1081e−10	6.44	2.9038e−11	7.07	8.9061e−12	4.11	2.9368e−11	-
1.2	7.4567e−08	9.6931e−10	6.27	7.9131e−11	6.18	1.3500e−11	6.15	3.4388e−12	6.13
1.3	9.5214e−08	1.2788e−09	6.22	1.0610e−10	6.14	1.8326e−11	6.10	4.7151e−12	6.08
1.4	1.2428e−07	1.6597e−09	6.23	1.3758e−10	6.14	2.3765e−11	6.10	6.1161e−12	6.08
1.5	1.7470e−07	2.2853e−09	6.26	1.8836e−10	6.16	3.2450e−11	6.11	8.3390e−12	6.09
1.6	2.8973e−07	3.5980e−09	6.33	2.9232e−10	6.19	5.0023e−11	6.14	1.2805e−11	6.11
1.7	8.4998e−07	8.3008e−09	6.68	6.3911e−10	6.32	1.0679e−10	6.22	2.6970e−11	6.17
1.8	1.3479e−06	3.8366e−08	5.13	4.9560e−09	5.05	1.1834e−09	4.98	3.9501e−10	4.92
1.9	3.1952e−07	5.3347e−09	5.90	4.8079e−10	5.94	8.6806e−11	5.95	2.2963e−11	5.96
2.0	1.871e−07	2.9306e−09	6.00	2.5712e−10	6.00	4.5736e−11	6.00	1.1985e−11	6.00

Table 12
Errors for p^1 approximation.

$m=1$	$N=4$	$N=8$		$N=16$		$N=32$		$N=64$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u-u_h\|_{L^2}$	8.6941e−02	2.3526e−02	1.89	6.5146e−03	1.85	1.7496e−03	1.90	4.5408e−04	1.95
$\|u-u_h\|_{H^1}$	2.0748e−00	1.0443e−00	0.99	5.1031e−01	1.03	2.5274e−01	1.01	1.2603e−01	1.00

Table 13
Errors for p^2 approximation.

$m=2$	$N=4$	$N=8$		$N=16$		$N=32$		$N=64$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u-u_h\|_{L^2}$	3.5660e−02	4.7269e−03	2.92	5.8490e−04	3.01	7.2484e−05	3.01	9.0224e−06	3.01
$\|u-u_h\|_{H^1}$	6.5713e−01	1.3638e−01	2.27	3.0642e−02	2.15	7.2967e−03	2.07	1.7843e−03	2.03

Table 14
Errors for p^3 approximation.

$m=3$	$N=4$	$N=8$		$N=16$		$N=32$		$N=64$	
	Error	Error	Order	Error	Order	Error	Order	Error	Order
$\|u-u_h\|_{L^2}$	3.1117e−03	1.5397e−04	4.34	7.1375e−06	4.43	3.7690e−07	4.24	2.2153e−08	4.09
$\|u-u_h\|_{H^1}$	8.4374e−02	8.6426e−03	3.29	9.1587e−04	3.24	1.0794e−04	3.08	1.3281e−05	3.02

$$\iint_K (\nabla u \nabla v + u v) dxdy - \int_{\partial K} \widehat{u_{\boldsymbol{n}}} v^{int_K} ds + \frac{1}{2} \int_{\partial K} [u] v_{\boldsymbol{n}}^{int_K} ds = \iint_K f v dxdy, \tag{3.7}$$

where $n=(n_x,n_y)$ is the outward normal unit along the cell boundary ∂K, v^{int_K} denotes v evaluated from inside K and $u_n = \nabla u \cdot n$ is differentiation in n direction. The numerical flux is of the form:

$$\widehat{u_{\boldsymbol{n}}} = \frac{9}{4}\frac{[u]}{h} + \overline{u_x n_x + u_y n_y}, \quad \text{for} \quad m=1,$$
$$\widehat{u_{\boldsymbol{n}}} = \frac{15}{4}\frac{[u]}{h} + \overline{u_x n_x + u_y n_y} + \frac{3}{80} h\left([u_{xx}] n_x^2 + 2[u_{xy}] n_x n_y + [u_{yy}] n_y^2\right), \quad \text{for} \quad m=2,$$
$$\widehat{u_{\boldsymbol{n}}} = \frac{225}{32}\frac{[u]}{h} + \overline{u_x n_x + u_y n_y} + \frac{1}{64} h\left([u_{xx}] n_x^2 + 2[u_{xy}] n_x n_y + [u_{yy}] n_y^2\right), \quad \text{for} \quad m=3,$$

Table 15
Errors on triangle mesh, p^m approximations with $m = 1,2,3$.

m		$N = 4$	$N = 8$		$N = 16$		$N = 32$		$N = 64$	
		Error	Error	Order	Error	Order	Error	Order	Error	Order
1	$\|u-u_h\|_{L^2}$	1.0912e−01	3.9458e−02	1.47	1.1739e−02	1.75	3.1447e−03	1.90	8.0822e−04	1.96
	$\|u-u_h\|_{H^1}$	2.3812e−00	1.2923e−00	0.88	6.5094e−01	0.99	3.2507e−01	1.00	1.6248e−01	1.00
2	$\|u-u_h\|_{L^2}$	1.8606e−02	2.2755e−03	3.03	2.7150e−04	3.07	3.2954e−05	3.04	4.0532e−06	3.02
	$\|u-u_h\|_{H^1}$	8.1619e−01	2.2212e−01	1.88	5.6835e−02	1.97	1.4296e−02	1.99	3.5802e−03	2.00
3	$\|u-u_h\|_{L^2}$	5.0229e−03	3.5181e−04	3.84	2.2769e−05	3.95	1.4370e−06	3.99	8.9576e−08	4.00
	$\|u-u_h\|_{H^1}$	1.3653e−01	1.5831e−02	3.11	1.8054e−03	3.13	2.1596e−04	3.06	2.6570e−05	3.02

where $h = l/2$ with l is the length of the interface, and

$$[u] = u^{ext_K} - u^{int_K} \quad \text{and} \quad \overline{u_{\boldsymbol{n}}} = \frac{u_{\boldsymbol{n}}^{ext_K} + u_{\boldsymbol{n}}^{int_K}}{2}.$$

Here u^{ext_K} represents u evaluated from outside of K. Errors and orders are reported in Table 15. We obtain $(m+1)$th order accuracy with P^m polynomial approximations.

4. Conclusion

In this paper, we proposed a novel recovery algorithm of normal derivatives for a smooth function from its piecewise L^2 projection, and then constructed simple, easy to implement DDG methods for elliptic equations with each numerical flux selected from the corresponding recovered formula. We numerically showed that the recovered fluxes lead to the desired $m+1$ order accuracy for P^m elements for $m \leqslant 4$. The recovery formulae apply well to one dimensional non-uniform meshes, as well as to two dimensional triangular meshes. For even higher order elements we exploit a numerical flux formula based on the recovery for $m = 2$ or $m = 3$, with some proper adaptation. Numerical tests for polynomials of degree up to seven are presented to valid this extension. Our numerical results show that the numerical flux with a large set of matched (β_0, β_1) pairs gives the desired accuracy with only a few exceptions. We expect that a large set of numerical flux of this type will work for more general settings, and the detailed analysis is left for a future work.

Acknowledgments

Huang's research is supported in part by the NSFC Key Project (11031006) and Hunan Provincial NSF Project (10JJ7001). Liu's research was partially supported by the National Science Foundation under Grant DMS09-07963. Yi's research was partially supported by Hunan Education Department Key Project 10A117 and Hunan Provincial Innovation Foundation for Postgraduate (Grant No. S2008yjscx05).

References

[1] D.N. Arnold, An interior penalty finite element method with discontinuous elements, SIAM J. Numer. Anal. 19 (4) (1982) 742–760.
[2] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2001/2002) 749–1779.
[3] G.A. Baker, Finite element methods for elliptic equations using nonconforming elements, Math. Comput. 31 (1977) 45–59.
[4] F. Brezzi, G. Manzini, D. Marini, P. Pietra, A. Russo, Discontinuous Galerkin approximations for elliptic problems, Numer. Methods Partial Differ. Equat. 16 (2000) 365378.
[5] C.E. Baumann, J.T. Oden, A discontinuous hp finite element method for convection-diffusion problems, Comput. Methods Appl. Mech. Eng. 175 (3–4) (1999) 311–341.
[6] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (2) (1997) 267–279.
[7] J.H. Bramble, A.H. Schatz, Higher order local accuracy by averaging in the finite element method, Math. Comput. 31 (1977) 74–111.
[8] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous Galerkin, mixed and continuous Galerkin methods for second order elliptic problems, SIAM J. Numer. Anal. 47 (2009) 1319–1365.
[9] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time-dependent convection-diffusion systems, SIAM J. Numer. Anal. 35 (6) (1998) 2440–2463.
[10] Y. Cheng, C.-W. Shu, A discontinuous Galerkin finite element method for time dependent partial differential equations with higher order derivatives, Math. Comput. 77 (2007) 699–730.
[11] S. Franz, L. Tobiska, H. Zarin, A new approach to recovery of discontinuous Galerkin, J. Comput. Math. 27 (2009) 697–712.
[12] G. Gassner, F. Lörcher, C.D. Munz, A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes, J. Comput. Phys. 224 (2) (2007) 1049–1063.
[13] B. Heimsund, X. Tai, J. Wang, Superconvergence for the gradient of the finite element approximations by L^2-projections, SIAM J. Numer. Anal. 40 (2002) 1538–1560.
[14] Y. Huang, N. Yi, The superconvergent cluster recovery method, J. Sci. Comput. 44 (2010) 301–322.
[15] B. van Leer, S. Nomura, Discontinuous Galerkin for diffusion, in: Proceedings of 17th AIAA Computational Fluid Dynamics Conference (June 6 2005), AIAA-2005-5108.
[16] H. Liu, J. Yan, The direct discontinuous Galerkin (DDG) method for diffusion problems, SIAM J. Numer. Anal. 47 (1) (2009) 675–698.

[17] H. Liu, J. Yan, The direct discontinuous Galerkin (DDG) method for diffusion with interface corrections, Commun. Comput. Phys. 8 (3) (2010) 541–564.
[18] J.T. Oden, I. Babuska, C.E. Baumann, A discontinuous hp finite element method for diffusion problems, J. Comput. Phys. 146 (2) (1998) 491–519.
[19] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation. Technical Report Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.
[20] M. van Raalte, B. van Leer, Bilinear forms for the recovery-based discontinuous Galerkin method for diffusion, Commun. Comput. Phys. 5 (2009) 683–693.
[21] C.-W. Shu, Different formulations of the discontinuous Galerkin method for the viscous terms, in: Z.-C. Shi, M. Mu, W. Xue, J. Zou (Eds.), Advances in Scientific Computing, Science Press, China, 2001, pp. 144–155.
[22] M. Tabbara, T. Blacker, T. Belytschko, Finite element derivative recovery by moving least square interpolants, Comput. Methods Appl. Mech. Eng. 117 (1994) 211–223.
[23] M.F. Wheeler, An elliptic collocation-finite element method with interior penalties, SIAM J. Numer. Anal. 15 (1978) 152–161.
[24] Z. Zhang, A. Naga, A new finite element gradient recovery method: superconvergence property, SIAM J. Sci. Comput. 26 (4) (2005) 1192–1213.
[25] O.C. Zienkiewicz, J.Z. Zhu, The supercovergent patch recovery and a posteriori error estimates, Int. J. Numer. Methods Eng. 33 (1992) 1331–1382.